TR l à NG ¤ I H ĐC BÁCH KHOA À N ¸ NG
KHOA CÔNG NGH ÊTHÔNG TIN
¾¾ Â ¾¾

GIÁO TRÌNH MÔN H Ư C

H¨ TH´ NG M ýNG LINUX

Ths. NGUY ÈN T ¨N KHÔI

(Lưu hành nội bộ)
à N ¹ng – 2004

0 éc l éc

Bài 1 ĐĂNG NHẬP HỆ THỐNG LINUX

1.1. Truy cập vào máy tính đã cài đặt hệ điều hành Linux

Khởi động máy đã cài đặt Linux, xuất hiện dấu nhắc khởi động hệ điều hành:

```
Boot : linux
```

Khi HệH Linux khởi động, xuất hiện dấu nhắc truy cập hệ thống :

```
login :
password :
```

Người dùng nhập vào username và password tương ứng, trên màn hình xuất hiện dấu nhắc của hệ thống như sau :

```
[user12@linux user12]$
```

1.2. Sử dụng Telnet để truy cập vào máy Linux từ xa

Truy cập vào Server LINUX từ máy Windows. Yêu cầu máy Windows đã cài đặt mạng. để kiểm tra hệ thống mạng, từ dấu nhắc của lệnh trên Windows, gõ lệnh :

```
C:\>ping 200.201.202.180
```

Nếu trên màn hình xuất hiện : `Reply from 200.201.202.180 …`

thì nghĩa là máy tính có khả năng truy cập vào Server LINUX, ngược lại, nếu có thông báo nào khác thông báo như trên thì nên kiểm tra lại cấu hình mạng trên máy. Tiếp theo, ta gõ lệnh :

```
telnet 200.201.202.180
```

Sau một khoảng thời gian thiết lập liên kết, trên của sổ telnet xuất hiện :

```
login :
password :
```

- Người dùng nhập vào username và password tương ứng.

Ví dụ : năng nhập vào với tài khoản *user12*, trên màn hình xuất hiện như sau :

```
login: user12
Password:
Last login: Wed Apr  7 08:35:50 from 131.16.16.21
[user12@linux user12]$
```

1.3. Thoát khỏi Hệ thống

Thoát khỏi phiên làm việc : `#exit` hoặc `#logout`

Chấm dứt hoạt động của hệ thống : `#shutdown –h now`

Bài 2 **6 ðD èNG E-Mail**

Thđ Ľi n t hi n nay Ľang tr thành phđhng ti n chính Ľ liên l c trên m ng. Thđ Ľi n t d s d ng, ti n l i và nhanh chóng. Trong ph n này ta s d ng đ ch v sendmail c a h th ng Linux.

2.1. G ã th mb µng sendmail

Cú pháp : mail <address1> <address2> <address3> . . .

```
$mail user01 root
```

§ Ti p theo, trên màn hình xu t hi n

```
Subject :
```

§ B n gõ vào ch Ľ b c thđ. Nh n Enter, b t Ľ u nhdp vào n i dung thđ.

§ Sau khi nhdp vào n i dung thđ, nh n CTRL-D Ľ g i thđ Ľi.

§ Trên màn hình xu t hi n :

```
CC :
```

§ Nhdp vào tên nh ng ngđ i cùng nhdn thđ hoHc nh n Enter Ľ b qua.

2.2. Nh ±n th m

§ Khi có thđ Ľ n, trên màn hình xu t hi n thông báo :

```
You have mail
```

§ n Ľ c thđ, gõ vào l nh : **$mail**

§ Trên màn hình sΕ li t kê các b c thđ theo th t 1, 2, 3 ... n Ľ c n i dung thđ nào, gõ vào s th t c a b c thđ Ľó.

§ D u & nhđc r ng b n Ľang ' chđhng trình Ľ c thđ.

§ n xóa thđ Ľang Ľ c, t i d u nhđc b n gõ : **&d**

§ n thoát chđhng trình Ľ c thđ, t i d u nhđc b n gõ : **&q**

Ví d M t phiên g i mail c a user12 :

```
[user12@linux user12]$ mail user15 root
Subject: Chao ban
        Thuc hanh LINUX
Cc:
[user12@linux user12]$
```

2.3. Các thao tác h Ût r ç

- n h y b thđ tr c khi g i, b n nh n CTRL-C hai l n.
- n ` c n i dung m t tdp tin trên thđ m c hi n hành vào mail : **~r filename**

- Thay L˘ i ch¸ L?Q c¸ a th¢ : ~s
- Xem t?t c? các th¢ l¢u trong h¸ p th¢ : $more mbox

Các l?nh thao tác trên sendmail

t <message list>	type messages
n	goto and type next message
e <message list>	edit messages
f <message list>	give head lines of messages
d <message list>	delete messages
s <message list>	file append messages to file
u <message list>	undelete messages
R <message list>	reply to message senders
r <message list>	reply to message senders and all recipients
pre <message list>	make messages go back to /usr/spool/mail
p <message list>	print message
m <user list>	mail to specific users
q	quit, saving unresolved messages in mbox
x	quit, do not remove system mailbox
h	print out active message headers
!	shell escape
cd [directory]	chdir to directory or home if none given

Bài 3 CÁC LỆNH TRÊN LINUX

3.1. Tổ chức hệ thống tệp tin trên Linux

/etc	Cấu hình hệ thống của máy
/usr/bin	Chứa hầu hết các lệnh người dùng.
/dev	Các tập tin thiết bị.
/usr/man	Chứa các tài liệu trực tuyến.
/usr/include	Chứa các tập tin include chuẩn của C.
/var/log	Các tập tin lưu giữ thông tin làm việc hiện hành của người dùng.
/home	Chứa các thư mục con của các user.
/usr/lib	Chứa các tập tin thư viện của các chương trình người dùng.

Khi truy cập vào hệ thống, thư mục làm việc của người dùng được xem như là thư mục chủ. Ví dụ : Thư mục chủ của user01 sẽ là `/home/user01`

Nếu đường dẫn bắt đầu bằng dấu "/", hệ thống xem đó như một tên đường dẫn đầy đủ bắt đầu từ thư mục gốc.

3.2. Các lệnh thao tác trên hệ thống tệp tin

Các tham số luôn bắt đầu bằng dấu "-", và trong hầu hết các trường hợp nhiều tham số một chữ cái có thể kết hợp dùng một dấu "-".

Ví dụ: Thay vì dùng lệnh `ls -l -F`, ta có thể dùng lệnh tương đương `ls -lF`.

Kí tự	Chức năng
*?[ ]	Kí tự đại diện hay theo mẫu
&	Chạy ngầm định ở chế độ nền (background), trả lại dấu nhắc hệ thống cho các tác vụ khác .
;	Dấu phân cách nhiều lệnh trên một dòng lệnh.
\	Tắt tác dụng của ký tự đặc biệt như ***, ?, [,], &, ;, >, <, |**
>	định hướng dữ liệu xuất ra file.
<	định hướng dữ liệu nhập từ file.
>>	định hướng dữ liệu xuất ra cuối file nếu file đã tồn tại.
|	định hướng dữ liệu xuất là dữ liệu nhập cho lệnh tiếp theo.
$	Sử dụng biến môi trường.

3.2.1. Tạo một thư mục

Cú pháp : `mkdir <dir1> <dir2> ... <dirN>`

`<dir1> . . . <dirN>` là tên các thư mục cần tạo.

```
[user01@linux user01]$ mkdir baitap
```

```
[user01@linux user01]$ mkdir document
[user01@linux user01]$ mkdir baitap\ltc
[user01@linux user01]$ ls
[user01@linux user01]$ mkdir baitap/ltc
[user01@linux user01]$ mkdir baitap/perl
```

3.2.2. Thay ÿ Ù th mm éc hi Ëh ành

Cú pháp : cd <directory>

<directory> là th Ø m c mu´ n chuyWn L Jn.

. : yêu c Fu chuyWn L Jn th Ø m c hi n hành.

. . : chuyWn L Jn th Ø m c cha.

```
[user01@linux user01]$ cd baitap
[user01@linux user01]$ cd /home
[user01@linux user01]$ cd
```

3.2.3. Xem th mm éc làm vi Ëc hi Ëh ành

Cú pháp : **pwd**

```
[user12@linux user12]$ pwd
/home/user12
[user12@linux user12]$
```

3.2.4. Xem thông tin v Åt ±p tin và th mm éc

Cú pháp : ls <file1> <file2> ... <fileN> <Tham s ố>

<file1> . . . <fileN> là danh sách tên tdp tin hay th Ø m c.

<Tham s´ > :

-F : dùng L W hiWn th m t vài thông tin vQ kiWu c a tdp tin

-l : (long) li t kê kích th c tdp tin, ng i t o ra, các quyQn ng i s d ng.

```
[user12@linux user12]$ ls -lF
total 75
drwxrwxr-x   2 user12     user12       1024 Apr  7 09:41 baitap/
drwxrwxr-x   2 user12     user12       1024 Apr  7 09:41 doc/
-rwxrwxr-x   1 user12     user12      71 Mar 31 10:39 hello*
-rw-rw-r--   1 user12     user12       126 Apr  7 09:26 baitho.txt
-rw-rw-r--   1 user12     user12      70 Apr  7 08:26 hello.c
[user12@linux user12]$
```

ls –lF

ls *a* : hiWn th t Ët c Γ tdp tin hay th Ø m c con có kí t a

ls F*E : hiWn th danh sách b t L Fu b ng F và k Jt thúc b ng E

3.2.5. Di chuy Çn m Ý hay nhi Åu t ±p tin

Cú pháp : mv <file1> <file2> ... <fileN> <destination>

<file1> . . . <fileN> là danh sách tên tdp tin cFn di chuyWn

<destination> là tdp tin hay thư mục Lích.

Lệnh mv có thW dùng LW Lỗi tên tdp tin.

· ChuyWn nhiQu tdp tin

```
$ mv  * directory
```

· Di chuyWn thư mục

```
[user01@linux user01]$ mkdir ctrinh
[user01@linux user01]$ ls –lF
[user01@linux user01]$ mv ctrinh baitap
```

Di chuyWn thư mục /home/user01/ctrinh vào thư mục /home/user01/baitap

3.2.6. Sao chép tệp tin

Cú pháp : cp <source> <destination>

```
[user01@linux user01]$ cd baitap
[user01@linux baitap]$ vi tho.txt
[user01@linux baitap]$ mv tho.txt baitho.doc
[user01@linux baitap]$ ls
baitho.doc  ctrinh     hello.c     ltc         perl
[user01@linux baitap]$ cp baitho.doc ~/document
```

· Sao chép tBt cF các tdp tin vào mJt danh mục

```
$ cp   *   directory
```

3.2.7. Tạo liên kết với tệp tin

TQo liên kJt vJi tdp tin là tQo thêm cho tdp tin tên mJi và Lồng dXn tGhng ng.

Cú pháp : ln <source> <destination>

ls -l : xem số liên kJt cJa tdp tin.

Muốn xóa mJt tdp tin ta phFi xoá tBt cF các liên kJt cJa nó.

```
[user01@linux user01]$ pwd
[user01@linux user01]$ ls  -l
[user01@linux user01]$ ls  -l  baitap
[user01@linux user01]$ ln  baitap/file1   file.link
[user01@linux user01]$ ls  -l  baitap
[user01@linux user01]$ ls  -l  file.link
```

3.2.8. *Tìm kiếm mít tệp tin*

Lệnh find cho phép tìm kiJm mJt hay nhiQu tdp tin trong mJt cây danh mục.

· Tìm theo tên: *find <path> -name <filename>*

· Tìm theo số i-node cJa tdp tin: *find <path> -inum <number>*

· Tìm theo tên ngQời sở hJu : *find <path> -user <username>*

ạW tránh các thông báo lỗi Ľ€a ra màn hình, ta có thW Ľ" i hệ ng Ľ₣u ra lỗi chuỗn (*standard error*) tệ i mỗ t tđp tin rỗ ng (/dev/null):

```
$ find  /  -name  filename  - print  2>/dev/null
```

Ví dụ :

```
$ pwd
/home/user01
$ find / -name ttyc2d1  - print  2>/dev/null
/dev/ttyc2d1
$ ls  -i  /unix
2810  - r- xr- - r- -  2  bin  bin  508516  Mar  10 1989  /unix
$find /  -inum  2810  - print  2>/dev/null
```

3.2.9. Xoá th mm éc r Ûng

Cú pháp : rmdir <dir1> <dir2> ... <dirN>

<dir1> ... <dirN> là tên nhỗ ng thệ mục c�₣n xóa.

rmdir /home/baitap xóa thệ mục /home/baitap

3.2.10. Xóa các tɔp tin hoỉc thỗ mλc

Cú pháp : rm <file1> <file2> ... <fileN>

3.2.11. Xem hỗ αng dỗ n sơ dλng lM̂nh

Cú pháp : man <command>
ho?c <command> --help
<command> /?

Trong Ľó <command> là tên cụ a mỗ t c�₣n xem hệ ng dẊn.

```
[user12@linux user12]$ man ls
[user12@linux user12]$ cp --help
[user12@linux user12]$ cp --help >cp.txt
```

3.2.12. Hi Çi th Ïn Ý c ëa các t ±p tin

Cú pháp : more <file1> <file2> ... <fileN>

<file1> <file2> ... <fileN> là nhỗ ng tđp tin c�₣n hiWn thị.

```
[user12@linux user12]$more baitho.txt          // hiWn thị tđp tin baitho.txt
[user12@linux user12]$more mbox          // Xem tËlt cΓ thệ lệu trong hỗ p thệ
```

3.2.13. N Õcác t ±p tin

Cú pháp : cat <file1> <file2> ... <fileN> [>filename]

Lệnh dùng ĽW hiWn thị toàn bỗ nỗ i dung cụ a nhiQu tđp tin cùng mỗ t lúc.

<file1> <file2> ... <fileN> là nhỗ ng tđp tin c�₣n hiWn thị nỗ i dung.

Ví dụ :

Hiển thị nội dung hai tập tin baitho.txt và vanban.doc

```
$cat baitho.txt vanban.doc
```

Kết nối nội dung hai tập tin baitho.txt và vanban.doc vào tập tin thop.doc

```
$cat baitho.txt vanban.doc >thop.doc
```

3.2.14. Xuất nội dung thông báo

Cú pháp : echo <arg1> <arg2> … <argN>

Trong đó <arg1> <arg2> … <argN> là các đối số dòng lệnh.

```
[user12@linux user12]$ echo "Chao cac ban"
Chao cac ban sinh vien"
[user12@linux user12]$echo *                    ® Hiển thị nội dung thư mục
```

3.2.15. Nén và giải nén tập tin

Cú pháp : gzip <filename>

Nén một tập tin. Tên tập tin nén giống như tên ban đầu, kèm theo đuôi .gz

```
[user12@linux user12]$ gzip vanban.txt          -> vanban.txt.gz
```

Cú pháp : gunzip <filename>
gzip -d <filename>

Lệnh dùng để giải nén tập tin.

```
[user12@linux user12]$gunzip vanban.txt.gz
```

3.3. Các lệnh hệ thống

3.3.1. Lệnh at

Thực hiện lệnh theo thời gian định trước

```
[user12@linux user12]$ at 8:15am Feb 27
echo Happy birthday | mail emily  <CR>
<^d>
[user12@linux user12]$atrm jobnumber      xóa lệnh trong hàng đợi
[user12@linux user12]$at -l               hiển thị danh sách các lệnh trong hàng đợi
```

3.3.2. Lệnh hostname

Hiển thị tên máy tính đang làm việc.

Hệ thống lưu thông tin vQ tên máy trong tập tin /etc/hosts

```
[user12@linux user12]$ hostname
linux.edu.vn
```

3.3.3. Lệnh ps

Xem danh sách các tiến trình đang hoạt động trên hệ thống.

```
[user12@linux user12]$ ps
 PID TTY STAT TIME COMMAND
```

```
4516  p4 S    0:00 -bash
4703  p4 S    0:00 /usr/bin/mc -P
4705  r0 S    0:00 bash -rcfile .bashrc
4727  r0 R    0:00 ps
[user12@linux user12]$ kill 4703          //Hủy bỏ tiến trình mc có số hiệu 4703
Terminated
```

3.3.4. Lệnh clear

Xóa màn hình.

3.3.5. Lệnh date

Hiển thị ngày tháng hiện hành của hệ thống

3.3.6. Lệnh cal <month> <year>

Xem lịch tương ứng với tháng và năm chỉ định.

3.3.7. Lệnh mount

Cú pháp : mount [-t <type>] <device> <mountpoint>

- Lệnh dùng để kết nối hệ điều hành với các thiết bị khác trên hệ thống.
- Lệnh này *chỉ thực hiện* được khi bạn vào hệ thống với tư cách là root.

type : Kiểu tệp tin

device : Tệp tin điều khiển thiết bị kết nối.

mountpoint : Vị trí thư mục trên hệ điều hành dùng để kết nối với file thiết bị.

- Tạo kết nối với đĩa logic 1 : #mount /dev/hda1 /mnt/hdisk
- Tạo kết nối với đĩa mềm MS-DOS: #mount /dev/fd0 /mnt/floppy
- Tạo kết nối với đĩa mềm LINUX : #mount -t ext2 /dev/fd0 /mnt/floppy
- Tạo kết nối với đĩa CDROM : #mount /dev/hda1 /mnt/cdrom
- Hủy kết nối với đĩa mềm : #umount /dev/fd0

Chú ý : ***Hệ thống Linux xem các thiết bị kết nối như các một tệp tin đặc biệt.***

3.3.8. Tiện ích mc

Tiện ích mc trên Linux có giao diện làm việc giống như trình NC Command của MS - DOS. Để khởi động mc gõ lệnh như sau :

```
#mc
```

3.3.9. TiMn ích máy tính bc

ChƠhng trình *bc* cung cBÏp m¸ t b¸ máy tính tay giúp ngÊ i dùng có thW tính toán các biWu th¸ c, các hàm toán h¸ c ...

3.3.9.1. Khởi động bc

T¸ dBÏu nhdc h¸ th´ ng, bÜn gõ :

```
#bc ¿
```

xuBÏt hi¸ n dBÏu nhdc, bÜn có thW nhdp vào các biWu th¸ c tính toán :

```
(4+5)*(12-10) ¿
18
1000000000000*1000000000000¿
1000000000000000000000000
```

¸nW BÏn L¸ nh s´ ch¸ s´ thdp phân, dùng l¸ nh scale = n :

```
scale=3¿
1/6¿
.166
```

Ldp trình trong bc.

```
define giaithua(n)
{
if (n<=1) return (1);
else return (gt(n-1)*n);
}
gt(5)
120
```

¸nW chuyWn L¸ i sang các ch s´ khác nhau, dùng các l¸ nh ibase và obase

ibase=chs´ ¸n¸ nh dÜng ch s´ L¸Fu vào

obase=chs´ ¸n¸ nh dÜng ch s´ xuBÏt ra

ibase và obase ngF¸m L¸ nh là ch s´ 10

```
ibase=16¿
FF¿
255
obase=2¿
FF¿
11111111
ibase¿
obase¿
```

3.3.9.2. Kết thúc bc

NhBÏn CTRL-D LW thoát.

#man bc -> Xem các hE¸ ng dX¸n s¸ d¸ ng bc.

Bài 4 QUẢN LÝ TÀI KHOẢN VÀ PHÂN QUYỀN SỬ DỤNG

Mô tả cách thức bảo vệ tập tin của LINUX: người sử dụng, nhóm người sử dụng, các quyền truy xuất trên tập tin.

4.1. Quản lý tài khoản của hệ thống

4.1.1. Tài khoản người dùng

Mỗi người sử dụng trên hệ thống được mô tả qua các thông tin sau:

- username : tên người sử dụng
- password : mật khẩu (nếu có)
- uid : số nhận dạng (user identify number)
- gid : số của nhóm (group identify number)
- comment : chú thích
- Thư mục chủ của tài khoản (home directory)
- Shell làm việc (chương trình chạy lúc bắt đầu phiên làm việc)

Các thông tin trên được chứa trong tập tin */etc/passwd*

4.1.2. Tài khoản nhóm người dùng

Một nhóm người sử dụng được mô tả bằng các thông tin sau:

- groupname : tên của nhóm
- gid : số của nhóm (gid: group identify number)
- danh sách các tài khoản thuộc nhóm

Các thông tin trên được chứa trong tập tin */etc/group*

4.2. Phân quyền người dùng trên hệ thống tập tin

4.2.1. Các quyền truy xuất trên tập tin

Khi tập tin được tạo lập, các thông tin sau đây sẽ được ghi lại:

- ***uid*** của người tạo tập tin
- ***gid*** của người tạo tập tin
- Các quyền thâm nhập tập tin khác . . .
- Tập tin được bảo vệ bởi một tập hợp các bit định nghĩa quyền thâm nhập

```
r w x       r w x       r w x
suid        sgid
owner       group       other
```

Trong đó:

r QuyQn L c n i dung tdp tin, th m c

w QuyQn tUo và xoá n i dung tdp tin, tUo và xóa tdp tin trong th m c

x QuyQn th c thi tdp tin. QuyQn truy xubt qua lUi trên th m c.

- *Các quyền với thư mục chỉ có hiều lực ở một mức nhất định, thư mục con có thể được bảo vệ trong khi thư mục cha thì không.*
- *Lệnh **ls -lF** liệt kê danh sách các tập tin và các thuộc tính của chúng trong một danh mục, qua đó ta có thể xem các thông tin như loại tập tin, quyền truy nhập, người sở hữu và kích thước của tập tin. . .*

4.2.2. L nh chmod

L nh **chmod** cho phép thay L i quyQn trên tdp tin c a ngd i dùng. Ch nh ng ngd i s h u tdp tin này m i có thW thay L i L c m c LHc quyQn L i v i tdp tin này. Có thW th c hi n l nh theo hai cách:

4.2.2.1. Dùng các ký hiệu tượng trưng:

Cú pháp : chmod {a,u,g,o}{+,-,=}{r,w,x} <filename>

Trong Ló : u (user), g (group), o (other), a (all)

Các toán t : + thêm quyQn. - b t quyQn. = gán giá tr khác

4.2.2.2. Dùng thông số tuyệt đối

Cú pháp : chmod <mode> <filename>

trong Ló mode là m t s ch s 8 (octal)

```
    r w x        r - x        r - -
    1 1 1        1 0 1        1 0 0
      7            5            4
$chmod 754 filename
$chmod g-w,o+r  baitho.doc
$chmod a+r baocao.txt
$chmod +r baocao.txt
$chmod og-x baocao.txt          không cho th c thi
$chmod u+rwx baocao.txt         cho phép ngd i s h u có thW L c, viJt và th c thi.
$chmod o-rwx baocao.txt         không cho truy nhdp tdp tin.
$chmod 777  *                   nHt các quyQn cho tBt cT các L i td ng s d ng .
                                trên toàn b tdp tin trong th m c hi n hành
```

4.2.3. Thay yUi ng mi cho »c nhóm s h u t p tin

- L nh **chown** cho phép thay L i ngd i s h u, nhóm s h u trên tdp tin.
- L nh **chgrp** cho phép thay L i nhóm s h u trên tdp tin.

THÔC HÀNH

1. Thay ÿ□i quy Å□n trên t±p tin

```
#cat bai1.sh
#ls -lF bai1.sh
#chmod u+x,g+wx bai1.sh
#ls -lF bai1.sh
#chmod 644 bai1.sh
#ls -lF bai1.sh
#chmod 764 bai1.sh
#ls -lF bai1.sh
#chmod 777 bai1.sh
#ls -lF bai1.sh
```

2. T¥o tài kho§n h Ëth Õng

Tïo nhóm cntt2004

```
#groupadd cntt2004
```

Xem tdp tin /etc/group

```
#cat /etc/group
```

Tïo m¸t account user01 m¸i thu¸c nhóm cntt2004

```
#useradd - g cntt2004 -c "Tai khoan user01" user01
#passwd user01
```

Xem tdp tin /etc/passwd, /etc/shadow

```
#cat /etc/passwd
#cat /etc/shadow
```

Th□ L□ng nhdp vào h□ th□ ng v¸i tài kho□n là user01

Tïo m¸t account user02

```
#useradd user02
#passwd user02
```

ņ□a user02 vào nhóm cntt2004

```
#usermod -g cntt2004 user02
```

Th□ L□ng nhdp vào h□ th□ ng v¸i tài kho□n là user02

Xóa user02

```
#userdel user02
#cat /etc/passwd
```

3. Thay ÿ□i quy Å□ s ñd éng cho các ÿÕt mng trên t±p tin

a. Tïo m¸t tdp tin m¸i /home/baocao.txt

b. ņ□i ch¸ s□ h¸u c¸a tdp tin /home/baocao.txt là user01

```
#chown user01 /home/baocao.txt
```

c. Phân quyQn rwxr--r-- cho các L´ i tê ng trên tdp tin /home/baocao.txt.

```
#chmod 744 /home/baocao.txt
```

d. ņŁng nhdp vào h´ th´ ng v¸ i tài khoƒn user01. Th¨ thay L¨ i n¸ i dung tdp tin /home/baocao.txt.

e. ņŁng nhdp vào h´ th´ ng v¸ i tài khoƒn khác. Th¨ thay L¨ i n¸ i dung tdp tin /home/baocao.txt. Nhdn xét ?

4. Phân quy Å s ñd éng cho các ÿÕ t mqg

a. T¶o nhóm ngê i s¨ d¸ ng có tên cntt2004.

b. B¨ sung các user01, user02 vào nhóm cntt2004.

```
#usermod -g cntt2004 user01
#usermod -g cntt2004 user02
```

c. T¶o thê m¸ c /home/common

```
#mkdir /home/common
```

d. ņ¨ i nhóm s´ h¸ u c¸ a thê m¸ c /home/common là nhóm cntt2004.

```
#chown :cntt2004 /home/common
```

ho#c

```
#chgrp cntt2004 /home/common
```

e. Phân quyQn rwx cho L´ i tê ng nhóm cntt2004 trên thê m¸ c /home/common

```
#chmod g+rwx /home/common
#ls -lF /home
```

f. ņŁng nhdp vào h´ th´ ng v¸ i tài khoƒn user01. T¶o thê m¸ c m¸ i trong /home/common.

g. ņŁng nhdp vào h´ th´ ng v¸ i m¸ t tài khoƒn khác không thu¸ c nhóm cntt2004. Th¨ t¶o thê m¸ c m¸ i trong /home/common. Nhdn xét ?.

Bài 5 6 ðD èNG TRÌNH SO ¤N TH ¦ O VI

Gi¸ i thi¾ u trình so¶n th¶o vi, các thao tác so¶n th¶o tdp tin b¾ng vi.

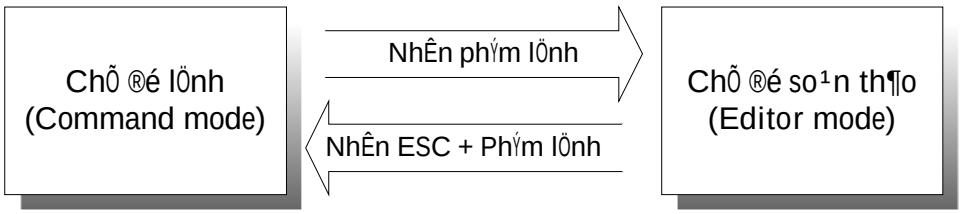
5.1. Gi ßi thi Ëu

vi là ch¢hng trình so¶n th¶o các tdp tin vŁn bÇn trên các h¾ th´ ng Unix :

- Màn hình Ł¢ c xem nh¢ m¸ t c¸ a s¾ m´ trên tdp tin.
- Có khÇ nŁng di chuyW n con tr¸ ŁJn bÖt k8 v¸ trí nào trên màn hình.
- C¸ a s¾ có thW di chuyW n t¸ do trên tdp tin.

PhÇn l¸ n các phím dùng Ł¸ c ldp hoHc kJt h¾ p v¸ i phím Shift và Ctrl ŁW t¶o ra các l¾ nh c¸ a vi. Các l¾ nh c¸ a vi có thW Ł¢ c g¸ i khi có dÖu " : " ¸ dòng cu´ i màn hình.

Có 2 chJ Ł¸ (mode) trong khi s¸ d¸ ng **vi:** *Append mode* và *Command mode*

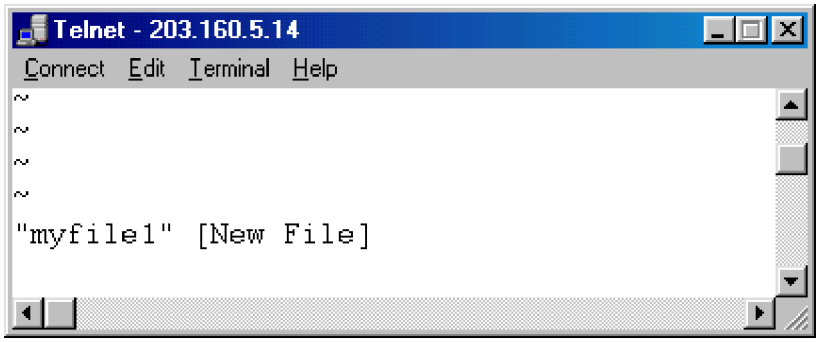


5.2. Kh ãi ÿÝng vi

Ta có thW g¸ i *vi* v¸ i tên tdp tin vŁn bÇn : `$ vi filename`

Ví dụ : `vi bai1.txt <Enter>`

Màn hình so¶n th¶o hi¾ n ra nh¢ sau (*ở đây đang dùng Telnet để nối vào UNIX*) :



- DÖu ngã (~) tr¢ c m¸ i dòng cho biJt dòng Ł¢ó còn r¸ ng (tr´ ng)
- Dòng d¢ i cùng cho biJt tên file Ł¸ang m´ , tr¶ng thái c¸ a file: nJu là file m¸ i thì "[new file]", nJu m´ file cÌ thì sE hiWn th¸ s´ dòng, s´ ký t¸ trong file (hình d¢ i).

5.3. So ¥n th §o v n b §n

- Chèn ký t¸ trên m¸ t dòng a < text > < ESC >
- S¸ d¸ ng các phím so¶n th¶o vŁn bÇn.

- Nhấn phím ESC để kết thúc chế độ chèn văn bản.

5.4. Thoát khỏi vi

Muốn ra khỏi vi và ghi lại nội dung tệp tin, bấm **nhóm phím ESC** và dùng một trong các lệnh như sau:

:ZZ hoặc :wq hoặc :x

Thoát khỏi vi và không ghi lại các thay đổi trên tệp đó

: q!

Khi đang trong chế độ soạn thảo của ***vi***, muốn chạy chương trình shell, dùng lệnh :

: ! <shellcommand>

hoặc gọi shell, sau đó chạy các lệnh của người dùng, khi kết thúc bấm Ctrl-D để trở lại ***vi***:

```
: ! sh
$ <command>
$ Ctrl-D
```

5.4.1. Dùng vi với danh sách các lệnh đã chạy của Shell

Lệnh **fc** (*fix command*) cho phép ta soạn thảo bằng *vi* và chạy lại các lệnh đã chạy của Shell. Cách dùng như sau:

- Soạn thảo và cho chạy lệnh cuối cùng: `$fc`
- Soạn thảo một nhóm lệnh và cho chạy: `$ fc m n`
- Xem danh sách 16 lệnh cuối cùng:

`$ fc -l` hoặc history

`$ fc -lr` (danh sách theo thứ tự ngược lại)

- Tạo một tệp tin chứa một số lệnh đã chạy (của history):
  `$fc -nl n1 n2 > cmd`

cmd là một tệp tin chứa các lệnh của history từ lệnh **n1** đến lệnh **n2**

Bảng tóm tắt các lệnh của *vi*

<i>	Inserts text before cursor
<I>	Enters text at start of line
<a>	Inserts text after cursor
<A>	Enters text at end of line
<o>	Opens a new line below cursor
<O>	Opens a new line above cursor
<d><w>	Deletes word
<d><d>	Deletes entire line

<D>	Deletes to end of line
<x>	Deletes character under cursor
<c><w>	Changes word
<c><c>	Changes line
<C>	Changes to end of line
<R>	Replaces character under cursor
<J>	Joins lines together
<e>	Moves to end of word
<w>	Moves to next word
<$>	Moves to end of line
<l>	Moves one space right
<k>	Moves one line up
<j>	Moves one line down
<h>	Moves one space left
<f><x>	Moves cursor to first occurrence of x
<F><x>	Moves cursor to last occurrence of x
<;>	Repeats the last f/F command
number<|>	Moves cursor to specified column *number*
<H>	Moves cursor to top line on-screen (not top line of file)
<L>	Moves cursor to bottom line on-screen
<M>	Moves cursor to middle line on-screen
<G>	Moves cursor to bottom line of file
number<G>	Moves cursor to specified line *number*
	(same as<ESC>:*number*)
<^>	Moves to beginning of line
<m>*x*	Marks current position with letter *x*
<Ctrl-d>	Scrolls for ward one half of the screen
<Ctrl-u>	Scrolls backward one half of the screen
<Ctrl-f>	Scrolls for ward one screen
<Ctrl-b>	Scrolls backward one screen
<Ctrl-l>	Redraws the screen
<Ctrl-G>	Shows the filename, current line, and column number
<z><z>	Redraws the screen with current line in middle of screen
<y><y>	Yanks entire line into buffer
<p>	Puts contents of buffer below cursor
<P>	Puts contents of buffer above cursor
x"[*number*]"<yy>	Yanks the indicated number of lines into the buffer named *x*
	(*x* can be any single character a–z)

x<p>	Places the contents of buffer *x* after the cursor
:w *[file]*	Writes contents to disk as *file*
:q	*Quits vi*
:q!	Quits file without saving changes
:wq	Saves changes and quits vi
:r *file*	Reads specified *file* into editor
:e *file*	Edits *file*
:!*command*	Executes specified shell *command*
:*number*	Moves to specified line number
:f	Prints out current line and filename (same as <Ctrl-G>)
/*string*	Searches forward for *string*
?*string*	Searches backward for *string*
:*x*,*ys*/*oldstring*/*newstring*	Replaces *oldstring* with *newstring* from line *x* to line *y* (entering y = $ will replace to end of file)
<ESC><u>	Undoes last command
<n>	Finds next occurrence of string. Repeats last command
~	Changes character to opposite case
<ESC>	Switches to command mode

TH ôC HÀNH

1. Dùng chẹhng trình vi Ľ soŸn thĮo tdp tin vanban.doc

```
$vi  vanban.doc
```

2. Sao chép vŁn bĮn

4dd	Cdt 4 dòng và Ľẹa vào vùng Ľ m
Ctrl+d	ChuyWn xu ng cu i vŁn bĮn
p	Sao t vùng Ľ m vào sau dòng hi n hành

3. ņHt và b chJ Ľ hiWn th s dòng :

```
:set  nu
:set  nonu
```

4. Lẹu n i dung tdp tin và thoát kh i vi:

```
:wq
```

5. Xem lai n i dung tdp tin vanban.doc.

Bài 6 **L °P TRÌNH SHELL**

6.1. Ch mkg trình tính t ùg 1-> n

- Minh h a các c u trúc while do done, và cách s d ng [], $(()).

- Tdp tin tong1.sh

```
#!/bin/sh
echo "Chuong trinh tinh tong 1- $1"
index=0
tong=0
while [ $index -lt $1 ]
do
        index=$(($index + 1))
        tong=$(($tong + $index))
done
echo "Tong 1-$1= $tong"
exit 0
```

- Ch y ch hng trình :

```
chmod a+x tong1.sh
./tong1 100
```

6.2. Ch mkg trình tính giai th a c a m Ý s Õ

- Minh h a các c u trúc while do done, và cách s d ng [], $(()).

- Tdp tin giaithua.sh

```
#!/bin/sh
echo "Chuong trinh tinh $1!"
index=0
gt=1
while [ $index -lt $1 ]
do
        index=$(($index + 1))
        gt=$(($gt * $index))
done
echo "$1!= $gt"
exit 0
```

- Ch y ch hng trình :

```
chmod a+x giaithua.sh
./giaithua 5
```

6.3. Ch mkg trình ÿÃm s Õdòng c a m Ý t p tin

- Minh h a các c u trúc if then fi, while do done, và cách s d ng [], $(()).

- Tdp tin demdong.sh

```
#!/bin/sh
echo "Chuong trinh dem so dong cua tap tin $1"
{
n=0
while read line
```

```
do
        n=$(($n + 1))
done
echo "So dong cua tap tin $1 la : $n"
}<$1
exit 0
```

- Chạy chương trình :

```
chmod a+x demdong.sh
./demdong bai1.txt
```

6.4. Chương trình đếm số từ của một tập tin

- Minh họa các cấu trúc for do done, while do done.

- Tập tin demtu.sh

```
#!/bin/sh
echo "Chuong trinh dem so tu cua tap tin $1"
{
n=0
while read line
do
        for wd in $line
        do
                n=$(($n + 1))
        done
done
echo "Tong so tu cua tap tin $1 la : $n"
}<$1
exit 0
```

- Chạy chương trình :

```
chmod a+x demtu.sh
./demtu bai1.txt
```

6.5. Chương trình tìm dòng có độ dài lớn nhất trong một tập tin

- Minh họa các cấu trúc if then fi, while do done.

- Tập tin dongmax.sh

```
#!/bin/sh
echo "Chuong trinh tim dong dai nhat trong tap tin $1"
{
n=0
max=0
dong=""
while read line
do
        n=`expr length "$line"`
        if [ $n -gt $max ]
        then
                dong="$line"
```

```
                max=$n
        fi
done
echo "Dong trong tap tin $1 co do dai max = $max la : $dong"
}<$1
exit 0
```

- Chạy chương trình :

```
chmod a+x dongmax.sh
./dongmax bai1.txt
```

6.6. Chương trình tìm một xâu trong một tập tin

- Minh họa các cấu trúc if then fi, while do done.

- Tập tin timxau.sh

```
#!/bin/sh
echo "Chuong trinh tim xau $1 trong tap tin $2"
{
wordlen=`expr length "$1"`                      # Do dai tu can tim
while read textline
do
        textlen=`expr length "$textline"`       # Do dai cua dong vua doc
        end=$(($textlen - wordlen + 1"
        index=1
        while [ $index -le $end ]
        do
                temp=`expr substr "$textline" $index $wordlen
                if [ "$temp" = $1 ]
                then
                        echo "Tim thay $1 tai dong $textline"
                        break
                fi
                index=$(($index + 1))
        done
done
}<$2
exit 0
```

- Chạy chương trình :

```
chmod a+x timxau.sh
./timxau abc bai1.txt
```

2. Ch⊖hng trình ***sample.c***

```
#include <stdio.h>
void printnum ( int );              /* Khai bỏo hàm*/
void printchar ( char );            /* Khai bỏo hàm */
main () {
        double tmp;                         /* Khai bỏo bi?n toàn c?c */
        tmp = 1.234;
        printf ("%f\n",tmp);                /* In giỏ tr? c?a bi?n toàn c?c tmp */
        printnum (5);                       /* In giỏ tr? s? 5 */
        printf ("%f\n",tmp);                /* In giỏ tr? c?a bi?n toàn c?c tmp */
        printchar ('k');                    /* in ký t? k */
        printf ("%f\n",tmp);                /* In giỏ tr? c?a bi?n toàn c?c tmp */
}
/* é?nh nghia hàm dó khai bỏo ? trờn */
/* Khai bỏo cú t? khoỏ void nghia là hàm khụng tr? v? m?t giỏ tr? */
void printnum (int inputnum) {
        int tmp;
        tmp = inputnum;
        printf ("%d \n",tmp);
}
void printchar (char inputchar) {
        char tmp;
        tmp = inputchar;
        printf ("%c \n",tmp);
}
```

Bài 8 **QU¦N LÝ TI ÂN TRÌNH**

8.1. Gi ßi thi Ëu

TiJn trình là m¸t môi trÒng th¸c hi¾n, bao g°m m¸t phân Ľoün l¾nh và m¸t phân Ľoün d¸ li¾u. CĘn phân bi¾t v¸i khái ni¾m chÐhng trình ch" g°m tdp h° p l¾nh.

Trên h¾ Ľiqu hành Linux, tiJn trình Ľ°c nhdn biJt thông qua s´ hi¾u c¸a tiJn trình, g`i là ***pid***. CÌng nhÐ Ľ´i v¸i user, nó có thW n?m trong nhóm. Vì thJ ĽW phân bi¾t ta nhdn biJt qua s´ hi¾u nhóm g`i là ***pgrp***. M¸t s´ hàm c¸a C cho phép lÜy Ľ°c nh¸ng thông s´ này:

```
int     getpid()      /* trĘ vQ giá tr· int là pid c¸a tiJn trình hi¾n tüi*/
int     getppid()     /*trĘ vQ giá tr· int là pid c¸a tiJn trình cha c¸a tiJn trình hi¾n tüi */
int     getpgrp()     /* trĘ vQ giá tr· int là s´ hi¾u c¸a nhóm tiJn trình*/
int     setpgrp()     /*trĘ vQ giá tr· int là s´ hi¾u nhóm tiJn trình m¸i tüo ra*/
```

Ví d¸ :

L¾nh *:* `printf("Toi la tien trinh %d thuoc nhom %d",getpid(),getgrp());`

KJt quĘ s€ là: `Toi là tien trinh 235 thuoc nhom 231`

8.1.1. T¥o m Ýti Ãn trình - l Ënh fork

`int fork()` tüo ra m¸t tiJn trình con. Giá tr· trĘ lüi là 0 cho tiJn trình con và dÞu hi¾u pid cho tiJn trình cha. Giá tr· s€ là -1 nJu không tüo Ľ°c tiJn trình m¸i.

Theo nguyên tdc ch bĘn c¸a h¾ th´ng, tiJn trình con và cha s€ có cùng Ľoün mã. ¸oün d¸ li¾u c¸a tiJn trình m¸i là m¸t bĘn sao chép chính xác Ľoün d¸ li¾u c¸a tiJn trình cha. Tuy nhiên tiJn trình con vXn khác tiJn trình cha ‘ pid, th° i gian x· lý, ...

8.1.2. D ïng m Ýti Ãn trình

L¾nh kill c¸a Shell có thW dùng ĽW chÞm d¸t hoüt Ľ¸ng c¸a m¸t tiJn trình. ví d¸ nhÐ khi mu´n d¸ng tiJn trình 234 ta dùng l¾nh: **kill 234**

C cÌng có l¾nh kill nhÐ sau:

```
int     kill(pid, sig);
int     pid;          là dÞu hi¾u nhdn biJt c¸a m¸t tiJn trình.
int     sig;          h?ng tín hi¾u giao tiJp tiJn trình.
```

8.1.3. Giao ti Ãp gi óa các ti Ãn trình

Vi ̧c giao tiJp gi ̣a các tiJn trình L'ę̊ c th ̣c hi ̧n thông qua các tín hi ̧u chuXn c ̧a h ̧ th ́ng. Tín hi ̧u là m ̣t s ̣ ngdt quãng logic L'ę̊ c g ̇i L'Jn các tiJn trình b ́i h ̧ th ́ng L'W thông báo cho chúng vQ nh ̣ng s ̣ vi ̧c không bình thể ng trong môi trể ng hoЏt L ̣ng c ̧a chúng (nhę l ̣i b ̣ nh ̣ , l ̣i vào ra). Nó cÌng cho phép các tiJn trình liên lЏc v ̣i nhau. M ̣t tín hi ̧u (tr ̣ SIGKILL) có thW L'ę̊ c xem xét theo ba cách khác nhau:

1. TiJn trình có thW L'ę̊ c b ̣ qua: Ví d ̣ chęhng trình có thW b ̣ qua s ̣ ngdt quãng c ̧a ngể i s ̇ d ̣ng h ̧ th ́ng (L'ó là s ̣ b ̣ qua khi m ̣t tiJn trình L'ang L'ę̊ c s ̇ d ̣ng ́ phFn nQn.
2. TiJn trình có thW L'ę̊ c th ̣c hi ̧n: Trong trể ng h ̊ p này, khi nhdn L'ę̊ c 1 tina sti ̧u, vi ̧c th ̣c hi ̧n 1 tiJn trình L'ę̊ c chuyWn vQ m ̣t quy trình do ngể i s ̇ d ̣ng xác L ̏nh tre ̧c, sau L'ó tr ́ lЏi nhi nó b ̊ ngdt.
3. L ̣i có thW L'ę̊ c tiJn trình tr Γ vQ sau khi nhdn L'ę̊ c tín hi ̧u này.

D ̧i L'ây là m ̣t s ́ tín hi ̧u thể ng gHp:

SIGHUP	Tín hi ̧u này L'ę̊ c phát L'Jn các tiJn trình vào lúc cu ́i khi mà nó t ̣ ngdt. Nó cÌng L'ę̊ c phát L'Jn m ̀ i tiJn trình có tiJn trình chính t ̣ ngdt.
SIGINT	Tín hi ̧u này L'ę̊ c phát L'Jn các tiJn trình khi ta ra l ̊nh ngdt.
SIGQUIT	Tęhng t ̣ nhę trên khi ta gõ vào ^D.
SIGILL	L ̊nh không h ̊ p l ̊, tín hi ̧u L'ę̊ c phát ra khi phát hi ̧n 1 l ̊nh không L'úng ́ cЏp L ̣ vdt lý (ví d ̣ nhę 1 tiJn trình th ̣c hi ̧n m ̣t l ̊nh mà máy tính chông có l ̊nh này).
SIGTRAP	Tín hi ̧u L'ę̊ c phát ra sau m ̣i l ̊nh trong trể ng h ̊ p tiJn trình có s ̇ d ̣ng l ̊nh ptrace().
SIGIOT	BXy L'ę̊ c phát khi có các vЏn L'Q vQ vdt lý.
SIGEMT	BXy c ̧a l ̊nh phát, L'ę̊ c phát ra khi có l ̣i vdt lý trong khi th ̣c hi ̧n.
SIGFPE	n ̣ę̊ c phát ra khi có l ̣i vQ tính toán nhę m ̣t s ́ có dЏu phXy n ́i có L ̏nh dЏng không h ̊ p lý. GFn nhę luôn ch ̈ ra l ̣i khi ldp trình.
SIGKILL	Trang b ̊ L'W kJt thúc tiJn trình. Không thW b ̣ qua hoHc cdt tín hi ̧u này.
SIGBUS	n ̣ę̊ c phát khi gHp l ̣i trên bus.
SYSGEGV	n ̣ę̊ c phát ra khi gHp l ̣i trên phân L'oЏn s ̣ truy cdp d ̣ li ̧u bên ngoài phân L'oЏn d ̣ li ̧u L'ę̊ c cЏp phát cho tiJn trình.

SIGSYS	®-i s- không L'úng cho h- th-ng g- i.
SIGPIPE	Vi□t trên m- t -ng d¥n không m- L'W L□ c.
SIGALRM	Phát ra khi L□°ng h□° ç- a m- t ti□n trình ng- ng l¶i. ®□°ng h□° L'ê c ho¶t L- ng b¾ng l□nh alrm().
SIGTERM	®ê c phát ra khi m- t ti□n trình k□t thúc bình th□ ng. Cì ng có th¥ dùng L'W d- ng 1 h- th-ng L'W k□t thúc t□t cÊ các ti□n trình ho¶t L- ng.

8.1.4. Liên l¥c gi÷a hai ti Ãn trình

T- m- t ch¬hng trình L'hn giÊn d¬- i L'ây s- d-ng các l- nh phát và nhdn tín hi- u, sau L'ó giúp liên l¶c gi- a hai ti□n trình.

N- i dung ç- a ví d- là s- liên l¶c gi- a m- t ti□n trình cha và m- t ti□n trình con thông qua các tín hi- u L'ã L'ê c trình bày phÊn tr¬- c.

```
#include      <errno. h>
#include      <signal. h>
void fils_atc()
{
 printf(" Tien trinh bi loai bo !!!\n");
 kill(getpid(), SIGINT);
}
/***********************************/
void fils()
{
 signal(SIGUSR1, fils_atc);
 printf(" Hinh thanh tien trinh moi. Nhung chuan bi loai bo tien trinh nay !!\n");
 while(1);
 }
/*******************************/
main()
{
int ppid, pid;

if ((pid = fork())==0) fils();
else
  {
    sleep(3);
     printf(" Chap nhan !! Tien trinh se bi loai bo.\n");
     kill(pid, SIGUSR1);
   }
}
```

Trong ví d- trên, ti□n trình con có s- d-ng hàm signal(SIGUSR1, fils_atc). Hàm này có tác d-ng m- i khi ti□n trình con nhdn L'ê c tín hi- u SIGUSR1 thì hàm fils_atc() sÏ L'ê c th- c thi.

Nhđ vdy ́ ví dụ trên mỗ t tiJn trình con Ľã Ľđ̃ c tψo ra nhđng nó lψi không mu ́ n tiJp tục c t̃̉ n tψi. Do vdy sau khi tψm dự ng lψi sleep(3), tiJn trình cha Ľã g ́ i ĽJn cho tiJn trình con mỗ t tín hiệ u là SIGUSR1 bằng lệ nh:

```
kill(pid, SIGUSR1);
```

́ tiJn trình con, tín hiệ u SIGUSR1 Ľã Ľđ̃ c gán vị i hàm fils_atc(). Hàm này ra mỗ t thông báo báo hiệ u tiJn trình này sđp chJt r̃̉ i tự g ́ i ĽJn chính mình (tiJn trình con) tín hiệ u SIGINT, tín hiệ u ngđt tiJn trình. Và tiJn trình con Ľã chJt.

```
kill(getpid(), SIGINT);
```

Mỗ t s ́ nhđ̃ c Ľiwm khi liên lψc trự c tiJp bằng tín hiệ u:

- Mỗ t tín hiệ u có thW bệ bỏ qua, kJt thúc mỗ t tiJn trình hoHc bệ chHn lψi. ọó là lý do chính Ľđa ra các tín hiệ u không thích ợ ng Ľđ̃ c Ľw tiJn hành liên lψc giữ a các tiJn trình. Mỗ t thông Ľiệ p Ľiệ p dđị i hình thự c tín hiệ u có thW sƐ bệ mĐt nJu nó Ľđ̃ c nhdn lúc loψi tín hiệ u này tψm thể i bệ bỏ qua.
- Mỗ t vĐn Ľ̀Q khác là các tín hiệ u có quyQn rĐt lớ n, khi ĽJn chúng làm ngđt quãng công việ c hiệ n tψi. Ví dụ việ c nhdn mỗ t tín hiệ u trong khi tiJn trình Ľang Ľ̉ i mỗ t sự kiệ n (mà có thW ĽJn khi sử dụ ng các lệ nh open(), read(), ...) làm cho việ c thự c thi hàm bệ chệ ch hđ ng. Khi trở lψi, lệ nh chính bệ ngđt g ́ i lψi mỗ t thông Ľiệ p báo lỗ i mà hoàn toàn không xử lý Ľđ̃ c.

Ngoài việ c liên lψc trự c tiJp nhđ ́ ví dụ trên, còn cho phép mỗ t phđhng pháp liên lψc giữ a các tiJn trình khác, Ľó là liên lψc qua "Ľể ng ́ ng".

8.2. L ±p trình ÿa ti Ãn trình

8.2.1. Õ¤g d ¯n liên l ¥c

́ ng dXn là mỗ t ch chJ ch bĽn Ľw liên lψc gián tiJp giữ a các tiJn trình. ọó là các file ĽHc biệ t (FIFO), ́ Ľó các thông tin Ľđ̃ c truyQn Ľi 1 Ļ̌u và thoát ra ́ mỗ t Ļ̌u khác.

Mỗ t s ́ ĽHc Ľiwm cự a "́ ng dXn":

- Các ́ ng dXn chệ mang tính chĐt tψm thể i, chệ tồ n tψi trong thể i gian thự c hiệ n cự a mỗ t tiJn trình tψo ra nó.
- Mu ́ n tψo ra mỗ t ́ ng dXn phĽi bđt Ļ̌u bằng mỗ t lệ nh Ľ̀Hc biệ t: ***pipe()***.
- NhiQu tiJn trình có thW viJt và Ľọ c trên cùng mỗ t ́ ng dXn. Tuy nhiên, không có mỗ t ch chJ nào Ľw phân biệ t thông tin cho các tiJn trình ́ Ļ̌u ra.
- Dung lđ̃ ng ́ ng dXn bệ hψn chJ (khoĽng 4KB). Do Ľó khi chúng ta c ́ gđng viJt khi ́ ng dXn bệ Ļ̌y thì sƐ gHp phĽi trể ng hồ p tđc nghƐn.

- Các tiJn trình liên lQc qua ´ ng dXn phQi có m´ i quan h¸ h` hàng và các ´ ng dXn n´ i phQi LQ° c m‘ trQ¸ c khi tQo ra các tiJn trình con.
- Không thW t¸ thay L° i v· trí thông tin trong ´ ng.

8.2.2. Thao tác v ßi " Õng d ¯n liên l ¥c"

TQo m¸ t ´ ng dXn:

```
int       p_desc[2];
int       pipe(p_desc);
```

Giá tr· trQ vQ là 0 nJu thành công, -1 nJu thÐt bQi.

p_desc[0] : ch¸ a các s´ hi¸ u mô tQ nh° LÓ có thW L° c trong ´ ng dXn.

p_desc[1] : ch¸ a các s´ hi¸ u mô tQ nh° LÓ có thW viJt trong ´ ng dXn.

NhQ vdy vi¸ c viJt trong p_desc[1] là LW truyQn d¸ li¸ u trong ´ ng và vi¸ c L° c trong p_desc[0] LW nhdn chúng.

Ví d¸ :

```
#include <errno. h>
#include <signal. h>
main()
{
int     i,ret, p_desc[2];
char    c;
pipe(p_desc);
write(p_desc[1], "AB", 2);
for (i=1; i<=3,i ++) {
        ret=read(p_desc[0], &c, 1);
        if (ret == 1)
          printf(" Gia tri:  %c\n",c);
        else
           perror("Loi ong dan rong");
        }
}
```

Ví d¸ trên ch" ra r¦ng ta có thW truyQn và nhdn thông tin trên ´ ng dXn. Chúng ta Lã dùng hàm read() và write() LW viJt (truyQn) và L° c (nhdn) trên ´ ng dXn.

8.2.3. Liên l ¥c gi óa ti Ãn trình cha và ti Ãn trình con

Trong ví d¸ dQ¸ i Lây, m¸ t tiJn trình tQo ra m¸ t ´ ng dXn, tQo ra m¸ t tiJn trình con, viJt m¸ t vŁn bQn vào ´ ng dXn.TiJn trình con th¸ a hQ‘ ng ´ ng dXn và các ký hi¸ u mô tQ c¸ a ´ ng dXn, th¸ c hi¸ n L° c trong ´ ng dXn:

```
#include <errno. h>
#include <stdio. h>

void code_fils(int number)   {
 int     fd, nread;
char    texte[100];
```

```
fd=number;
printf(" So hieu mo ta la %d\n",fd);
switch (nread=read(fd, texte, sizeof(texte)))
{
case -1:
        perror("Loi doc.");
case 0:
        perror("EOF");
default:
        printf("Van ban nhan duoc co %d ky tu: %s\n",fd, texte);
}
}
main()  {
int     fd[2];
char    chaine[10];

if (pipe(fd)==-1)
        { perror("Loi khoi tao pipe.");
        exit(1);
}

switch (fork())  {
case   -1:
        perror(" Loi khoi tao tien trinh.");
        break;
case   0:
        if (close(fd[1])==-1)
        perror(" Error.");
        code_fils(fd[0]);
        exit(0);
}
close(fd[0]);
if (write(fd[1]),"hello",6)==-1)
        perror("Loi truyen.");
}
```

KJt quT chehng trình:

```
So hieu mo ta la: 5
Van ban nhan duoc co 6 ky tu: hello
```

Chú ý rwng, tiJn trình con L c trong ng dXn mà không viJt Ló nên nó bdt LFu bwng cách Lóng phFn viJt fd[1] LW tiJt ki m các tín hi u mô tT c a t h p. Tehng t , vì tiJn trình cha ch s d ng phFn viJt nên nó Lóng phFn L c lQi (fd[0]). Sau Ló tiJn trình cha viJt vào ng dXn 6 ký t và tiJn trình con Lã L c chúng.

Bài 9 / ±p trình m ¥ng TCP/IP

9.1. L ±p trình client /server theo giao th íc TCP/IP

· ***Ch ng trình* tcpClient.c**

```
/* Chuong trinh tcpClient.c */
```

```
/* Khai báo các file thư viện cần thiết để gọi hàm socket*/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>                 /*gethostbyname*/
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>                     /* close */

#define SERVER_PORT 1500
#define MAX_MSG 100

int main (int argc, char *argv[]) {

  /* Khởi tạo các biến dùng trong chương trình */
  int sd, rc, i;
  struct sockaddr_in localAddr, servAddr;
  struct hostent *h;

  if(argc < 3) {
    printf("usage: %s <IPserver> <data1> <data2> ... <dataN>\n",argv[0]);
    exit(1);
  }

  /* Hàm gethostbyname() lấy về địa chỉ IP theo tên nhập vào trong tập tin /etc/hosts */

  h = gethostbyname(argv[1]);
  if(h==NULL) {
    printf("%s: unknown host '%s'\n",argv[0],argv[1]);
    exit(1);
  }

  servAddr.sin_family = h->h_addrtype;
  memcpy((char *) &servAddr.sin_addr.s_addr, h->h_addr_list[0], h->h_length);
  servAddr.sin_port = htons(SERVER_PORT);

  /*    Gán các giá trị cho đối tượng socket.
        Tạo socket cho máy Client.  Lưu lại số mô tả socket */

  sd = socket(AF_INET, SOCK_STREAM, 0);
  if(sd<0) {
    perror("cannot open socket ");
    exit(1);
  }

  /*    Đặt tên socket cho chương trình Client
        Gán địa chỉ kết nối cho socket theo giao thức Internet */

  localAddr.sin_family = AF_INET;
  localAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  localAddr.sin_port = htons(0);

 /* Hàm htons() dùng để chuyển đổi trật tự byte của số nguyên trước khi gởi đi – do hệ
    thống sử dụng cơ chế giao tiếp TCP/IP */

/* Ràng buộc tên với socket */
```

```
  rc = bind(sd, (struct sockaddr *) &localAddr, sizeof(localAddr));
  if(rc<0) {
    printf("%s: cannot bind port TCP %u\n",argv[0],SERVER_PORT);
    perror("error ");
    exit(1);
  }

  /* Thực hiện kết nối đến server theo tên/địa chỉ nhập vào từ dòng lệnh */

  rc = connect(sd, (struct sockaddr *) &servAddr, sizeof(servAddr));
  if(rc<0) {
    perror("cannot connect ");
    exit(1);
  }

  /* Sau khi socket đã kết nối, thực hiện gửi các dữ liệu đến chương trình Server */

  for(i=2;i<argc;i++) {

    rc = send(sd, argv[i], strlen(argv[i]) + 1, 0);

    if(rc<0) {
      perror("cannot send data ");
      close(sd);
      exit(1);

    }/* if */
    printf("%s: data%u sent (%s)\n",argv[0],i-1,argv[i]);

  }/* for */
return 0;
}/*main*/
```

· ***Ch^^ ng trình*** **tcpServer.c**

```
/* Chuong trinh tcpServer.c */
/* Khai báo các file thư viện cần thiết để gọi hàm socket*/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>                /* close */


#define SUCCESS 0
#define ERROR   1

#define END_LINE 0x0
#define SERVER_PORT 1500
#define MAX_MSG 100

/* function readline */
int read_line();
```

```
int main (int argc, char *argv[]) {

  int sd, newSd, cliLen;

  struct sockaddr_in cliAddr, servAddr;
  char line[MAX_MSG];


  /*    Gán các giá trị cho đối tượng socket.
        Tạo socket cho máy Server. Lưu lại số mô tả socket */

  sd = socket(AF_INET, SOCK_STREAM, 0);
   if(sd<0) {
    perror("cannot open socket ");
    return ERROR;
  }

  /*    Đặt tên socket cho chương trình Server
        Gán địa chỉ kết nối cho socket theo giao thức Internet */
  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = htons(SERVER_PORT);

  if(bind(sd, (struct sockaddr *) &servAddr, sizeof(servAddr))<0) {
    perror("cannot bind port ");
    return ERROR;
  }

  /*    Tạo hàng đợi lắng nghe kết nối của client
        Cho phép hàng đợi nhận tối đa 5 kết nối */
  listen(sd,5);

 /* Lặp liên tục chờ và lxy kết nối của client */
  while(1) {

    printf("%s: waiting for data on port TCP %u\n",argv[0],SERVER_PORT);
    cliLen = sizeof(cliAddr);

    /* Chấp nhận kết nối */
    newSd = accept(sd, (struct sockaddr *) &cliAddr, &cliLen);
    if(newSd<0) {
      perror("cannot accept connection ");
      return ERROR;
    }

    /* init line */
    memset(line,0x0,MAX_MSG);

    /* Đọc dữ liệu do Client gởi đến - xử lý dữ liệu nhận được  */
    while(read_line(newSd,line)!=ERROR) {

      printf("%s: received from %s:TCP%d : %s\n", argv[0], inet_ntoa(cliAddr.sin_addr),
                                                  ntohs(cliAddr.sin_port), line);

      /* init line */
      memset(line,0x0,MAX_MSG);
```

```
   } /* while(read_line) */

  } /* while (1) */

}


/* WARNING */
/* this function is experimental. I don't know yet if it works  */
/* correctly or not. Use Steven's readline() function to have  something robust.*/
/* rcv_line is my function readline(). Data is read from the socket when */
/* needed, but not byte after bytes. All the received data is read.      */
/* This means only one call to recv(), instead of one call for  each received byte. */
/* You can set END_CHAR to whatever means endofline for you. (0x0A is \n)*/
/* read_lin returns the number of bytes returned in line_to_return       */

/* Hàm có chức năng đọc dữ liệu từ socket*/
int read_line(int newSd, char *line_to_return) {

  static int rcv_ptr=0;
  static char rcv_msg[MAX_MSG];
  static int n;
  int offset;

  offset=0;

  while(1) {
    if(rcv_ptr==0) {
      /* read data from socket */
      memset(rcv_msg,0x0,MAX_MSG);              /* init buffer */
      n = recv(newSd, rcv_msg, MAX_MSG, 0);     /* wait for data */
      if (n<0) {
         perror(" cannot receive data ");
         return ERROR;
      } else if (n==0) {
         printf(" connection closed by client\n");
         close(newSd);
         return ERROR;
      }
    }

    /* if new data read on socket  OR if another line is still in buffer */
    /* copy line into 'line_to_return' */

    while(*(rcv_msg+rcv_ptr)!=END_LINE && rcv_ptr<n) {
      memcpy(line_to_return+offset,rcv_msg+rcv_ptr,1);
      offset++;
      rcv_ptr++;
    }

    /* end of line + end of buffer => return line */
    if(rcv_ptr==n-1) {
      /* set last byte to END_LINE */
      *(line_to_return+offset)=END_LINE;
      rcv_ptr=0;
      return ++offset;
    }
```

```
    /* end of line but still some data in buffer => return line */
    if(rcv_ptr <n-1) {
      /* set last byte to END_LINE */
      *(line_to_return+offset)=END_LINE;
      rcv_ptr++;
      return ++offset;
    }

    /* end of buffer but line is not ended => */
    /*  wait for more data to arrive on socket */
    if(rcv_ptr == n) {
      rcv_ptr = 0;
    }

  } /* while */
}/*main*/
```

9.2. L ±p trình client /server theo giao th ía UDP/IP

· ***Ch¸ ^ ng trình* udpClient.c**

```
/* udpClient.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>                 /* memset() */
#include <sys/time.h>               /* select() */

#define REMOTE_SERVER_PORT 1500
#define MAX_MSG 100


int main(int argc, char *argv[]) {

  int sd, rc, i;
  struct sockaddr_in cliAddr, remoteServAddr;
  struct hostent *h;

  /* check command line args */
  if(argc<3) {
    printf("usage : %s <server> <data1> ... <dataN> \n", argv[0]);
    exit(1);
  }

  /* get server IP address (no check if input is IP address or DNS name */
  h = gethostbyname(argv[1]);
  if(h==NULL) {
    printf("%s: unknown host '%s' \n", argv[0], argv[1]);
    exit(1);
  }
```

```
  printf("%s: sending data to '%s' (IP : %s) \n", argv[0], h->h_name,
        inet_ntoa(*(struct in_addr *)h->h_addr_list[0]));

  remoteServAddr.sin_family = h->h_addrtype;
  memcpy((char *) &remoteServAddr.sin_addr.s_addr,
        h->h_addr_list[0], h->h_length);
  remoteServAddr.sin_port = htons(REMOTE_SERVER_PORT);

  /* socket creation */
  sd = socket(AF_INET,SOCK_DGRAM,0);
  if(sd<0) {
    printf("%s: cannot open socket \n",argv[0]);
    exit(1);
  }

  /* bind any port */
  cliAddr.sin_family = AF_INET;
  cliAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  cliAddr.sin_port = htons(0);

  rc = bind(sd, (struct sockaddr *) &cliAddr, sizeof(cliAddr));
  if(rc<0) {
    printf("%s: cannot bind port\n", argv[0]);
    exit(1);
  }


  /* send data */
  for(i=2;i<argc;i++) {
    rc = sendto(sd, argv[i], strlen(argv[i])+1, 0,
                (struct sockaddr *) &remoteServAddr,
                sizeof(remoteServAddr));

    if(rc<0) {
      printf("%s: cannot send data %d \n",argv[0],i-1);
      close(sd);
      exit(1);
    }

  }

  return 1;

}
```

· ***Ch*** ***ng trình*** **udpServer.c**

```
/* udpServer.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h> /* close() */
```

```
#include <string.h> /* memset() */

#define LOCAL_SERVER_PORT 1500
#define MAX_MSG 100

int main(int argc, char *argv[]) {

  int sd, rc, n, cliLen;
  struct sockaddr_in cliAddr, servAddr;
  char msg[MAX_MSG];

  /* Tạo socket trên máy Server - Đặt tên cho socket của chương trình Server */

  sd=socket(AF_INET, SOCK_DGRAM, 0);
  if(sd<0) {
    printf("%s: cannot open socket \n",argv[0]);
    exit(1);
  }

  /* bind local server port – ràng buộc tên với socket */

  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = htons(LOCAL_SERVER_PORT);
  rc = bind (sd, (struct sockaddr *) &servAddr,sizeof(servAddr));

  if(rc<0) {
    printf("%s: cannot bind port number %d \n",  argv[0], LOCAL_SERVER_PORT);
    exit(1);
  }

  printf("%s: waiting for data on port UDP %u\n", argv[0],LOCAL_SERVER_PORT);

  /* Thực hiện vòng lặp vô hạn trên Server để chờ và xử lý kết nối đến từ máy client */
  while(1) {

    /* Khởi tạo bộ đệm */
    memset(msg,0x0,MAX_MSG);

    /* Nhận dữ liệu gởi đến từ client */
    cliLen = sizeof(cliAddr);
    n = recvfrom(sd, msg, MAX_MSG, 0, (struct sockaddr *) &cliAddr, &cliLen);

    if(n<0) {
      printf("%s: cannot receive data \n",argv[0]);
      continue;
    }

    /* In dữ liệu nhận được */
    printf("%s: from %s:UDP%u : %s \n",  argv[0],inet_ntoa(cliAddr.sin_addr),
                                          ntohs(cliAddr.sin_port),msg);

  }/*while*/
return 0;
}
```

Bài 10

DỊCH VỤ TRUYỀN FILE FTP

FTP (File Transfer Protocol) là dịch vụ cho phép truyền các tệp tin giữa hai máy tính Client và Server, quản lý các thư mục và truy cập vào thư tín điện tử. FTP không được thiết lập để truy cập vào một máy khác và chạy các chương trình ở máy đó, chỉ dùng cho việc truyền tệp tin.

Để kết nối FTP, gõ lệnh sau : `ftp <IPAddressServer>`

Lệnh dùng FTP	**Mô tả**
ascii	Chuyển sang chế độ truyền ascii
bell	âm thanh của chương trình sau khi truyền mỗi tệp tin
binary	Chuyển sang chế độ truyền nhị phân
cd directory	Chuyển đổi thư mục hiện hành trên server
cdup	Lùi thư mục hiện hành về một cấp trước đó
close	Huỷ kết nối
delete filename	Xoá một tệp tin trên server
dir directory	Hiển thị thư mục *directory* của server
get filename	Truyền tệp tin trên server về máy cục bộ
hash	Hiển thị/làm mất dấu # cho mỗi khối các ký tự đã truyền được
help	Hiển thị các trợ giúp
lcd directory	Chuyển đổi thư mục hiện hành trên máy cục bộ
ls directory	Xem danh sách các tệp tin trong thư mục directory trên Server
mdelete files	Xóa nhiều tệp tin trên máy Server
mdir directories	Liệt kê các tệp tin trong nhiều thư mục trên máy Server
mget files	Tải nhiều tệp tin trên máy Server về thư mục hiện hành của máy cục bộ
mkdir <directory >	Tạo thư mục trên máy Server
mput files	Gởi một số tệp tin từ máy cục bộ lên máy Server
open host	Kết nối với Server host từ xa
put filename	Truyền tệp tin từ máy cục bộ lên máy Server
pwd	Hiển thị thư mục hiện hành trên server

status	HiWn th trUng thái c a ftp
rename file1 file2	n i tên *file1* trên máy Server thành *file2*
quote	Cung cBp m t l nh FTP m t cách tr c tiJp
quit	ChBm d t kJt n i và thoát kh i ftp
?	HiWn th danh sách l nh

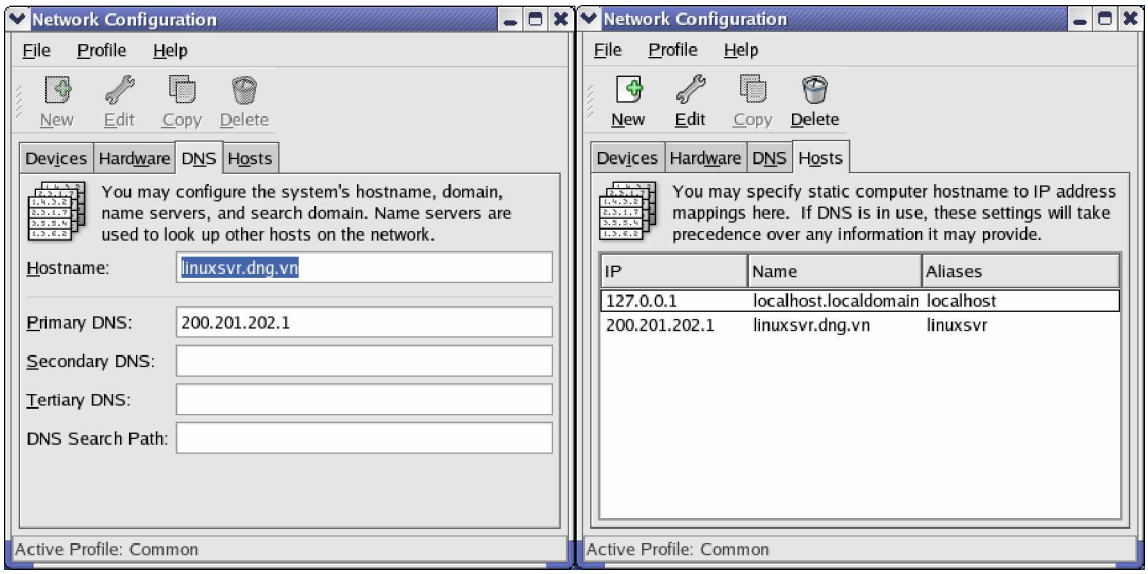
Khi truy cdp vào h th ng, nJu chEa có account, ngE i dùng có thW login v i account L'Hc bi t là anonymous, không có mdt khXu.

Th õc hành

C:\>ftp ¿	Khởi động ftp từ thư mục hiện hành C:\
(to) : 200.201.202.180	
user : user01	Nhập vào tên user
Password :	Nhập vào mật khẩu tương ứng
ftp> dir	Xem nội dung thư mục
ftp> ?	Xem nội dung các lệnh của ftp
ftp>put autoexec.bat autoexec.dos	Chuyển tập tin từ Client lên Server với tên mới là autoexec.dos
ftp> ls	Xem kết quả truyền file
ftp>get autoexec.dos LINUX.TXT	Lấy tập tin autoexec.dos trên Server về Client với tên mới là LINUX.TXT
ftp>mget autoexec.dos	Lấy tập tin autoexec.dos trên Server về Client thư mục C:\
ftp>cd /home/user01	Chuyển đến thư mục hiện hành là user01 là thư mục có toàn quyền của user user01
ftp>mdir document	Tạo trong thư mục user01 thư mục mới có tên document
ftp> help dir	Xem hướng dẫn sử dụng lệnh dir
ftp>help get	Xem hướng dẫn sử dụng lệnh get
ftp> quit	Kết thúc phiên làm việc

7. Xem thông tin vQ cBlu hình thiЛt b mЧng

```
[root@linuxsvr root]#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:06:7B:02:71:21
          inet addr:200.201.202.1  Bcast:200.201.202.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2326 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70927 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:218392 (213.2 Kb)  TX bytes:6939053 (6.6 Mb)
          Interrupt:9 Base address:0x4c00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:933 errors:0 dropped:0 overruns:0 frame:0
          TX packets:933 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:87261 (85.2 Kb)  TX bytes:87261 (85.2 Kb)
```



Hình 1. CBlu hình d ch v mЧng bYng ti n ích redhat-config-network.

Bài 12 **& U HÌNH D ÎCH V è DNS**

12.1. Các t ±p tin c @u hình d Ëch v é DNS

12.1.1. Tɔp tin /etc/host.conf

```
order hosts,bind
```

12.1.2. Tɔp tin /etc/resolv.conf

```
:search dng.vn
nameserver 200.201.202.1
```

12.1.3. Tɔp tin /etc/named.conf

```
# named.conf - configuration for bind
# Generated automatically by redhat-config-bind, alchemist et al.
# Any changes not supported by redhat-config-bind should be put
# in /etc/named.custom
controls {
      inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};
include "/etc/named.custom";
include "/etc/rndc.key";
zone  "0.0.127.in-addr.arpa" {
      type master;
      file  "0.0.127.in-addr.arpa.zone";
};
zone  "localhost" {
      type master;
      file  "localhost.zone";
};
zone  "dng.vn" {
      type master;
      file  "dng.vn.zone";
};
zone  "edu.vn" {
      type master;
      file  "edu.vn.zone";
};
```

12.1.4. Tɔp tin /var/named/dng.vn.zone

```
$TTL 86400
@      IN      SOA     dng.  root.localhost (
                       1 ; serial
                       28800 ; refresh
                       7200 ; retry
                       604800 ; expire
                       86400 ; ttl
                       )
       IN      NS      200.201.202.1.
```

```
www                 IN    A     200.201.202.1
tankhoi01     IN    A     200.201.202.1
tankhoi02     IN    A     200.201.202.2
```

12.1.5. Tɔp tin /var/named/edu.vn.zone

```
$TTL 86400
@     IN    SOA    edu.  root.localhost (
                2 ; serial
                28800 ; refresh
                7200 ; retry
                604800 ; expire
                86400 ; ttl
                )
               IN    NS    200.201.202.1.
www                 IN    A     200.201.202.1
tankhoi01     IN    A     200.201.202.1
tankhoi02     IN    A     200.201.202.2
```

12.1.6. Tɔp tin /var/named/0.0.127.in-addr.arpa.zone

```
$TTL 86400
@     IN    SOA    localhost.     root.linuxsvr.dng.vn (
                36 ; serial
                28800 ; refresh
                7200 ; retry
                604800 ; expire
                86400 ; ttk
                )
@     IN    NS     localhost.
1     IN    PTR    localhost.
1     IN    PTR    www.
1     IN    PTR    tankhoi01.
2     IN    PTR    tankhoi02.
1     IN    PTR    www.
1     IN    PTR    tankhoi01.
2     IN    PTR    tankhoi02.
```

12.1.7. Tɔp tin /var/named/localhost.zone

```
$TTL 86400
@     IN    SOA    @  root.localhost (
                1 ; serial
                28800 ; refresh
                7200 ; retry
                604800 ; expire
                86400 ; ttl
                )
     IN    NS     localhost.
@     IN    A      127.0.0.1
```

12.1.8. L□nh khεi l'íng d□ch vλ DNS

```
/etc/init.d/named restart
```

12.2. Các l Ënh và ti Ën ích h Ût r ç

12.2.1. LMnh nslookup

```
#nslookup
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.  Run nslookup with
the `-sil[ent]' option to prevent this message from appearing.
> www.dng.vn
Server:         200.201.202.1
Address:        200.201.202.1#53

Name:   www.dng.vn
Address: 200.201.202.1
> tankhoi02.edu.vn
Server:         200.201.202.1
Address:        200.201.202.1#53

Name:   tankhoi02.edu.vn
Address: 200.201.202.2
```

12.2.2. LMnh host

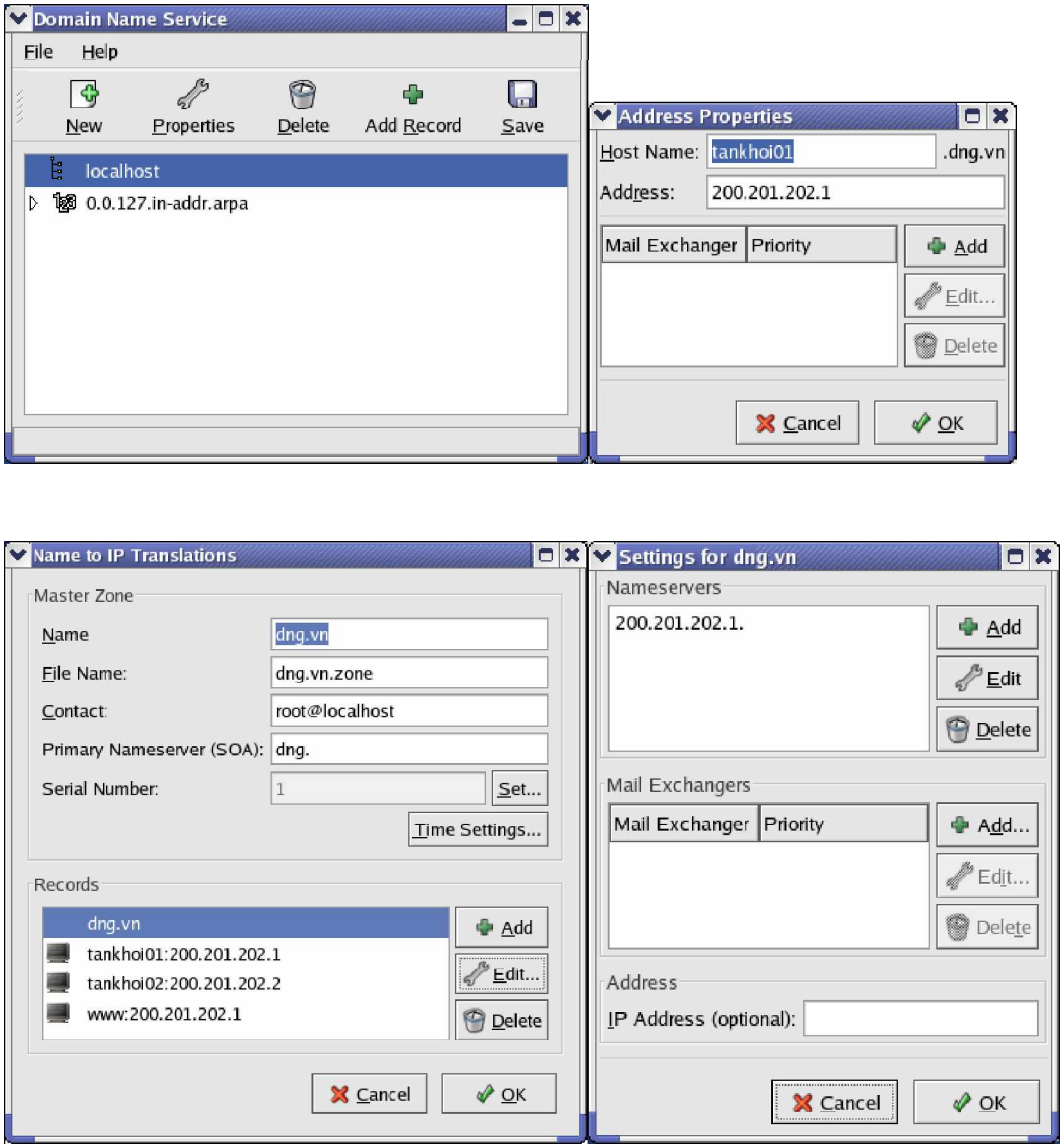
```
#host tankhoi01.dng.vn

tankhoi01.dng.vn has address 200.201.202.1
```

12.2.3. LMnh dig

```
# dig dng.vn
; <<>> DiG 9.2.1 <<>> dng.vn
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58922
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;dng.vn.                        IN      A

;; AUTHORITY SECTION:
dng.vn.             86400   IN      SOA     dng. root.localhost.dng.vn. 1 28800 7200
604800 86400

;; Query time: 28 msec
;; SERVER: 200.201.202.1#53(200.201.202.1)
;; WHEN: Mon Mar 22 09:14:13 2004
;; MSG SIZE  rcvd: 78
```

12.2.4. Ti Ën ích redhat-config-bind

```
#redhat-config-bind
```



Hình 2. Cấu hình dịch vụ BIND bằng tiện ích redhat-config-bind.

Mở đầu : Giới thiệu tài liệu

Có lẽ bạn đọc đã từng nghe đến tên hệ điều hành nguồn mở Linux thông qua các phương tiện truyền thông đại chúng. Vào tháng 3-2004 tại Hà Nội, Bộ Khoa học và Công nghệ sẽ cùng CICC (Trung tâm Hợp tác Quốc tế về Tin học hoá, Nhật Bản) tổ chức Diễn đàn Châu Á lần thứ 3 về phần mềm nguồn mở (OSS). Năm 2000 và 2002 hội nghị toàn quốc về Linux cũng đã họp tại Hà Nội. Năm 2003, một số quốc gia như Trung Quốc, Nhật Bản và Hàn Quốc đã quyết định đưa việc bản địa hoá Linux vào kế hoạch phát triển công nghệ phần mềm của mình. Còn ở Việt Nam cũng đã có khoảng hai dự án với ý định như thế.

Chưa tròn 10 tuổi, Linux đã mở rộng ra ngoài phạm vi nghiên cứu đại học để phục vụ cho mục đích thương mại và hành chính, hoặc dùng làm hệ điều hành cho các mạng máy tính. Quả thật Linux đã tiến triển và hoàn thiện liên tục với những phiên bản mới, thậm chí năm 2003 các dòng Linux ManDrake và RedHat v.v. đều đã có đến bản 9.0. Mặt khác Linux càng ngày càng có thêm nhiều người sử dụng và vì vậy mà tài liệu trong tay bạn đã ra đời.

Tài liệu này dành cho ai?

Ngoài chương 1 dành cho các nhà quản lý dự án công nghệ thông tin (CNTT), bất cứ ai quan tâm sâu hơn đến Linux đều có thể sử dụng tài liệu này để tìm hiểu việc cài đặt, thiết lập cấu hình và sử dụng nó, đặc biệt trong môi trường mạng.

Các chương trong tài liệu này chủ yếu sẽ hướng dẫn cách cài đặt và sử dụng dòng sản phẩm RedHat vì có lẽ đó là dòng Linux phổ biến nhất và cũng dễ cài đặt nhất từ trước đến nay. Riêng chương 4 dành cho dòng sản phẩm Caldera. Ngoài ra, tài liệu còn cung cấp những hiểu biết khác, thí dụ cập nhật và nâng cấp các phần mềm tương hợp với Linux, hoặc in ấn, hỗ trợ an ninh và quản trị hệ thống một cách thuận tiện.

Đối với những ai đã biết hệ điều hành UNIX thì việc nắm bắt Linux sẽ rất đơn giản.

Ngược lại, mặc dù Linux không phải là UNIX nhưng nhiều thao tác và tiến trình cần thiết để chạy Linux cũng giống như trong UNIX. Do đó khi biết sử dụng Linux thì bạn cũng có thể nắm được các hệ điều hành kiểu UNIX.

Tài liệu này cũng phù hợp cho những người muốn biết thêm về Linux và UNIX mà chưa có dịp sử dụng hai hệ điều hành ấy. Thậm chí, tài liệu sẽ có ích với những người tuy biết cách cài đặt Linux và sử dụng UNIX, nhưng chưa có dịp thực hiện các công việc quản trị hệ thống bao giờ. Tài liệu sẽ giải thích chi tiết về cách quản trị và duy trì hệ thống Linux/UNIX. Một người sử dụng UNIX bình thường khó có quyền làm quản trị hệ thống, song với Linux thì có thể trở thành chủ nhân của toàn bộ hệ thống.

Linux dẫn xuất từ UNIX nên cũng là một hệ điều hành đa người dùng và đa nhiệm (phục vụ nhiều người và thực hiện nhiều việc cùng lúc). Nó có thể chạy trên nhiều bộ vi xử lý (đặc biệt trên họ Intel từ đời 386 trở lại đây) và tương thích với chuẩn mở POSIX. POSIX là một tiêu chuẩn quốc tế cho các hệ điều hành và phần mềm khả chuyển với những thành phần có thể sử dụng chung, đảm bảo tính mở của chúng.

Những phần cứng tương thích

Phần lớn Linux được soạn thảo qua Internet bởi những đồng đội của Torvald, do đó những phần cứng của họ là tương thích nhất với hệ điều hành này. May sao, nhiều hãng sản xuất phần cứng đã coi Linux như một thị trường có tương lai, họ cung cấp các thông tin chi tiết về sản phẩm của mình để tạo điều kiện cho các nhà phát triển phần mềm viết trình điều khiển (driver) giúp cho phần cứng này chạy được trên Linux. Thậm chí có các công ty đặt hàng cho những nhà phát triển viết các driver tương thích Linux, sau đó phổ biến cho mọi người biết và cùng hưởng quyền lợi theo giấy phép GPL, trong khi một số công ty khác còn giấu thông tin vì lý do bản quyền và cạnh tranh.

Có thể tham khảo trên Internet các trang Web về "Linux Hardware Compatibility" để biết chính xác những phần cứng nào tương thích với Linux. Các máy tính Unix thường là chuyên dụng và chỉ làm việc được với rất ít phần cứng do chính các hãng sản xuất quy định (Sun, IBM, HP, Intergraph v.v.).

Trong trường hợp phổ biến nhất, ta có thể chạy Linux một cách dễ dàng trên rất nhiều máy tính tương hợp IBM-PC mà không cần dùng đến hệ điều hành DOS hoặc Windows; hơn nữa, Linux còn thay thế được hai chương trình ấy. Nên phân vùng lại đĩa cứng để có chỗ cài đặt một bản Linux riêng, mặc dù cũng có phiên bản chạy ngay trên DOS hoặc Windows. Linux là một chương trình luôn tiến triển, cho nên khi cập nhật phiên bản mới bạn phải biết rằng mình có thể đánh mất những dữ liệu đang có trong máy. Đừng cài đặt Linux nếu chưa sao lưu dữ liệu.

Quy ước cách đọc tài liệu

Linux và UNIX phân biệt chữ hoa và chữ thường. Khi tài liệu này yêu cầu bạn nhập (gõ) các chữ tại dòng lệnh từ sau dấu nhắc shell thì bạn phải nhập chính xác những gì được ghi trong tài liệu, tức là chữ hoa nhập khác chữ thường.

Kiểu chữ `Courier` thường được sử dụng cho câu lệnh để phân biệt với câu mô tả. Còn trong câu mô tả nếu có chỗ nào cần nhấn mạnh thì chỗ ấy sẽ được **in đậm.**

Đôi khi tài liệu yêu cầu bạn phải bấm vài phím. Thí dụ các phím <Ctrl-h> được thực hiện bằng cách bấm và giữ phím <Ctrl>, rồi bấm phím <h>, sau đó nhả cả hai phím ra.

Ghi chú: Nhằm tránh tình trạng nhầm lẫn chữ in hoa hoặc in thường ở môi trường Linux và UNIX, tài liệu này sử dụng chữ in thường đối với phím cần phải bấm. Thí dụ, trong tài liệu sẽ in là <Ctrl-c> thay vì <Ctrl-C> (nếu in theo cách thứ hai, bạn đọc sẽ phân vân không hiểu liệu mình có nên bấm nút <Ctrl> rồi sau đó phải bấm cả <Shift> và <c> để có chữ C in hoa hay không.

Các thí dụ cũng thường được in bằng kiểu chữ `Courier`. Thí dụ dưới đây sẽ minh hoạ phản ứng (hồi đáp) của Linux trên màn hình hiển thị sau khi bạn nhập vào một câu lệnh. Những thí dụ ấy thường mở đầu bằng dấu nhắc shell được hiển thị như dấu dollar ($). Sau dấu đó là câu lệnh mà bạn cần nhập vào, vì vậy bạn đừng nhập dấu nhắc nữa.

```
$ lp report.txt &
3146
$
```

Trong thí dụ trên bạn chỉ cần nhập những gì sau dấu $ ở dòng thứ nhất (nghĩa là chỉ nhập chuỗi ký tự **lp report.txt &** và bấm phím <Enter>). Những gì được hiển thị ở các dòng sau chính là hồi đáp của Linux đối với câu lệnh vừa nhập vào.

Khi trình bày cú pháp của một lệnh Linux, tài liệu này thường sử dụng kiểu chữ *in nghiêng* để phân biệt giữa phần phải có và phần biến thiên. Hãy xem thí dụ sau:

```
lp filename
```

Ở đây phần *filename* của câu lệnh là một biến, nghĩa là nó sẽ thay đổi tuỳ theo bạn muốn lệnh **lp** làm việc với cái gì. Phần **lp** là phải có bởi vì đó là tên câu lệnh. Thông tin về biến được in dưới dạng nghiêng nhưng phải nhớ rằng khi nhập vào thì không nên nhập ở dạng chữ nghiêng.

Trong vài trường hợp, thông tin câu lệnh lại mang tính tuỳ chọn, có hay không có cũng được, nghĩa là câu lệnh vẫn chạy mà không cần đến thông tin ấy. Phần tuỳ chọn được quy ước đóng trong các ngoặc vuông ([…]). Xem thí dụ sau:

```
lp filename [device1] [abc]
```

Ở đây, **lp** là tên câu lệnh, không phải tuỳ chọn hoặc biến. Thông số *device1* vừa là biến thiên vừa là tuỳ chọn (được in chữ nghiêng và nằm trong ngoặc vuông). Nói rõ hơn, bạn có thể nhập bất kỳ tên thiết bị nào vào vị trí của *device1* (nhưng không nhập ngoặc vuông), hoặc bạn không nhất thiết phải nhập thông số nào vào cả. Thông số *abc* là tuỳ chọn (nếu bạn không thích sử dụng thì đừng nhập vào), song đó lại không phải là biến. Nếu sử dụng nó, bạn phải nhập vào đúng y như là in trên tài liệu (và, nhắc lại lần nữa, không nên nhập ngoặc vuông vào).

Các từ ***Ghi chú, Cẩn thận*** và ***Lưu ý*** trong tài liệu này sẽ được in bằng chữ ***nghiêng đậm*** để dễ nhìn thấy.

Các phần bổ sung dài hơn và tách ra khỏi ý chính trong đoạn văn sẽ được đặt trong một khung cửa sổ mang tiêu đề riêng.

Tài liệu cũng dùng các chỉ dẫn trỏ đến chương, mục, bảng, danh sách tham khảo ở trong hoặc ở ngoài giáo trình này. Mỗi tham khảo chéo sẽ có dạng in *nghiêng giữa dòng* như sau:

Xem "Sử dụng X Window"

Những thông tin ít quan trọng hơn có thể được in bằng chữ nhỏ để tiết kiệm giấy

Tóm tắt nội dung tài liệu

Tài liệu này ngoài phần mở đầu còn có 4 phần, mỗi phần lại bao gồm nhiều chương.

Phần I: Cài đặt Linux

Phần này gồm có 6 chương, cung cấp một cái nhìn chung về hệ điều hành Linux cũng như các chỉ dẫn để nhanh chóng cài đặt được Linux.

Chương 1, "Tổng quan về Linux", giới thiệu hệ điều hành Linux và tổng quan về các yếu tố hợp thành hệ Linux.

Chương 2, "Chuẩn bị cài đặt Linux", ngoài cách cài đặt tổng quát còn bàn thêm về các loại phần cứng thích hợp, cùng với những vấn đề tiềm tàng và giải pháp.

Chương 3, "Cài đặt RedHat Linux", hướng dẫn cách cài đặt bản phát hành RedHat.

Chương 4, "Cài đặt Caldera OpenLinux", hướng dẫn cách cài đặt bản phát hành Caldera.

Chương 5, "Bắt đầu sử dụng Linux", giới thiệu sử dụng một số ứng dụng cơ bản trên hệ Linux sau khi cài đặt xong.

Chương 6, "Nâng cấp và cài đặt phần mềm với RPM", cung cấp thông tin cần thiết để cài đặt những phần mềm mới bằng cách sử dụng RedHat Package Management (RPM). Ngoài ra chương 6 còn giúp tìm hiểu việc cài đặt phần mềm từ Internet và cập nhật các chương trình hiện có.

Phần II: Quản trị hệ thống

Phần này gồm có 7 chương, cung cấp các thông tin dùng để thiết lập cấu hình và quản lý một hệ Linux tiêu biểu.

Chương 7, "Quản trị hệ thống Linux", cung cấp kiến thức ngắn gọn về các tiến trình cần thiết để cấu hình và duy trì một hệ điều hành Linux.

Chương 8, "Trình soạn thảo văn bản vi", giúp bạn biết sử dụng công cụ soạn thảo văn bản phổ biến nhất trong các hệ Linux/UNIX.

Chương 9. "Khởi động và đóng tắt", giới thiệu những gì diễn ra khi khởi động hoặc đóng tắt hệ điều hành Linux, sau đó sẽ giải thích tại sao không thể tắt máy bằng cách ngắt nguồn điện. Chương này cũng mô tả những tệp mà Linux sẽ sử dụng để khởi động máy.

Chương 10, "Quản lý trương khoản", cho biết cách thêm vào, gỡ bỏ và quản lý trương khoản của người sử dụng trên hệ thống Linux.

Chương 11, "Sao lưu dữ liệu", giải thích sự cần thiết phải sao lưu dữ liệu và cách thức để cất giữ một hệ thống chạy Linux.

Chương 12, "An ninh hệ thống", cung cấp một cái nhìn tổng thể về an ninh dữ liệu trên hệ Linux, sau đó giải thích những thao tác cần thiết nhằm duy trì hệ thống an toàn một cách hợp lý.

Chương 13, "Thiết lập cấu hình lõi Linux", minh hoạ cách thức thiết lập cấu hình của một lõi (kernel) Linux trên phần cứng, áp dụng cho bất kỳ bản phát hành nào.

Phần III: Hệ thống tệp và thư mục

Phần III gồm có 4 chương, giúp bạn tận dụng khai thác các lợi điểm của Linux. Tất cả kiến thức có được từ bốn chương sau đây cũng sẽ dễ dàng áp dụng cho các hệ điều hành thuộc họ UNIX:

Chương 14, "Quản lý hệ thống tệp", cho biết cách tạo ra các tệp, thao tác và sử dụng hệ thống tệp trên Linux.

Chương 15, "Sử dụng Samba", trình bày cách cấu hình cho Linux sử dụng được Samba với những hệ thống tệp khác.

Chương 16, "Hệ thống tệp Linux", giải thích về các quyền hạn của tệp đối với người sử dụng hệ thống Linux và các loại tệp.

Chương 17, "Quản lý tệp và thư mục", giới thiệu chi tiết về tổ chức và cơ cấu của hệ thống tệp trên Linux, quy ước cách đặt tên tệp, cùng với các thứ bậc của thư mục.

Phần IV: Làm việc với Linux

Phần IV gồm có 4 chương, bổ sung kỹ năng thao tác trên các công cụ dòng lệnh và những tiện ích khác.

Chương 18, "Các shell của Linux", giới thiệu về shell script và các dạng shell thường thấy ở những phiên bản khác nhau của Linux.

Chương 19, "Quản lý đa tiến trình", khám phá các khả năng của Linux khi chạy cùng lúc nhiều tiến trình khác nhau. Ta sẽ học cách khởi động đa tiến trình, cách quản lý, kiểm soát và ngừng tiến trình.

Chương 20, "In ấn", lược giải quá trình in, từ các hiểu biết cơ bản về in ấn cho đến việc ra lệnh in, kiểm tra hiện trạng máy in để ngưng tiến trình in, và giải quyết những vấn đề phổ quát liên quan đến in ấn.

Chương 21, "Sử dụng X Window", cung cấp thông tin cần thiết để chạy X Window trên Linux.

Chương 1. Tổng quan về Linux

Chương này chủ yếu dành cho các nhà quản lý dự án công nghệ thông tin. Tuy không thật cần thiết cho việc cài đặt và sử dụng Linux, nhưng nội dung của nó cũng có thể bổ ích cho bất kỳ ai muốn tìm hiểu về những chủ đề sau đây:

-Linux là gì?
-Tại sao Linux phát triển?
-Các bản phát hành Linux
-Lợi ích của Linux
-Ai phát triển Linux?
-Linux cộng sinh với Windows
-Thương mại hóa Linux
-UNIX và Linux
-Tác quyền và bản quyền Linux

1.1 Linux là gì?

Linux xuất hiện như một sản phẩm nguồn mở miễn phí và đến nay đã có thể sánh vai với các hệ điều hành thương phẩm như MS Windows, Sun Solaris v.v. Linux ra đời từ một dự án hồi đầu những năm 1990 có mục đích tạo ra một hệ điều hành kiểu UNIX cài đặt trên máy tính cá nhân chạy với bộ vi xử lý Intel, tương hợp họ máy tính IBM-PC (còn gọi tắt là PC). Từ lâu, UNIX đã nổi tiếng là một hệ điều hành mạnh, tin cậy và linh hoạt, nhưng vì khá đắt nên chủ yếu chỉ dùng cho các trạm tính toán hoặc máy chủ cao cấp.

Ngày nay Linux có thể cài đặt trên nhiều họ máy tính khác nhau, không chỉ riêng cho họ PC. Qua Internet, Linux được hàng nghìn nhà lập trình khắp trên thế giới tham gia thiết kế, xây dựng và phát triển, với mục tiêu không lệ thuộc vào bất kỳ thương phẩm nào và để cho mọi người đều có thể sử dụng thoải mái. Khởi thuỷ, Linux xuất phát từ ý tưởng của Linus Torvalds, khi đó chàng sinh viên Đại học Helsinki ở Phần Lan đã muốn thay thế Minix, một hệ điều hành nhỏ kiểu UNIX.

Về cơ bản, Linux bắt chước UNIX cho nên cũng có nhiều ưu điểm của UNIX. Tính đa nhiệm thực sự của Linux cho phép chạy nhiều chương trình cùng lúc. Với Linux, bạn có thể đồng thời thực hiện một số thao tác, thí dụ chuyển tệp, in ấn, sao tệp, nghe nhạc, chơi game v.v.

Linux là hệ điều hành đa người dùng, nghĩa là nhiều người có thể đăng nhập và cùng lúc sử dụng một hệ thống. Ưu điểm này có vẻ không phát huy mấy trên máy PC ở nhà, song ở trong công ty hoặc trường học thì nó giúp cho việc dùng chung tài nguyên, từ đó giảm thiểu chi phí đầu tư vào máy móc.

Ngay cả khi ở nhà, bạn cũng có thể đăng nhập vào Linux với nhiều trương khoản (account) khác nhau qua các terminal ảo và tổ chức dịch vụ trên mạng riêng cho mình bằng cách sử dụng Linux với nhiều modem (xem chương 10).

Có thể kể tên các hệ điều hành miễn phí khác như FreeBSD, OpenBSD, NetBSD v.v. Cũng phải kể đến ảnh hưởng lớn công ty Sun (chủ nhân của ngôn ngữ Java) vì Sun muốn cung cấp hệ điều hành Solaris dùng miễn phí trên máy PC. Phiên bản Solaris chạy trên chip Intel sẽ trở nên một đối thủ đáng gờm của Linux với mã nguồn mở và nhờ danh tiếng là hệ điều hành rất ổn định và tương thích với hệ Solaris chạy trên chip Sun SPARC.

Bản thân việc độc lập với những công ty lớn cũng tiềm tàng một điểm yếu của Linux. Khi chưa có một mạng lưới riêng cung cấp dịch vụ bảo trì thì tất nhiên người ta sẽ ngại sử dụng Linux. Tuy thế, với sự phát triển của Internet, các tổ chức hỗ trợ người dùng Linux đã tạo nên các Website và forum để tháo gỡ cho bạn nhiều vấn đề khó khăn.

Hơn nữa Linux có thể không chạy tốt với một số phần cứng ít phổ biến, thậm chí việc hỏng hóc hoặc xoá mất dữ liệu đôi khi cũng xảy ra, bởi vì Linux luôn thay đổi và khó được thử nghiệm đầy đủ trước khi đưa lên Internet.

Linux không phải là đồ chơi sẵn có, nó được thiết kế nhằm mang đến cho người sử dụng cảm giác cùng tham gia vào một dự án mới. Tuy nhiên thực tế cho thấy Linux chạy tương đối ổn định và cho bạn một cơ may không tốn kém để học và sử dụng UNIX, một họ hệ điều hành chuyên nghiệp hiện nay đang được rất nhiều người dùng trên các máy chủ và trạm tính toán cao cấp.

1.2 Tại sao Linux phát triển?

Trước hết, Linux phát triển vì là một trong những hệ điều hành miễn phí và có khả năng đa nhiệm cho nhiều người sử dụng cùng lúc trên các máy tính tương thích với PC. So với những hệ điều hành thương phẩm, Linux giúp bạn ít phải nâng cấp và lại không cần trả tiền, cũng như phần lớn các phần mềm ứng dụng cho nó. Hơn nữa, Linux và những ứng dụng đó được cung cấp với cả mã nguồn miễn phí mà bạn có thể lấy về từ Internet, sau đó chỉnh sửa và mở rộng chức năng của chúng theo nhu cầu riêng.

Linux có khả năng thay thế một số hệ điều hành thuộc họ UNIX đắt tiền. Nếu tại nơi làm việc mà bạn sử dụng UNIX thì ở nhà bạn cũng thích sử dụng một hệ nào đó giống như thế nhưng rẻ tiền. Linux giúp bạn dễ dàng truy cập, lướt qua các Website và gửi nhận thông tin trên mạng Internet. Nếu bạn là một quản trị viên UNIX thì về nhà bạn cũng có thể sử dụng Linux để thực hiện mọi công việc quản trị hệ thống.

Một nguyên nhân khác làm cho Linux dễ đến với người dùng là nó cung cấp mã nguồn mở cho mọi người.

Chính điều này đã khiến một số tổ chức, cá nhân hay quốc gia đầu tư vào Linux nhằm mở rộng sự lựa chọn ra ngoài các phần mềm đóng kín mã nguồn. Họ cho rằng, mặc dù có dịch vụ hậu mãi nhưng không gì đảm bảo được rằng khi dùng các sản phẩm đóng kín này trên Internet, các thông tin cá nhân hay quốc gia của họ có bị gửi về một tổ chức hay một quốc gia nào khác hay không. Thí dụ Trung Quốc đã phát triển hệ điều hành Hồng Kỳ từ kernel của Linux để không bị lệ thuộc Microsoft Windows, cũng như họ đang tự nghiên cứu bộ vi xử lý Hồng Tâm để thay thế cho họ chip Intel.

Tại Việt Nam, việc nghiên cứu xây dựng một hệ điều hành từ kernel Linux đã thu được một số thành công nhất định. Chắc bạn cũng đã biết đến Vietkey Linux và CMC RedHat Linux (phiên bản tiếng Việt của RedHat Linux 6.2).

Gần đây, các công ty nổi tiếng như IBM, Sun, Intel, Oracle cũng bắt đầu nghiên cứu Linux và xây dựng các phần mềm ứng dụng cho nó.

1.3 Các bản phát hành Linux

Nhiều người đã biết đến các nhà sản xuất phần mềm RedHat, ManDrake, SuSE, Corel và Caldera. Có thể chính bạn cũng đã từng nghe đến tên các phiên bản Linux như Slackware, Debian, TurboLinux và VA Linux, v.v. Quả thật, Linux được phát hành bởi nhiều nhà sản xuất khác nhau, mỗi bản phát hành là một bộ chương trình chạy trên nhóm tệp lõi (kernel) của Linus Tordvalds. Mỗi bản như vậy đều dựa trên một kernel nào đó, thí dụ bản RedHat Linux 6.2 sử dụng phiên bản kernel 2.2.4.

Hãng RedHat đã làm ra chương trình quản lý đóng gói RPM (RedHat Package Manager), một công cụ miễn phí giúp cho bất cứ ai cũng có thể tự đóng gói và phát hành một phiên bản Linux của chính mình. Thí dụ bản OpenLinux của Caldera cũng đã được tạo ra như thế.

1.4 Lợi thế của Linux

Tại sao có thể chọn Linux thay vì chọn một trong những hệ điều hành khác chạy trên PC như DOS, Windows 95/98, Windows NT, hoặc Windows 2000 ?

Linux cung cấp cho bạn một môi trường học lập trình mà hiện nay chưa có hệ nào sánh được. Với Linux, bạn có đầy đủ cả mã nguồn, trong khi đó các sản phẩm mang tính thương mại thường không bao giờ tiết lộ mã nguồn.

Cuối cùng, Linux mang đến cho bạn cơ hội sống lại bầu không khí của cuộc cách mạng vi tính trước kia. Cho đến giữa thập niên 1970, máy tính điện tử còn là sân chơi riêng của các tổ chức lớn, chẳng hạn như chính quyền, tập đoàn doanh nghiệp và trường đại học. Người dân thường đã không thể sử dụng những thành tựu kỳ diệu của công nghệ thông tin.

Song với sự xuất hiện của bộ vi xử lý đầu tiên (1971) rồi máy tính cá nhân (1975), mọi việc đã thay đổi. Thoạt tiên, đó là đất dụng võ của các tay hacker say mê vi tính. Họ thậm chí có thể tự làm ra những máy tính cá nhân và hệ điều hành đơn giản, nhưng các hệ này chưa làm gì được nhiều ở góc độ hiệu năng. Với kinh nghiệm tích luỹ dần theo năm tháng, một số hacker đã trở thành nhà doanh nghiệp, rồi cùng với khả năng tích hợp ngày càng cao của các vi mạch, PC đã trở thành phổ biến (rất tiếc hiện nay xã hội thường nghĩ xấu về chữ "hacker", xin mời bạn xem thêm mục "Ai phát triển Linux?" ở cuối chương này để phân biệt rõ hơn hacker và cracker là những ai).

Ngày nay Linux đang làm một cuộc cách mạng ở lĩnh vực phần mềm hệ thống. Linux là lá cờ tập hợp những con người không muốn bị kiểm sát bởi các hãng khổng lồ nhân danh kinh tế thị trường để làm xơ cứng óc sáng tạo và cải tiến.

Với Linux bạn sẽ khai thác được nhiều thế mạnh của UNIX. Trong số những hệ điều hành thông dụng hiện nay, Linux là hệ điều hành miễn phí được nhiều người sử dụng

rộng rãi nhất. Bản thân Linux đã hỗ trợ sẵn sàng bộ giao thức mạng TCP/IP, giúp bạn dễ dàng kết nối Internet và gửi thư điện tử. Linux thường đi kèm XFree86 là một giao diện đồ hoạ cho người sử dụng (GUI) và cũng được phát hành miễn phí. XFree86 cung cấp cho bạn các chức năng phổ biến ở một số thương phẩm khác, chẳng hạn như Windows.

Tính khả chuyển của một hệ điều hành giúp bạn chuyển nó từ một nền này sang nền khác mà vẫn hoạt động tốt. Thí dụ UNIX là một hệ có tính khả chuyển cao. Ban đầu UNIX chỉ hoạt động trên một nền duy nhất, đó là máy tính mini DEC PDP-7.

Hiện nay UNIX và Linux có khả năng chạy trên bất kỳ nền nào, từ máy xách tay cho đến máy tính lớn. Nhờ tính khả chuyển, các máy tính chạy UNIX và Linux trên nhiều nền khác nhau có thể liên lạc với nhau một cách chính xác và hữu hiệu. Những hệ này có thể hoạt động mà không cần phải bổ sung thêm bất kỳ giao diện liên lạc đắt tiền nào, mà thông thường bạn phải mua thêm sau khi mua những hệ điều hành khác.

Linux đã có hàng ngàn ứng dụng, từ các chương trình bảng tính điện tử, quản trị cơ sở dữ liệu, xử lý văn bản đến các chương trình phát triển phần mềm cho nhiều ngôn ngữ, chưa kể nhiều phần mềm viễn thông trọn gói. Ngoài ra Linux cũng có hàng loạt trò chơi giải trí trên nền ký tự hoặc đồ hoạ. Phần lớn những chương trình tiện ích và ứng dụng có sẵn cho Linux lại không mất tiền mua. Các bạn chỉ phải trả chi phí cho việc tải chúng từ Internet xuống hoặc trả cước phí bưu điện.

Đến với Linux, giới lập trình sẽ có một loạt các công cụ phát triển chương trình, bao gồm các bộ biên dịch cho nhiều ngôn ngữ lập trình hàng đầu hiện nay, chẳng hạn như C, C++. Bạn cũng có thể dùng ngôn ngữ Pascal thông qua trình biên dịch FreePascal. Nếu bạn không thích sử dụng những ngôn ngữ vừa kể, Linux có sẵn các công cụ như Flex và Bison để bạn xây dựng ngôn ngữ riêng cho mình.

Hai khái niệm hiện nay được đề cập rất nhiều là hệ thống mở (open system) và tính liên tác (interoperability) đều gắn với khả năng của những hệ điều hành có thể liên lạc với nhau. Phần lớn các hệ mở đòi hỏi phải thoả mãn tương thích tiêu chuẩn IEEE POSIX (giao diện hệ điều hành khả chuyển). Linux đáp ứng những tiêu chuẩn ấy và được lưu hành với mã nguồn mở.

1.5 Ai phát triển Linux?

Nói chung, Linux là một hệ thống được xây dựng bởi các hacker và cho các hacker. Mặc dù hiện nay trong xã hội từ hacker thường có hàm ý tiêu cực, song nếu theo nghĩa ban đầu thì hacker không phải là tội phạm. Hacker tìm hiểu những gì có bên trong một hệ thống cho đến từng chi tiết và có khả năng sửa chữa nếu hệ thống ấy bị hỏng hóc. Đa số các hacker không xâm nhập hệ thống vì tiền bạc hoặc ác ý, mặc dù sau này đã có những người vượt qua giới hạn ấy và bị tập thể các hacker gọi là cracker (tin tặc) hay hacker mũ đen. Giới hacker cảm thấy bị xúc phạm khi mọi người xem họ như lũ phá hoại và gọi chung là tin tặc.

Thực ra, những hacker chân chính, còn gọi là hacker mũ trắng, rất có công trong việc phát hiện kẽ hở của các phần mềm, giúp mọi người và chủ nhân của những phần mềm ấy cảnh giác trước sự tấn công của giới tin tặc. Cũng nhờ công cuộc bảo vệ này mà

Linux và các ứng dụng Linux (nói rộng hơn là các phần mềm nguồn mở) càng ngày càng an toàn hơn

Ngoài đời, phần lớn những người sử dụng UNIX chỉ được cấp cho một số trương khoản với quyền hạn thu hẹp, do đó một người bình thường không thể thử nghiệm đầy đủ các câu lệnh UNIX. Với Linux bạn có một phiên bản hoạt động tương tự UNIX nhưng cho phép quản trị, sử dụng, vào ra thoải mái không giới hạn, một điều hiếm gặp trong cuộc sống. Linux cho bạn biết thế nào là làm hacker, song chúng tôi hy vọng từ đó bạn sẽ không trở thành cracker.

1.6 Linux cộng sinh với Windows

Về nguyên tắc, tất cả các phần mềm đang chạy trên DOS hoặc Windows sẽ không chạy trực tiếp với Linux, nhưng 3 hệ điều hành này có thể cộng sinh trên cùng một máy PC, dĩ nhiên mỗi lúc chỉ chạy được một hệ điều hành thôi. Bạn cũng có thể cài thêm một chương trình đặc biệt tên là "VMWARE" để phỏng tạo một hay nhiều hệ điều hành khác nhau chạy đồng thời trên cùng một máy với điều kiện máy của bạn phải có một cấu hình thích hợp và đủ mạnh.

Người ta còn xây dựng những chương trình phỏng tạo môi trường Windows và DOS trên nền Linux. Công ty Caldera đã chuyển WABI (Windows Applications Binary Interface) của Sun sang Linux., cho phép các ứng dụng Windows 3.1 chạy với Linux. Caldera bán sản phẩm vừa kể trên và nhiều ứng dụng Linux song vẫn biếu không phiên bản RedHat để chạy các ứng dụng do hãng bán ra. Caldera còn thử chuyển một phiên bản DR DOS sang Linux.

Chương trình WINE cũng được sử dụng như một môi trường phỏng tạo Windows để có thể chạy các ứng dụng Windows trong Linux. Nói chung, Linux có khả năng chạy các ứng dụng Macintosh, DOS và Windows.

Ngược lại, cũng có nhiều người đang soạn thảo những chương trình phỏng tạo Linux trên nền Windows như đã từng có chương trình cho phép chạy các phần mềm Macintosh trên nền Sun và Windows. Bạn có thể xem các thông tin liên quan mới nhất trên các Web site về Linux.

Muốn cài đặt Linux bạn phải phân vùng lại ổ cứng máy mình, mặc dù không phải lúc nào cũng nhất thiết làm như thế. Bạn phải xoá một phần ổ cứng chứa chương trình và dữ liệu có sẵn trong đó. Hiện nay, việc cài đặt Linux mà không phân vùng lại ổ cứng đã được giải quyết nhưng khi chạy vẫn còn chậm. Do đó khi dự định cài đặt Linux bạn nên sao lưu ổ cứng ra vài ba bản.

Ổ cứng cũng cần phải còn đủ chỗ cho cả Linux và những hệ điều hành khác, bạn phải quyết định cái nào giữ lại và cái nào bỏ đi. Bạn có nhiều lựa chọn để phân vùng lại ổ cứng. Chẳng hạn bạn có thể dành chỗ riêng cho DOS và Linux, hoặc bạn chạy một chương trình phân vùng ổ cứng mà không phải xoá các tệp có sẵn. Tuy nhiên rủi ro mất dữ kiện khi cài đặt vẫn còn đó.

Khi phân vùng lại ổ cứng, bạn sẽ kiểm sát vùng đĩa dành riêng cho Linux hữu hiệu hơn, và Linux cũng chạy tốt hơn. Dung lượng đĩa cứng dành cho Linux sẽ tuỳ vào việc bạn muốn cài bao nhiêu ứng dụng và đó là phiên bản Linux nào. Bạn cần có ít nhất 300 MB trống trên ổ đĩa cứng nếu muốn cài RedHat 7.2, chưa kể đến tất cả các

chương trình và dữ liệu mà bạn muốn giữ lại từ hệ điều hành trước đó. Nếu ổ cứng của bạn còn nhiều hơn thì càng tốt.

Bạn cần phải học cách quản lý hệ thống Linux để trở thành quản trị viên hệ thống (system administrator hoặc sys admin). Công việc của quản trị viên hệ thống bao gồm: thêm bớt trương khoản cho những người sử dụng, đều đặn sao lưu dữ liệu, cài đặt thêm phần mềm mới, thiết lập cấu hình hệ thống, và giải quyết các hỏng hóc. Linux càng ngày càng phổ biến vì thế nguồn tài liệu hiện nay rất phong phú. Phần lớn các bản phát hành Linux đều kèm theo hàng ngàn trang tài liệu. Có thể dễ dàng tìm thấy những thông tin tương tự tại thư mục /DOCS trên các CD chứa Linux.

1.7 Thương mại hoá Linux

Cũng như mọi phần mềm, Linux chưa thể khắc phục hết ngay những bất tiện và sai sót. Nhưng rõ ràng càng ngày càng có thêm công ty mới đầu tư cho Linux và đưa ra các giải pháp ít nhiều có tính thương mại với giá rất rẻ. Xin nêu tên hai trong số các công ty đó là RedHat và Caldera.

Cả hai công ty này đều trợ giúp kỹ thuật qua e-mail, fax và qua mạng cho những người đã mua các phiên bản Linux và sản phẩm của họ mà không dành cho những người sao chép các bản miễn phí.

Vì tính kinh tế, Linux và các chương trình kèm theo thường được chạy trên mạng nội bộ của nhiều doanh nghiệp, chẳng hạn làm các dịch vụ Web, tên miền (DNS), định tuyến (routing) và bức tường lửa. Nhiều nhà cung cấp dịch vụ Internet (ISP) cũng dùng Linux làm hệ điều hành chính.

Ngoài việc phân phối RedHat Linux với RPM, doanh nghiệp RedHat còn có những sản phẩm khác, thí dụ bộ ứng dụng văn phòng Applixware, bao gồm một phần mềm xử lý văn bản, một phần mềm bảng biểu, một phần mềm trình diễn, một công cụ thư điện tử cùng với nhiều công cụ triển khai lập trình và giao diện đồ hoạ XFree86... Nhưng chỉ cần trả khoảng một nửa giá bán của riêng Windows XP thôi, bạn sẽ nhận được một bản RedHat kèm các phần mềm nói trên mà không cần phải mua thêm MS Office, v.v.

Caldera lúc đầu chỉ phát hành từ mạng Internet các sản phẩm dựa trên RedHat và Novell, trước khi có OpenLinux, một hệ điều hành giá rẻ với kernel 2.x. Sản phẩm này bao gồm một giao diện đồ hoạ có khả năng quản lý hệ thống và tài nguyên mạng, cùng với các ứng dụng mạng chủ yếu. OpenLinux tích hợp một X server thương mại của MetroLink và một phiên bản trình duyệt đã đăng ký đầy đủ của Netscape Navigator. Hiện nay, Caldera tách riêng OpenLinux thành 2 sản phẩm khác nhau: một để dùng cho máy tính cá nhân và một để dùng cho máy chủ. Caldera cũng phát hành bản Corel WordPerfect cho Linux, cùng với một bộ ứng dụng văn phòng hướng Internet. Ngoài ra Caldera còn phát triển phần mềm tương thích công nghệ WABI của SunSoft, cho phép người dùng cuối chạy các ứng dụng Windows trên nền Linux.

1.8 UNIX và Linux

Lịch sử Linux phát xuất từ UNIX và cụ thể liên quan đến Minix. Minix là một hệ điều hành nhỏ kiểu UNIX, minh hoạ bộ sách giáo khoa rất nổi tiếng do Tannebaum viết từ

giữa những năm 1980. Minix đã từng phổ biến trên nhiều máy tính mini và PC. Còn Bell Laboratories thuộc công ty AT&T là nơi hệ điều hành UNIX sinh ra, song chính các tập thể và cá nhân khác đã cải thiện UNIX qua nhiều năm. Từ năm 1969, Thompson và các cộng sự ở Bell Laboratories đã phát triển UNIX, một hệ điều hành rất linh động và phù hợp với nhiều yêu cầu khác nhau của giới lập trình. Khởi thuỷ, hệ điều hành MULTICS của Viện MIT đã gợi ý cho Thompson viết được sản phẩm của mình, nhưng sau này chỉ có UNIX trở thành một tiêu chuẩn công nghiệp cho các hệ điều hành đa nhiệm và đa người dùng.

Năm 1978, Berkeley Software Distribution (BSD) thuộc Đại học Berkeley tại California đã phát triển phiên bản UNIX đầu tiên của mình từ nền phiên bản UNIX v.7 của AT&T, với ý đồ sao cho UNIX trở nên thân thiện hơn với người sử dụng. Mặc dù không hoàn toàn tương thích với UNIX nguyên thuỷ của AT&T, phiên bản BSD UNIX vẫn đạt được mục tiêu đề ra nhờ những tiện ích mới đã làm nhiều người hài lòng.

Sau đó BSD đã phát hành FreeBSD, một phiên bản dành cho họ vi xử lý Intel 386 và phân phối khá hạn chế qua Internet hoặc CD-ROM, rồi các tác giả đã công bố bản này trên tạp chí Dr. Dobb's. Hiện nay bản thương phẩm của FreeBSD đã trở thành một hệ điều hành thông dụng tương tự như Linux.

UNIX System Laboratories (USL) là một công ty ra đời từ AT&T và đã từng triển khai UNIX System V từ đầu thập niên 1980. Trước khi được Novell mua lại hồi năm 1993, USL sở hữu mã nguồn của tất cả các phiên bản xuất xứ từ UNIX System V. Tuy nhiên hồi ấy USL chưa bán ra được những bản sẵn sàng cho người tiêu dùng. Bản phát hành đáng nhớ nhất của USL là UNIX System V Release 4.2 (SVR4.2). Đây là lần đầu tiên mà USL tham gia vào thị trường với qui mô lớn. Lúc ấy Novell và USL khai trương một công ty liên doanh mang tên Univel để sản xuất hàng loạt phiên bản SVR4.2 gọi là UnixWare. Khi mua lại USL, Novell đã chuyển vai trò trọng tâm của USL từ nhà sản xuất mã nguồn thành nhà phát hành UnixWare. Cuối cùng Novell lại bán UNIX của mình cho công ty Santa Cruz Operation (SCO). Gần đây SCO phát hành bản SCO UNIX một người dùng (single-user), tuy nhiên chi phí lên đến 19 USD, khó cạnh tranh được với Linux đa người dùng. Hơn nữa SCO không công bố mã nguồn hệ điều hành của mình.

Từ cuối thập niên 1970, Microsoft cũng đã từng phát triển phiên bản UNIX của mình, gọi là XENIX. Đến năm1981, trong thời kỳ cao điểm của cuộc cách mạng vi tính, máy tính cá nhân IBM-PC ra đời với hệ điều hành đơn nhiệm một người dùng DOS. Khả năng xử lý của PC tăng dần và bắt đầu sánh ngang các máy tính mini vào cuối thập niên 1980, khi sự ra đời của bộ vi xử lý Intel 386 cho phép XENIX có thể chạy trên PC. Microsoft và AT&T đã đồng ý nhập XENIX và UNIX vào thành một phiên bản duy nhất gọi là System V/386 Release 3.2, có khả năng hoạt động hầu như trên mọi cấu hình phần cứng của PC 386.

Sun Microsystems có đóng góp lớn lao vào việc mở rộng thị trường UNIX khi sản xuất ra các máy chủ và máy trạm chạy với hệ điều hành SunOS trên nền UNIX BSD. Cuối cùng BSD và SVR4 cũng đã hội tụ và tương thích với nhau.

IBM bước vào thế giới của UNIX bằng sản phẩm mang tên hệ điều hành AIX (Advanced Interative Executive). Các công ty HP và Apple cũng phát triển phiên bản UNIX của mình, gọi là HP-UX và A/UX. Mặc dù AIX, HP-UX và A/UX không nổi tiếng bằng vài phiên bản UNIX khác, song chúng chạy rất tốt và có một thị phần đáng kể.

Các công ty nói trên đều giữ bản quyền phiên bản UNIX của mình, trong khi DOS và MS Windows thuộc về Microsoft. Vậy ai là chủ sở hữu của Linux?

1.9 Tác quyền và bản quyền Linux

Nói chung, Linux không phải là phần mềm công cộng, bởi vì các thành tố của nó đã được nhiều người khác đăng ký tác quyền. Linus Torvalds giữ tác quyền về kernel Linux. Công ty RedHat là chủ của phiên bản RedHat Linux, và Patrick Voldkerding giữ tác quyền bản Slackware Linux v.v.

Nhưng nhiều tiện ích Linux lại có giấy phép công cộng GPL (GNU General Public License). Quả thực, Torvalds cùng nhiều người đóng góp cho Linux đã đặt công trình của mình dưới sự bảo vệ của GPL. Bạn có thể xem toàn văn GPL trên Internet hoặc trong tệp mang tên "copying" của mọi bản phát hành Linux. Bản quyền ấy đôi khi được gọi dí dỏm là Copyleft để đối lập chữ Copyright. GPL áp dụng cho phần mềm thuộc phong trào GNU (cũng chơi chữ: GNU's Not UNIX) và FSF (Free Software Foundation), cho phép tạo ra phần mềm tự do cho tất cả mọi người. Tự do hiểu là mỗi người đều có quyền sử dụng phần mềm GPL và tùy thích chỉnh sửa nó theo nhu cầu riêng của mình nhưng phải nhớ rằng không được giữ riêng bản chỉnh sửa ấy mà phải phổ biến rộng rãi để cho người khác cùng sử dụng và tiếp tục thay đổi theo ý họ.

GPL cho phép tác giả chương trình được giữ tác quyền pháp lý; song tác giả phải để cho người khác thao tác, thay đổi, và thậm chí bán chương trình mới được viết lại. Tuy nhiên một khi đã bán đi rồi thì người bán không được cấm người mua thay đổi chương trình đó và phải cung cấp mã nguồn. Đó là lý do tại sao Linux đến với bạn cùng toàn bộ mã nguồn đầy đủ và mở.

Chương 2. Chuẩn bị cài đặt Linux

Bạn sẽ tìm thấy sau đây các thông tin cần thiết trước khi cài đặt bất cứ bản phát hành Linux nào trên PC. Xin nhớ rằng Linux không phải là một thương phẩm, do đó cần phải chuẩn bị đối phó các trục trặc nếu có. Bạn có thể đọc thêm các HOW-TO, ngoài những mục hướng dẫn khá đầy đủ trong chương này như:

-Chọn cấu hình phần cứng

-Dung lượng đĩa và bộ nhớ

-Những cách cài đặt Linux

-Phân vùng ổ đĩa cứng

Ghi chú: Tài liệu này giả định rằng bạn đã có kiến thức về DOS và những thao tác như tạo khuôn dạng (format) ổ đĩa cứng, lập bảng phân vùng (partition) và kích cỡ của cung đĩa (sector). Nếu bạn còn chưa rõ những thuật ngữ này thì hãy tìm đọc một tài liệu về sử dụng DOS 6.2 hoặc nhờ sự giúp đỡ của một người hiểu biết máy PC.

Lưu ý: Cài đặt hệ điều hành có nghĩa là thay đổi cả hệ thống quản lý máy tính, vì vậy hãy cẩn thận và chuẩn bị sẵn giấy bút đề ghi chép các thông tin cần thiết.

2.1 Chọn cấu hình phần cứng

Điều kiện cài đặt Linux thành công là có các phần cứng phù hợp. Muốn chọn cấu hình cho tương xứng, bạn phải biết trước bao nhiêu người sẽ sử dụng hệ thống và sẽ chạy những ứng dụng nào. Từ đó bạn tính ra các yêu cầu về bộ nhớ, dung lượng ổ đĩa cứng, chủng loại thiết bị đầu cuối, v.v.

Ngày nay, đa số các máy tính có cài đặt Linux đều là PC và thường cũng chỉ cài đặt phiên bản cho một người sử dụng, mặc dù các máy ấy có thể liên kết với nhiều hệ thống Linux và UNIX lớn hơn.

Nếu bạn cài đặt phiên bản Linux cho một người dùng (trường hợp hay gặp nhất) thì bạn cũng là quản trị viên của hệ thống. Bạn có trách nhiệm hiểu rõ hệ thống để thực hiện chức năng quản trị, sao cho hệ thống chạy tối ưu. Bạn phải bảo đảm dung lượng tối thiểu trên ổ đĩa cứng, sao lưu đều đặn, các thiết bị kết nối với hệ đều có trình điều khiển (driver) và các phần mềm cài đặt thích hợp, v.v..

Bạn nên chọn lựa các loại phần cứng mà chính đa số những người tạo ra Linux đã sử dụng. Các công ty phát triển phần mềm thương phẩm thường chạy thử sản phẩm của họ trên nhiều phần cứng khác nhau, còn cộng đồng tình nguyện triển khai Linux chỉ có máy tính của chính mình.

Cũng may là cộng đồng Linux khá đông đảo cho nên hầu hết những phần cứng tiêu chuẩn của PC đều được Linux chấp nhận.

Lưu ý: Linux là một hệ thống tiến hoá và thỉnh thoảng lại có thông tin cập nhật. Bản phát hành RedHat sử dụng trong tài liệu này chạy khá ổn định, tuy nhiên thực tế có những phần cứng thay đổi mà chưa được Linux biết đến. Mặc dù nhiều phần cứng có

thể đã thay đổi bằng các linh kiện "nhái" hoặc tương thích Intel, song không phải tất cả những phần cứng ấy đều chạy được với Linux.

2.1.1 Bộ xử lý

Hệ thống phần cứng phù hợp Linux thường là một PC có bộ xử lý Intel 386 hoặc hiện đại hơn, chẳng hạn như 486, 586 hoặc Pentium. Những bộ xử lý nhái Intel như của Cyrix hoặc AMD cũng đều chấp nhận Linux.

Một số PC không có bộ đồng xử lý toán học, nhưng Linux không nhất thiết cần đến bộ phận này vì có thể phỏng tạo nó bằng cách sử dụng các chương trình con, dù rằng như thế sẽ giảm tốc độ thi hành.

Kernel Linux cũng được phát triển cho một số bộ xử lý khác, chẳng hạn như DEC Alpha, IBM PowerPC và Sun Sparc, thậm chí cho cả các bộ xử lý dùng trong hệ thống nhúng (embedded) như Network PC của Caldera.

2.1.2 Bus hệ thống

Linux thường chạy với các loại bus như ISA, EISA và PCI. Các kernel mới của Linux (từ 2.2 trở đi) có thể chạy với bus AGP. Với bus MCA trên máy tính PS/2 của IBM, chỉ các bản kernel từ 2.0.7 là chạy được. Một số hệ thống sử dụng loại bus cục bộ, gọi là VLB, để truy cập đĩa cứng và hiển thị màn hình nhanh hơn cũng được Linux chấp nhận.

2.1.3 Bộ nhớ

Linux không đòi hỏi nhiều RAM, nhất là khi so sánh với các hệ điều hành khác như Windows 2000, XP hoặc Windows NT.

Theo kernel và HOW-TO phiên bản ngày 11-7-2001, Linux chỉ cần 2 MB RAM, nhưng trong thực tế sử dụng thì Linux cần ít nhất 4 MB RAM. Thật sự, cấu hình thấp với 4 MB RAM chỉ có thể chạy ở chế độ văn bản, không có giao diện đồ hoạ. Từ phiên bản RedHat v.7.2, bạn cần ít nhất là 64 MB RAM và hiện nay 128 MB là yêu cầu trung bình. Nếu có ít hơn 4 MB RAM, bạn phải chạy với tệp hoán chuyển (swap file) ở trên đĩa cứng, được dùng như bộ nhớ ảo và do đó làm chậm hệ thống. Lượng RAM cần thiết còn phụ thuộc vào việc bạn sử dụng máy để làm gì. Càng muốn có nhiều chức năng bạn càng cần thêm RAM. Khi bạn dùng máy để quản trị một cơ sở dữ liệu thì lượng RAM cần thiết sẽ tăng lên rất nhiều.

Việc sử dụng giao diện đồ hoạ X Window (bằng phần mềm XFree86) làm tăng nhu cầu về bộ nhớ. Bạn cần ít nhất 8 MB RAM vật lý và 8 MB tệp hoán chuyển, tức là 16 MB RAM ảo để có một hệ thống hoạt động hiệu quả.

2.1.4 Đĩa cứng

Bạn có thể khởi động Linux từ một đĩa mềm. Mặc dù trên nguyên tắc vẫn có thể chạy Linux từ đĩa mềm, song thực tế không ai làm như thế.

Nếu sử dụng PC ở nhà, bạn cần có một ổ đĩa mềm loại 3.5" (1.44 MB), cho dù có thể chạy thẳng Linux từ CD với bản demo của SUSE Linux 7.3 hoặc KNOPIX.

Để hệ thống chạy hiệu quả hơn bạn nên cài đặt Linux vào ổ đĩa cứng có giao diện IDE. Linux chấp nhận cả giao diện ESDI, nhưng đối với loại ổ đĩa IDE cải tiến (tức EIDE) thì chỉ có các bản kernel Linux từ 2.2.x trở lên mới tương thích hoàn toàn.

Linux chấp nhận giao diện SCSI với bìa điều khiển của các hãng Adaptec, Future Domain, Seagate, UltraStore, cũng như với các bộ thích nghi (adapter) trên bìa ProAudio Spectrum 16 và Western Digital.

Xem "SCSI Controller" trong các HOW-TO về Linux

2.1.4.1 Dung lượng ổ đĩa cứng

Sau khi có bìa điều khiển thích hợp cho ổ đĩa rồi, bạn phải quan tâm đến các yêu cầu về dung lượng ổ đĩa. Linux chấp nhận một lúc nhiều ổ đĩa cứng và có thể cài đặt nó không cùng trên một ổ duy nhất.

Muốn sử dụng Linux cho có hiệu quả, bạn phải phân vùng lại ổ đĩa cứng và cấp phát đủ dung lượng đĩa cho các tệp hệ thống Linux và cho các tệp dữ liệu của bạn. Phân vùng (Partition) là chia những vùng theo ý của người sử dụng khi bắt đầu thiết lập thông số cho ổ đĩa cứng và trước khi định dạng ổ đĩa cứng.

Dung lượng đĩa cần thiết tuỳ thuộc vào phần mềm bạn sẽ cài đặt và số lượng dữ liệu mà phần mềm ấy sinh ra. So với hầu hết các hệ điều hành kiểu UNIX, Linux đòi hỏi dung lượng đĩa ít hơn. Bạn có thể chạy toàn bộ hệ Linux (không có phần X Window-tức là chỉ ở Text mode) với chỉ 80 MB (bản kernel 2.2.4-10). Nếu cài đặt không sót một thứ gì trong bản phát hành, bạn sẽ cần từ 1.8 GB đến 3.5 GB tuỳ theo phiên bản và nhà sản xuất.

Thông thường lệnh DOS **fdisk** hoặc một vài thương phẩm khác cho phép bạn phân chia lại ổ đĩa cứng và Linux cũng có tiện ích tương tự gọi là FIPS.

Chú ý: Nếu bạn cài đặt Linux vào một ổ đĩa cứng mới nguyên thì không sao, còn đối với ổ đang dùng thì phải phân vùng và định dạng lại. Việc này sẽ xoá sạch toàn bộ thông tin trên ổ đĩa cứng, do đó bạn phải sao lưu cẩn thận trước khi cài đặt Linux. Nếu ổ đĩa cứng có dung lượng lớn, bạn có thể phân thành nhiều vùng và sao chép thông tin trở lại vào các vùng đã khai báo.

2.1.4.2 Phân vùng hoán chuyển

Như đã nói ở trên, nếu bạn có ít RAM thì phải cần đến phân vùng hoán chuyển (swap partition).

Một số hệ điều hành như Microsoft Windows lưu trữ tệp hoán chuyển trên ổ đĩa cứng như bất kỳ tệp nào khác, trong khi đó Linux cho phép tệp hoán chuyển cư trú trên một phân vùng dành riêng cho nó. Khi cài đặt, nhiều người sử dụng phân vùng hoán chuyển thay vì tệp hoán chuyển. Bởi vì có thể tạo ra nhiều phân vùng trên cùng một ổ đĩa cứng vật lý nên bạn có thể đặt phân vùng hoán chuyển trên cùng ổ đĩa với Linux. Tuy nhiên nếu đặt phân vùng hoán chuyển trên ổ đĩa khác, Linux sẽ chạy tốt hơn. Linux cho phép bạn tạo ra đến 8 phân vùng hoán chuyển. Nên đặt kích cỡ phân vùng hoán chuyển to gấp đôi số lượng RAM vật lý của máy bạn. Ví dụ máy bạn có 8 MB RAM thì phân vùng hoán chuyển nên là 16 MB.

2.1.5 Yêu cầu về màn hình

Đối với các thiết bị cuối làm việc ở chế độ văn bản (ASCII terminal), Linux chấp nhận tất cả mọi loại màn hình (video monitor) và bìa điều khiển màn hình (video adapter) hợp các chuẩn Hercules, CGA, EGA, VGA và SuperVGA.

Khi làm việc ở chế độ đồ hoạ, bạn cũng có thể chạy được bất kỳ tổ hợp màn hình và bìa điều khiển nào. Để tận dụng việc Linux có khả năng hiển thị đầy đủ các màu, bạn nên sử dụng màn hình màu.

Nhưng phiền toái có thể sinh ra khi bạn chạy XFree86 (phiên bản giao diện đồ hoạ X Window phát hành kèm theo Linux). Muốn chạy XFree86 bạn cần có một trong các bìa điều khiển màn hình liệt kê tại bảng sau.

Ghi chú: Bộ xử lý đồ hoạ là một nhóm các mạch tích hợp (chip, intergrated circuit) có chức năng lấy thông tin đầu ra từ máy vi tính và chuyển chúng thành một hình gồm những điểm sáng hiển thị trên màn hình. Muốn biết chính xác bìa điều khiển dùng bộ xử lý đồ hoạ nào, cần xem kỹ hồ sơ đi kèm với bìa đó.

Một số nhà sản xuất bìa điều khiển màn hình không cung cấp đủ thông tin cần thiết để lập trình trong XFree86, do đó khi chạy phần mềm này có thể các thông tin không được hiển thị trơn tru. Một vài hãng bằng lòng cung cấp thông tin nhưng đòi hỏi phải trả phần trăm quyền sở hữu, hoặc yêu cầu bảo mật.

Ghi chú: Trước đây các bìa điều khiển màn hình của hãng Diamond không chạy được với Linux vì những lý do liên quan đến quyền sở hữu. Hiện Diamond đã bắt đầu làm việc với nhóm XFree86 để tìm giải pháp tương thích với Linux.

Xem "Video Cards" trong các HOW-TO về Linux

2.1.6 Ổ CD

Muốn cài đặt Linux từ đĩa CD, máy bạn phải có ổ CD tương thích với Linux. Đa số các ổ CD trước kia sử dụng giao diện SCSI, do đó bất kỳ bộ điều khiển SCSI nào được liệt kê ở mục "Ổ đĩa cứng" nói trên đều được Linux chấp nhận. Hiện nay Linux tương thích với nhiều ổ CD loại mới, sử dụng giao diện EIDE và ATAPI đang có trên thị trường.

Nhiều ổ CD được bán theo dạng trọn gói multimedia có thể tương thích hay không tương thích với Linux, tuỳ vào việc bộ điều khiển có giao diện chuẩn SCSI thật hay chỉ là bộ thích nghi theo chuẩn riêng. Hầu hết các bộ thích nghi theo chuẩn riêng không hoạt động với Linux. Tuy nhiên Linux lại tương thích với các ổ CD kiểu Creative Labs Soundblaster và cung cấp một cấu hình cài đặt riêng biệt cho các CD này. Sau đây là một số ổ CD tương thích với Linux:

2.1.6.1 Các ổ đĩa CD phổ quát

Các ổ đĩa có giao diện SCSI (xem tài liệu CD HOW-TO): Bất kỳ ổ SCSI CD với khối (block) 512 hay 2048 bytes đều có thể làm việc được trong Linux.

Các ổ đĩa có giao diện EIDE (ATAPI) CD và IDE CD: Hầu như tất cả các ổ đĩa CD tốc độ 2X, 4X, 6X đều được Linux hỗ trợ

Thí dụ các ổ đĩa CD phổ quát : Mitsumi FX400, Nec-260, Sony 55E v.v.

2.1.6.2 Các ổ đĩa CD đặc chủng

Aztech CDA268-01A, Orchid CDS-3110, Okano/Wearnes CDD-110

Conrad TXC, CyCDROM CR520ie/CR540ie/CR940ie (AZTCD)

Creative Labs CD-200(F) (SBPCD)

Funai E2550UA/MK4015 (SBPCD)

GoldStar R420 (GSCD)

IBM External ISA (SBPCD)

Kotobuki (SBPCD)

Lasermate CR328A (OPTCD)

LMS Philips C MB 206 (CM206)

Longshine LCS-7260 (SBPCD)

Matsushita/Panasonic CR-521/522/523/562/563/(SBPCD)

MicroSolutions Backpack parallel portdrive (BPCD)

Mitsumi CR DC LU05S (MCD/MCDX)

Mitsumi FX001D/F (MCD/MCDX)

Optics Storage Dolphin 8000AT (OPTCD)

Sanyo CDR-H94A (SJCD)

Sony CDU31A/CDU33A (CDU31A)

Sony CDU-510/CDU-515 (SOMYCD535)

Sony CDU-535/CDU-531 (SONYCD535)

Teac CD-55A SuperQuad (SBPCD)

Xem "Những loại ổ CD được chấp nhận" trong các HOW-TO về Linux

2.1.7 Truy cập mạng

Hiện nay có thể kết nối các hệ thống Linux bằng nhiều cách, song hai cách phổ biến nhất và có sẵn nhiều thiết bị nhất là sử dụng giao diện mạng hoặc modem.

Các giao diện mạng bao gồm Token Ring, FDDI, ATM và Ethernet. Hầu hết các mạng thông thường đều sử dụng giao diện Ethernet.

2.1.7.1 Truy cập qua Ethernet

Khởi thuỷ do Xerox, DEC và Intel đề ra, Ethernet từ lâu đã trở thành giao diện mạng phổ biến nhất. Mặc dù ở nhà ít ai dùng máy Linux để kết nối vào mạng Ethernet, song ở các cơ quan, doanh nghiệp và trường học thì đó là điều thường thấy.

Bảng sau đây liệt kê những bìa giao diện mạng Ethernet được Linux chấp nhận:

3Com 3c501, 3c503, 3c505, 3c507

3Com 3c509/3c509B (ISA)

3Com 3c579 (EISA)

3Com Etherlink III Vortex Ethercards 3c590, 3c592, 3c595, 3c597 (PCI)

3Com Etherlink XL Boomerang 3c900, 3c905 (PCI)
3Com Cyclone 3c905B, 3c980
3Com Fast EtherLink Ethercard 3c515 (ISA)
3Com 3ccfe575 Cyclone Cardbus (3c59x driver, PCMCIA)
3Com 3c575 series Cardbus (3c59x driver, PCMCIA)
AMD LANCE 79C960/PCnet (ISA/PCI)
AT&T GIS WaveLAN
Allied Telesis AT1700
Allied Telesis LA100PCI-T
Allied Telesyn AT2400T/BT ("ne" module)
Ansel Communications AC3200 (EISA)
Apricot Xen-II/82596
Cabletron E21xx
Cogent EM110
Crystal Lan CS8920, Cs8900
Danpex EN-9400
DEC DE425 (EISA) / DE434/DE435 (PCI) / DE450/DE500 (DE4x5 driver)
DEC DE450/DE500-XA (dc21x4x) (Tulip driver)
DEC DEPCA và EtherWORKS
DEC EtherWORKS 3 (DE203, DE204, DE205)
DEC QSilver's (Tulip driver)
Digi International RightSwitch
DLink DE-220P, DE-528CT, DE-530+, DFE-500TX, DFE-530TX
Fujitsu FMV-181/182/183/184
HP PCLAN (27245 và 27xxx series)
HP PCLAN PLUS (27247B và 27252A)
HP 10/100VG PCLAN (J2577, J2573, 27248B, J2585) (ISA/EISA/PCI)
ICL EtherTeam 16i/32 (EISA)
Intel EtherExpress
Intel EtherExpress Pro
KTI ET16/P-D2, ET16/P-DC ISA
Macromate MN-220P (PnP hoặc NE2000 mode)
NCR WaveLAN
Novell NE2000/NE1000
Netgear FA-310TX (Tulip)
New Media Ethernet
PureData PDUC8028, PCI8023
SEEQ 8005
SMC Ultra/EtherEZ (ISA)
SMC 9000 series

SMC PCI EtherPower 10/100 (Tulip driver)

SMC EtherPower II (epic100.c driver)

Sun LANCE adapters (kernel 2.2 và mới hơn)

Sun Intel adapters (kernel 2.2 và mới hơn)

Schneider & Koch G16

Western Digital WD80x3

Zenith Z-Note / IBM ThinkPad 300 built-in adapter

Znyx 312 EtherArray (Tulip driver)

Xem "Network Interface Cards" trong các HOW-TO về Linux

2.1.7.2 Truy cập qua modem

Khi làm việc ở nhà, bạn thường kết nối với bên ngoài qua modem và mạng điện thoại bằng các giao thức liên lạc nối tiếp như SLIP hoặc PPP.

Nói chung, Linux tương thích với mọi loại modem sử dụng cổng Serial RS-232 đang bán trên thị trường. Đa số các loại modem nối qua cổng USB và modem gắn trong (internal modem) các PC cũng chạy được dưới Linux.

Trong một số trường hợp, bạn không có trình điều khiển riêng cho modem chạy với Linux và sẽ phải sử dụng trình điều khiển phổ quát (generic driver).

Nếu sử dụng được một modem dưới DOS thì bạn sẽ không có khó khăn gì cho nó chạy dưới Linux.

Dưới đây là danh sách các modem tương thích Linux hiện nay:

3Com 3CXM256/3CCM256 và 3CXM656/3CCM656 PCMCIA

AOpen FM56-P và FM56-H

AT&T/Lucent winmodem

Boca Research 28.8 internal modem (model MV34AI)

Boca Research 33.6 internal modem (model MV34)

MC2920A-3.3, E6030D 4035-01 và 1721 8011 A

Cirrus Logic CL-MD3450D-SC-B

Cirrus Logic MD1724-11VC-D

Datatronic VLM301-1

Omron G5V-1

AST M628032-20E1

Cirrus Logic CD-MD4450C-SC-A

Abracon 23-040-20

Compaq 192PCMCIA modem/serial card

HP Fastmodem D4810B

IBM Mwave ("Dolphin" card

Multiwave Innovation CommWare V.34 modem

Megahertz XJ/CC2560 PCMCIA

New Media Winsurfer PCMCIA modem/serial card

```
Rockwell SoftK56
US Robotics WinModem Series
Zoltrix 33.6 Win HSP Voice/Speaker Phone modem
Zoltrix Phantom 56K, model FM-HSP56PCI, bộ xử lý PCTel (PCI)
```

Xem "SLIP", "PPP" và "Modems" trong các HOW-TO về Linux

2.1.8 Các thiết bị khác

Những mục sau đây sẽ nói về tính tương thích với Linux của các thiết bị ngoại vi khác như chuột, ổ băng từ, máy in. Các thiết bị này giúp bạn sử dụng Linux tiện lợi hơn, song không nhất thiết phải có.

2.1.8.1 Chuột

Chuột máy tính là thiết bị dùng để điều khiển con chạy (cursor) trên màn hình.

Nếu chỉ chạy ở chế độ văn bản thì chẳng cần đến chuột, mặc dù Linux cho phép bạn dùng chuột cắt những đoạn chữ từ màn hình rồi dán sang dòng lệnh, trong khi UNIX thường không làm được như vậy.

Muốn chạy ở chế độ đồ hoạ dưới giao diện X Window thì bạn phải sử dụng chuột. Linux tương thích với hầu hết các loại chuột nối tiếp của những hãng như Logitech, Kensington, Mouseman, Microsoft, v.v. Linux cũng chấp nhận các loại chuột bus của Microsoft, Logitech, ATIXL và IBM, v.v.

Nhiững thiết bị khác dùng để điều khiển con chạy như quả cầu (trackball) và màn hình xúc giác (touch screen), mô phỏng các loại chuột vừa liệt kê, cũng đều chạy được với Linux.

2.1.8.2 Ổ băng từ

Ổ băng từ có dung lượng lớn nên cho phép thoải mái sao lưu dữ liệu hệ thống. Linux tương thích với nhiều ổ băng từ có giao diện SCSI được liệt kê ở bảng dưới đây. Linux cũng chấp nhận các ổ băng từ rẻ tiền như Colorado Memory Systems loại 120 MB và 250 MB, được cắm thẳng vào giao diện điều khiển ổ đĩa. Các ổ băng từ cắm thẳng vào cổng song song (cổng máy in) hiện chưa được Linux chấp nhận. Hầu hết các ổ băng từ tương thích chuẩn QIC-02 chạy được với Linux.

Sau đây là một số ổ băng từ tương thích với Linux:

Hãng sản xuất	Kiểu
Exabyte	Tất cả các kiểu có giao diện SCSI
Sanko	CP150SE
Tandberg	3600
Wangtek	552ES, 515ES, 5099EN

Xem "Magnetic Tape Drives" trong các HOW-TO

2.1.8.3 Máy in

Linux tương thích với toàn bộ các máy in nối qua cổng song song. Với máy in song song, thì khó khăn lớn nhất có thể gặp là hiệu ứng nấc thang. Hiệu ứng nấc thang xảy ra do cách UNIX và Linux xử lý khi xuống dòng và điều khiển đầu in quay về đầu dòng. Với UNIX, lệnh đưa giấy lên thêm một dòng (LF: line feed) sau đó đặt đầu in tại vị trí đầu dòng mới (CR: carriage return) do một ký tự duy nhất là LF điều khiển, trong khi các hệ như DOS hoặc Windows lại sử dụng cặp ký tự CR-LF cho hai lệnh trên. Khi bạn in một tệp UNIX bằng máy in được cấu hình cho DOS, bạn sẽ bị hiệu ứng nấc thang bởi vì tệp chỉ chứa ký tự LF chứ không chứa ký tự CR.

Ở chiều ngược lại, các tệp văn bản soạn trong môi trường DOS/Windows cũng cần được chỉnh lý (cặp ký tự CL-LF đổi thành LF) khi chuyển sang môi trường Linux hoặc UNIX.

Việc đặt cấu hình Linux để chạy với máy in nối tiếp thường khó hơn. Các chương trình cài đặt Linux về cơ bản không có sẵn công cụ hỗ trợ máy in nối tiếp.

Xem "Lập cấu hình máy in"

2.2 Dung lượng đĩa và bộ nhớ

Tuỳ theo cách cài đặt, mở đầu bạn có thể cần 2 đĩa mềm loại 1.44 MB đã được định dạng để tạo ra đĩa mồi cho Linux. Sau đó bạn phải dành đủ dung lượng đĩa cứng; nếu cài hết mọi thứ có trên CD thì phải cần 3.5 GB, tuy nhiên vẫn có thể sử dụng ít hơn. Cần tính xem có bao nhiêu trương khoản người dùng. Nếu hệ thống chỉ có một trương khoản thì 80 MB là đủ cho RedHat 6.1, nếu không cài X Window. Tuy nhiên, càng ngày yêu cầu về dung lượng đĩa càng tăng. Bản RedHat 7.2 cài tối thiểu cũng cần tới 350 MB đĩa cứng.

Tiếp theo bạn tính xem cần bao nhiêu cho vùng hoán chuyển (swap space) mà thường là khoảng 500 MB. Nhưng nếu bạn phải tạo một máy chủ cơ sở dữ liệu (database server) thì dung lượng này là không đủ. Đặc biệt với ORACLE 9i, bạn cần vùng hoán chuyển lớn gấp 2 hay 3 lần bộ nhớ vật lý (mà bộ nhớ vật lý tối thiểu cho ORACLE 9i đã là 256 MB).

Cuối cùng bạn nên chừa ít nhất 1 GB cho thư mục gốc (root). Đây là thư mục chính mà từ đó bạn truy cập đến tất cả các thư mục thứ cấp của Linux.

Xem "Các thư mục chuẩn của Linux"

Cần nhắc lại là cách cài đặt Linux tối thiểu sẽ chỉ cần 350 MB (bản RedHat 7.2), trong khi cài đặt đầy đủ và dành chỗ cho nhiều user sẽ cần khoảng 5 GB.

Ghi chú: Bạn có thể chạy một phần hệ thống Linux từ CD mà không nhất thiết phải cài đặt toàn bộ lên đĩa cứng, song như thế không đọc được các CD khác.

Nếu bạn quyết định dùng giao diện đồ hoạ X Window, trước tiên nên ghi ra giấy xem bìa đồ hoạ của bạn được bộ xử lý nào điều khiển. Nếu bạn có chuột nối tiếp và modem, cũng nên ghi rõ tên cổng nối tiếp của chúng. Trong quá trình cài đặt bạn sẽ cần những thông số ấy.

2.3 Những cách cài đặt Linux

Có đến 4 cách cài đặt Linux, đó là từ CD, NFS, FTP, hoặc từ ổ đĩa cứng.

a) Cách phổ biến nhất là cài đặt Linux từ đĩa CD. Muốn cài đặt trực tiếp từ CD, bạn phải khởi đầu bằng DOS. Ở dấu nhắc DOS, bạn gõ lệnh:

```
[ổ CD]:\dosutils\autoboot
```

Trong đó [ổ CD] là tên ổ CD trên máy của bạn.

Có thể bắt đầu bằng việc đặt lại cấu hình cho BIOS để có thể khởi động bằng đĩa CD. Đa số các BIOS sản xuất sau 1997 đều hỗ trợ việc này. Khi đó, bạn chỉ cần tắt nguồn điện và khởi động lại với đĩa CD Linux đặt sẵn trong ổ.

Nếu ổ đĩa cứng có sẵn một phân vùng trống, bạn có thể cài đặt Linux vào đó bên cạnh hệ thống hiện hành để khỏi xoá mất những thông tin trước đó. Như vậy những gì bạn cần là một ổ CD, một phân vùng trống và một đĩa mồi.

b) Cách cài đặt Linux từ mạng máy tính nhờ một máy chủ NFS (Network hệ thống tệp) yêu cầu bạn trước hết phải lắp ráp logic (**mount**) ổ CD vào một máy tính chấp nhận hệ thống tệp ISO-9660 với phần mở rộng RockRidge, rồi bạn công bố hệ thống tệp qua NFS. Bạn phải biết đường dẫn của hệ thống tệp này, cũng như địa chỉ IP của máy, hoặc tên của máy nếu có DNS.

c) FTP (File Transfer Protocol) là một giao thức truyền tệp qua mạng LAN hoặc WAN. Việc cài đặt qua FTP yêu cầu phải có đĩa mồi và đĩa phụ trợ.

d) Cài đặt Linux từ ổ đĩa cứng cũng cần đĩa mồi và các đĩa phụ trợ nói trên. Trước tiên phải tạo ra thư mục mang tên Linux, sau đó sao chép thư mục tương ứng từ CD cùng với tất cả các thư mục thứ cấp (subdirectory) vào thư mục Linux.

Bạn có thể sử dụng các lệnh DOS sau đây để thi hành việc cài đặt:

```
cd\Linux
xcopy/s e:\Linux
```

Lệnh **cd** chọn thư mục làm việc là thư mục Linux trên ổ đĩa cứng; lệnh **xcopy** sao chép thư mục tương ứng từ đĩa CD đặt ở ổ E.

Cho dù bạn sử dụng phương pháp cài đặt nào, bạn luôn cần có đĩa mồi. Nhưng trước hết bạn phải tìm một số thông tin cần thiết.

2.3.1 Tìm các thông tin cần thiết

Trước khi bắt đầu cài đặt, bạn cần tìm các thông tin sau đây về máy của bạn:

-Loại bìa điều khiển, bộ xử lý đồ họa và màn hình

-Cổng nối tiếp của chuột

-Cổng nối tiếp của modem

-Thông tin về địa chỉ IP, cổng gateway và tên miền (nếu kết nối với mạng)

-Loại ổ đĩa cứng, ổ CD và các bộ điều khiển của chúng

-Tên dự định đặt cho hệ thống

-Nếu kết nối Internet, cần yêu cầu ISP cung cấp các thông tin liên quan.

Nếu bạn dự định sử dụng một lúc vài hệ điều hành khác nhau trên cùng một máy (chẳng hạn như Windows95, Windows NT hoặc Windows 2000), bạn phải tạo ra các phân vùng riêng cho những hệ điều hành ấy.

Bạn cần sử dụng chương trình tạo phân vùng của chính hệ điều hành gốc, bởi vì Linux không quản lý được hết mọi loại phân vùng của các hệ điều hành khác. Sau đó bạn nên kiểm tra xem vào giờ chót bản phát hành Linux có những thay đổi nào mới không bởi vì Linux thường xuyên được nâng cấp, gỡ lỗi, hoặc bổ sung các tiện ích.

Nếu không cài đặt trực tiếp từ CD, bạn sẽ phải phân vùng lại ổ đĩa cứng để dành chỗ cho Linux. Ở đây có thể sinh vấn đề bởi vì việc phân vùng lại ổ đĩa cứng sẽ xoá toàn bộ dữ liệu nằm trên các phân vùng bị tác động. Sau khi dành chỗ cho Linux, bạn phải khởi động hệ thống Linux để tạo các phân vùng mới và hệ thống tệp cho Linux. Thông thường Linux cần một phân vùng ban đầu để lưu trữ tệp và một phân vùng dành cho tệp hoán chuyển, nhất là khi bộ nhớ máy bạn chỉ có từ 8 MB trở xuống (nếu phải cài đặt trên một máy có cấu hình thấp, bạn chỉ nên sử dụng các bản RedHat Linux cũ - từ 6.0 trở xuống).

Ghi chú: hệ thống tệp về cơ bản là một phân vùng trên ổ đĩa cứng hay trên một thiết bị lưu trữ nào đó và được định dạng theo một chuẩn nhất định. UNIX và cả Linux sử dụng được nhiều loại hệ thống tệp để biểu đạt từng nhánh thứ cấp của cây thư mục. Điều này không giống với DOS, vì DOS đặt các thư mục thứ cấp của cây thư mục vào cùng một ổ đĩa logic.

Xem "Tìm hiểu hệ thống tệp Linux"

Sau khi tạo ra các hệ thống tệp, bạn có thể cài đặt hệ điều hành Linux, các tệp hỗ trợ cùng với vài gói ứng dụng phát hành kèm theo Linux. Để cài đặt Linux, đầu tiên bạn khởi động một phiên bản stripdown của hệ điều hành. Muốn làm được việc này, bạn phải tạo ra một đĩa mồi (boot) và một đĩa phụ trợ (supp) có chứa phiên bản stripdown.

2.3.2 Tạo ra đĩa mồi và đĩa phụ trợ

Bạn phải dùng chương trình rawrite, chương trình này nằm trong thư mục thứ cấp /dosutils của CD Linux. Bạn cần chuẩn bị hai đĩa mềm đã định dạng, một đĩa ghi nhãn "boot" và đĩa kia ghi nhãn "supp". Bạn đặt đĩa mồi vào ổ A và gõ:

```
E:\dosutils>rawrite

Enter disk image source file name: e:\images\boot.img

Enter target diskett drive: A:

Please insert a formatted diskette into drive A: and press –ENTER-
```

Nếu không muốn tiếp tục, bạn chỉ cần bấm <Ctrl-c>. Còn đi tiếp mà lệnh rawrite không hoạt động được thì bạn thử cho vào một đĩa mềm khác đã định dạng. Nếu vẫn không tiến triển tốt, bạn phải kiểm tra lại phần cứng.

Sau khi ghi đĩa mồi, bạn tạo ra đĩa phụ trợ. Bạn chỉ cần gõ tên tệp supp.img như là tên tệp nguồn tại dòng lệnh.

2.4 Phân vùng ổ đĩa cứng

Sau khi đã sao lưu dữ liệu và tạo ra đầy đủ các đĩa mồi và đĩa phụ trợ, hãy chuẩn bị ổ đĩa cứng cho Linux.

Lưu ý: Đây là một thao tác nguy hiểm nhất bởi vì sẽ xóa mất dữ liệu cũ. Nếu chưa sao lưu hệ thống, bạn hãy làm ngay đi. Mặc dù có thể sử dụng chương trình thử nghiệm FIPS và các chương trình thương mại như Partition Magic để phân chia lại ổ đĩa cứng mà không phá huỷ dữ liệu, chúng tôi vẫn khuyên bạn sao lưu toàn bộ và đầy đủ.

2.4.1 Tìm hiểu về phân vùng

Khi PC xuất hiện, phần lớn các hệ điều hành, chương trình và dữ liệu đều để trên đĩa mềm. Khi máy PC XT ra đời, hãng IBM mới có thêm ổ đĩa cứng 10 MB. Các hệ điều hành sơ khai như DOS chỉ truy cập được một dung lượng rất hạn chế trên ổ đĩa cứng. Sau đó các nhà sản xuất hàng năm đều tăng dung lượng của ổ đĩa cứng khiến cho DOS khó theo kịp để quản lý những dung lượng mới lớn hơn. DOS tránh né vấn đề bằng cách cho phép người sử dụng chia ổ đĩa cứng thành nhiều vùng logic, gọi là phân vùng. Các phân vùng này lưu giữ được những tệp chương trình, các hệ điều hành khác, hoặc dữ liệu. DOS thường được khởi động tại một ổ đĩa cứng gọi là ổ C. Nếu chia ổ này làm ba thì các phân vùng ổ đĩa logic sẽ gọi là C, D, E. DOS cho phép lắp nhiều ổ đĩa (ổ cứng hoặc CD), do đó nếu lắp thêm một ổ đĩa nữa, nó sẽ được gọi là F, v.v.

UNIX và Linux không dùng các chữ cái ấy để gọi phân vùng, mà dùng cách khác. Người sử dụng Linux có thể đặt nhiều thư mục khác nhau trên những phân vùng khác nhau (của cùng một ổ đĩa cứng) và ngay cả trên các ổ đĩa cứng khác. Bạn cũng có thể đặt các hệ điều hành khác nhau trên các phân vùng khác nhau.

Thông tin phân vùng được ghi rõ trên cung đầu tiên của ổ đĩa cứng gọi là Master boot record (MBR) và mang tên là bảng phân vùng. Bảng này được sử dụng để biết xem phải khởi động hệ điều hành ở phân vùng nào. Chức năng của MBR là mồi (boot), nghĩa là để móc nối với cơ chế khởi động hệ điều hành. Chương trình mồi LILO của Linux và các phần mềm quản lý mồi khác đều sử dụng MBR để biết xem nên khởi động hệ điều hành nào.

Bảng phân vùng ghi rõ vị trí và kích thước của nhiều phân vùng trên ổ đĩa cứng. Có ba loại phân vùng: sơ cấp (Primary), mở rộng (Extended) và logic (Logical). DOS và vài hệ điều hành khác phải khởi động từ phân vùng sơ cấp. Chỉ có thể tạo tối đa 4 phân vùng sơ cấp trên một ổ đĩa cứng. Bản thân một phân vùng mở rộng không chứa dữ liệu mà chỉ ghi lại cách phân vùng cho các phân vùng khác trên ổ đĩa cứng. Số phân vùng logic trên một ổ đĩa cứng là không giới hạn. Do đó để giải quyết giới hạn của bốn phân vùng sơ cấp, bạn có thể chỉ định một phân vùng mở rộng và khai báo một số phân vùng logic khác ở bên trong phân vùng mở rộng.

DOS và các phiên bản của OS/2 trước phiên bản 2.0 đòi hỏi phải được cài đặt trên một phân vùng sơ cấp, tuy nhiên chúng có thể truy cập các ổ logic bên trong các phân vùng mở rộng. Việc này rất quan trọng nếu bạn muốn cài DOS và Linux trên cùng ổ đĩa cứng. DOS phải nằm trên phân vùng sơ cấp.

2.4.2 Sử dụng lệnh FDISK

Trên PC các phân vùng được tạo ra, xoá đi và quản lý bởi một chương trình gọi là **fdisk**. Mỗi hệ điều hành có **fdisk** riêng của mình, do đó trước khi sử dụng bạn phải dò lại xem có đúng phiên bản hay chưa. Nếu đang sử dụng DOS hoặc dự định sử dụng nó, trước tiên bạn phải phân chia lại ổ đĩa cứng bằng **fdisk** của DOS. Sau này bạn sẽ dùng **fdisk** của Linux để tạo phân vùng của Linux.

2.4.2.1 Các yêu cầu về phân vùng

Trước tiên bạn phải xác định mình cần bao nhiêu phân vùng. Trong khi DOS cần phân vùng sơ cấp thì Linux có thể cư trú trên các phân vùng khác. Nên nhớ nếu bạn nén một phân vùng hiện hành của DOS để dành chỗ cho Linux, thì tất cả các tệp của bạn không thể phục hồi hết trên phân vùng DOS nhỏ hơn mới được tạo.

Sau đó bạn quyết định số phân vùng cần thiết và mỗi phân vùng như thế cần bao nhiêu dung lượng ổ đĩa cứng.

Ghi chú: Từ Linux, bạn có thể vào các phân vùng của DOS và thực hiện các lệnh di dời, sao lưu, chỉnh sửa các tệp DOS, nhưng không thể chạy các chương trình DOS bằng Linux.

Hai phần mềm Linux cho phép bạn phỏng tạo DOS trên Linux và cài đặt Linux trên DOS. Cả hai hệ này chủ yếu thích hợp cho các hacker. Một trong những phần mềm ấy, gọi là UMSDOS, lại không tương thích với RedHat Linux.

Xem "Chạy các chương trình DOS trên Linux"

2.4.2.2 Các yêu cầu về DOS

Nếu bạn khởi động bằng DOS, máy sẽ vào một phân vùng sơ cấp. Một phiên bản mồi được (bootable) của DOS không đòi hỏi nhiều chỗ trên ổ đĩa cứng, chỉ cần đủ chỗ cho các tệp hệ thống COMMAND.COM, CONFIG.SYS và những driver cần thiết để khởi động hệ thống. Thực tế chỉ cần 5 MB cho phân vùng sơ cấp để khởi động DOS. Một khi đã tải xong và chạy DOS, bạn có thể vào bất cứ phân vùng mở rộng và phân vùng logic nào của hệ thống.

Nhưng trong khi Linux có thể truy cập các tệp DOS trong một phân vùng DOS thì trái lại DOS lại không thể truy cập các tệp Linux trong phân vùng Linux.

2.4.2.3 Các yêu cầu về Linux

Như đã nói, Linux thao tác trên các hệ thống tệp và chúng có thể trú trên nhiều phân vùng khác nhau, chủ yếu là để phòng xa. Linux đòi hỏi một phân vùng cho mỗi hệ thống tệp. Việc tiếp theo phải quan tâm là phân vùng hoán chuyển. Phần lớn các hệ điều hành đều cho phép tạo bộ nhớ ảo, Linux cũng lấy một phần ổ đĩa cứng làm tệp hoán chuyển hoặc phân vùng hoán chuyển để mô phỏng bộ nhớ vật lý. Kích thước phân vùng hoán chuyển tuỳ thuộc số lượng RAM vật lý của hệ thống máy. Một thông lệ được chấp nhận mặc nhiên là: phân vùng hoán chuyển lớn gấp đôi lượng RAM. Do đó nếu máy bạn có 8 MB RAM, phân vùng hoán chuyển phải là 16 MB. Nếu có từ 4 MB RAM trở xuống, bạn phải kích hoạt một phân vùng hoán chuyển.

Đối với RedHat Linux phiên bản 6.x trở lên, tổng dung lượng dành cho các phân vùng hoán chuyển trên Linux là tuỳ ý (tối thiểu là gấp đôi dung lượng bộ nhớ vật lý), do đó bạn chỉ cần tính đến các yêu cầu của những ứng dụng sẽ được cài trên máy để xác định dung lượng cần cho phân vùng hoán chuyển. Ví dụ như nếu bạn cần cài thêm ORACLE 7.x trên máy như một máy chủ cơ sở dữ liệu thì ít nhất phần phân vùng hoán chuyển phải có dung lượng là 500 MB. Vì thế nếu ngoài một phân vùng hoán chuyển mà hệ thống Linux của bạn lại cần hai phân vùng khác (một cho tệp hệ thống và một cho tệp người dùng) thì tổng cộng bạn phải phân 3 vùng cho Linux.

2.4.2.4 Phân vùng lại ổ DOS

Trước tiên bạn phải thi hành FDISK bằng cách gõ **fdisk** tại dấu nhắc DOS. Một menu gồm 4 tuỳ chọn FDISK sẽ hiện ra trên màn hình.

Qua các tuỳ chọn, bạn biết phân vùng nào hiện tồn tại, biết tạo phân vùng mới và xoá phân vùng cũ. Tuỳ vào phiên bản DOS mà bạn đang sử dụng, màn hình sẽ hơi khác nhau một chút.

Chọn "Display Partition Information" trên menu. Khi màn hình Display Partition Information xuất hiện, bạn nên chép lại các thông tin. Bạn sẽ cần những thông tin này nếu quyết định ngưng lại quá trình cài đặt Linux và phục hồi hệ thống nguyên thuỷ trên máy bạn.

Trong DOS 6.x, nếu muốn xem tất cả các thông tin về phân vùng hiện hữu, bạn cũng sử dụng tùy chọn Display Partition Information.

2.4.2.5 Cách tránh phân vùng đĩa cứng

Mặc dù việc phân vùng lại ổ đĩa cứng sẽ giúp Linux chạy tốt hơn, song không nhất thiết phải thực hiện như trên vì e mất dữ liệu. Có thể dùng FIPS để phân vùng mà không phá huỷ các thông tin trên ổ đĩa cứng.

FIPS (First non-destructive Interactive Partition Splitting) là một chương trình phát triển cho Linux. Như tên gọi, FIPS sẽ di chuyển các phân vùng DOS để dọn chỗ cho các phân vùng Linux mà không phá hỏng thông tin.

Muốn biết thêm, hãy tham khảo tệp fips.doc ở thư mục /utils/fips trong CD Linux. FIPS chỉ có ích khi nào ổ đĩa cứng trong máy bạn còn đủ khoảng trống cần thiết để cài đặt Linux, nếu không bạn phải xoá những tệp nào xét thấy không cần thiết.

Với bản phát hành Slackware Linux, bạn có thể cài đặt Linux trên cùng phân vùng với DOS (nhưng lúc này gọi là UMSDOS). UMSDOS là một dự án nhằm tạo điều kiện cho Linux hiện diện trên các phân vùng DOS. Nói cách khác, UMSDOS cho phép bạn tạo hệ thống tệp gốc của Linux trong một thư mục DOS đã có sẵn. Tuy nhiên bạn không thể sử dụng UMSDOS với RedHat Linux.

2.4.2.6 Xoá bỏ phân vùng

Rất tiếc là **fdisk** không cho phép bạn đặt lại kích thước của phân vùng một cách đơn giản. Trước tiên bạn phải xoá bỏ phân vùng ấy, sau đó lại tạo ra chính nó nhưng với kích thước mới. Từ màn hình tuỳ chọn **fdisk**, chọn tuỳ chọn 3, "Delete Partition" hoặc

"Logical DOS Drive" để xoá phân vùng được chọn. Màn hình Delete Partition hoặc Logical DOS Drive sẽ hiện ra.

Chọn tuỳ chọn tương ứng với loại phân vùng bạn sẽ muốn xoá, thí dụ phân vùng DOS sơ cấp. Chẳng hạn tuỳ chọn 1 (Delete Primary DOS Partition) giúp bạn xoá các phân vùng sơ cấp của DOS. Chọn tuỳ chọn 1 để hiển thị màn hình Delete Primary DOS Partition. Màn hình sẽ hỏi tên (volume) của phân vùng và đòi hỏi xác nhận lại lần nữa trước khi bạn quyết định xoá hẳn phân vùng ấy, cùng với tất cả mọi thông tin trên đó.

2.4.2.7 Thêm phân vùng mới

Sau khi xoá các phân vùng cần xóa, bạn phải thêm các phân vùng thích hợp cho hệ thống DOS của bạn bằng tuỳ chọn "Create a DOS Partition".

Hãy chọn các tuỳ chọn sau để được thấy hồi đáp màn hình cho "Create a DOS Partition" hoặc "Logical DOS Drive".

Ghi chú: Bạn không thể thêm phân vùng Linux vào bằng chương trình FDISK của DOS. Việc phân chia lại ổ đĩa cứng cho Linux sẽ được bàn sau ở mục "Sử dụng lệnh **fdisk** của Linux".

Các hồi đáp màn hình của **fdisk** bao gồm thông số khoảng trống cho phân vùng (tính bằng MB) và chỉ báo về phân vùng hiện hành (sáng rõ). Phân vùng hiện hành la phân vùng có thể khởi động được.

Để khởi động DOS, bạn phải chỉ định phân vùng sơ cấp là phân vùng hiện hành. Với sự chọn lựa đầu tiên như trên màn hình này, bạn nên chọn N (no) để sau đó có thể ấn định dung lượng dành cho phân vùng DOS.

Nếu bạn chọn "no", màn hình sẽ hiển thị "Specify Disk Space **for** the Partition Screen".

Bạn có thể chọn dung lượng dành cho phân vùng DOS bằng đơn vị MB hoặc bằng tỷ lệ phần trăm và bấm <Return>. Sau đó bạn chỉ định đấy là phân vùng hiện hành. Từ màn hình menu của **fdisk**, bạn chọn tuỳ chọn 2, "Set Active Partition", sau đó làm theo các hướng dẫn.

2.4.2.8 Định dạng phân vùng

Sau khi phân vùng lại ổ đĩa cứng, bạn phải chuẩn bị phân vùng mới cho DOS và phục hồi các tệp thích hợp vào phân vùng ấy. Dùng đĩa mồi đã tạo sẵn trước đó để khởi động lại máy. Tiếp theo bạn định dạng (format) ổ đĩa cứng thích hợp và chuyển các tệp hệ thống lên đó bằng lệnh DOS:

```
format c: /s
```

Khi phân vùng đã được định dạng xong, bạn có thể phục hồi phần sao lưu vào ổ mới. Nếu do giảm kích thước của phân vùng mà hết chỗ để phục hồi, bạn nên chuyển các tệp dư thừa vào các ổ DOS hoặc phân vùng DOS khác.

Chương 3. Cài đặt RedHat Linux

Chương này giúp bạn cài đặt bản phát hành RedHat Linux với những thông tin chi tiết trong các chủ đề sau đây:

-Các cách cài đặt
-Trình tự cài đặt
-Thiết lập cấu hình mạng
-Các thiết lập khác
-Tạo đĩa mềm khởi động
-Nâng cấp và gỡ bỏ RedHat
-Hỏi đáp

3.1 Các cách cài đặt

Như đã trình bày trong chương trước, có nhiều cách thực hiện việc cài đặt Linux: qua CD, qua đĩa cứng và qua mạng (FTP, NFS, SMB). Bảng 3.1 sau đây tóm tắt các điểm mạnh yếu của từng phương pháp trên.

Cài qua	Điểm mạnh	Điểm yếu
CD	Nhanh và ổn định	Bản phát hành sớm lỗi thời
FTP	Tiện lợi cập nhật, có thể truy cập từ mọi nơi trên thế giới	Không ổn định và chậm nếu máy chủ nằm xa máy trạm
NFS	Tiện lợi, đặc biệt trong trường hợp máy bạn không có ổ CD	Chậm và đòi hỏi phải có mạng UNIX
SMB	Tiện lợi trên mạng MS Windows	Phải qua mạng và phải có hiểu biết về Samba
Đĩa cứng	Khi tất cả các phương pháp kể trên đều không áp dụng được	Đòi hỏi nhiều dung lượng phụ trội

Bảng 3.1 : Tóm tắt các cách cài đặt Linux

Nếu định cài đặt cả giao diện X Window, bạn cần chuẩn bị trước các thông tin về loại chipset của bìa điều khiển màn hình, cũng như về cổng nối tiếp cho chuột và modem.

Ở đây chúng ta chỉ bàn đến việc cài đặt RedHat, bản phát hành phổ biến nhất của Linux. Mặc dù bạn có thể cài đặt bằng nhiều cách khác nhau như qua NFS, qua FTP, qua ảnh SMB trên một ổ đĩa cứng dùng chung, hoặc qua một ổ đĩa cứng khác, song cách phổ biến nhất là cài đặt RedHat từ CD.

Tóm lại, dù chọn phương pháp cài đặt nào, bạn vẫn phải nắm trước một số thông tin cần thiết về cấu hình vật lý và mục tiêu sử dụng như được nêu sau đây:

-Loại bìa điều khiển màn hình, chipset và màn hình
-Tên cổng nối tiếp của chuột
-Tên cổng nối tiếp của modem
-Địa chỉ IP, cổng gateway và tên miền (nếu nối máy với mạng)
-Loại ổ đĩa cứng, ổ CD và bìa điều khiển của chúng

-Tổ chức các thư mục mà bạn muốn có trên máy của mình
-Tên bạn định đặt cho máy của mình với hệ Linux (hostname).

Nếu có kết nối Internet, bạn có thể yêu cầu người quản trị mạng hoặc ISP (công ty cung cấp dịch vụ Internet) cho biết các thông tin kể trên.

Nếu dự định sử dụng nhiều hệ điều hành khác nhau trên cùng một máy (chẳng hạn như Window 95, Windows NT hoặc Windows 2000), bạn phải tạo ra phân vùng cho mỗi hệ điều hành ấy. Bạn phải sử dụng chương trình tạo phân vùng của chính hệ điều hành đó, bởi vì Linux không quản lý được mọi loại phân vùng của các hệ điều hành khác.

Sau cùng, bạn nên kiểm tra xem vào giờ chót bản phát hành RedHat có những chỉnh sửa nào không bởi vì Linux được nâng cấp thường xuyên, hoặc được thêm vào những tiện ích và sửa lỗi.

3.2 Trình tự cài đặt

Đặt đĩa boot mà bạn đã khởi tạo vào ổ đĩa mềm rồi khởi động lại máy. Sau khi hệ thống đã kiểm tra xong phần cứng và BIOS, các thông báo như sau sẽ hiện trên màn hình, bạn cũng sẽ nhận được màn hình này nếu bạn đưa CD RedHat vào và cho máy khởi động từ CD.

Minh hoạ 3.1 : Màn hình khi bắt đầu cài đặt

Nếu màn hình có dòng nhắc, thông thường bạn chỉ cần bấm phím <Enter> để tiếp tục cài đặt theo các tham số mặc định. Bảng 3.2 sau mô tả các phím chức năng có thể hỗ trợ khi bạn gặp khó khăn.

Phím	Mô tả
F1	Hiển thị màn hình khởi động (phím <F1> luôn hiển thị màn hình này)
F2	Hiển thị màn hình cung cấp thông tin tổng quát về tiến trình cài đặt
F3	Hiển thị màn hình giải thích chế độ chuyên gia (expert mode). Thông thường, tiến trình cài đặt Linux tự động kiểm tra phần cứng và có thể làm treo máy, khi đó cần phải vào chế độ chuyên gia để xác định từng phần cứng hệ thống
F4	Hiển thị màn hình cung cấp thông tin về việc khởi tạo và sử dụng đĩa cấp cứu
F5	Hiển thị màn hình trợ giúp. Từ ver. 5.1 trở đi, nhờ ở tệp cấu hình việc cài đặt không cần trợ giúp này nữa
F6	Hiển thị màn hình khi cài đặt không suôn sẻ và cần phải khai báo thêm vài tham số cho kernel khởi động

Bảng 3.2 : Các phím chức năng

Màn hình hiển thị các thông báo như sau, rồi bắt đầu nạp hệ thống:

```
Loading initrd.img...
Loading vmlinuz...
```

Minh hoạ 3.2 : Màn hình kiểm tra hệ thống

Chú ý dòng thông báo ở cuối màn hình trên: *Unable to probe*. Nó báo cho biết Linux không hiểu bìa điều khiển màn hình và màn hình của bạn. Bạn cần phải cấu hình lại bằng tay.

Sau khi khởi động xong (minh hoạ trên), hệ thống của bạn sẽ hiển thị màn hình đón chào (Welcome) như ở minh hoạ sau.

Lưu ý rằng nếu sử dụng chuột qua cổng USB, bạn nên di chuyển chuột trong khi chờ hiện ra màn hình Welcome để chương trình cài đặt có thể phát hiện ra loại chuột đó.

Minh hoạ 3.3 : Màn hình đón chào 1

3.2.1 Cấu hình hệ thống

Bấm phím <Enter> để tiếp tục.

Minh hoạ 3.4 : Màn hình chọn ngôn ngữ cài đặt

Màn hình hỏi bạn chọn ngôn ngữ nào (Anh, Pháp v.v. hay Việt ?) trong khi cài đặt. Ngôn ngữ được chọn sẽ là ngôn ngữ mặc định cho hệ thống của bạn sauu khi cài đặt.

Ghi chú: Bạn có thể sử dụng phím để di chuyển trong hộp thoại như chương trình có nhắc nhở ở dòng cuối màn hình. Muốn di chuyển từ trường này sang trường kia, hãy bấm phím <Tab> hoặc <Alt-Tab>. Dùng Spacebar (phím ký tự trống) để chọn lựa danh sách hoặc để đánh dấu vào ô lựa chọn. Sau khi chọn ô OK hay Cancel, hãy bấm phím <Enter>. Muốn di chuyển qua danh sách liệt kê, hãy dùng phím mũi tên.

Hộp thoại như ở minh hoạ dưới yêu cầu bạn chọn loại bàn phím cho hệ thống. Sau này khi muốn thay đổi lựa chọn về loại bàn phím, hãy dùng */usr/sbin/kbdconfig* (khi bạn đóng vai trò là *superuser* hoặc *root*).

Minh hoạ 3.5 : Màn hình chọn bàn phím

Hộp thoại kế tiếp yêu cầu bạn khai báo loại chuột đang dùng. Từ phiên bản RedHat 7.x, bạn đã có sự hỗ trợ cho loại chuột dùng cổng USB và cả loại chuột dùng cổng hồng ngoại (Infrared), còn gọi là loại chuột không dây (wireless). Cần chú ý kiểm tra cổng kết nối của chuột với máy tính xem đó là cổng COM (Serial), USB hay PS/2. Hãy nhớ rằng cổng PS/2 có dạng hình tròn, cổng USB có dạng hình chữ nhật và cổng COM có dạng hình thang. Nếu chuột của bạn kết nối bằng cổng COM, bạn phải chú ý xem nó kết nối qua cổng số mấy: COM1 hay COM2.

Minh hoạ 3.6 : Màn hình chọn chuột

Bạn cũng cần nghĩ xem có nên sử dụng tính chất phỏng tạo chuột 3 phím hay không, nếu bạn chỉ có chuột 2 phím. Mặc định là không nhưng bạn nên chọn có, điều này sẽ hỗ trợ cho bạn khi chạy trong môi trường X Window, khi đó có thể bấm cùng lúc cả 2 phím trái-phải của chuột để thay cho phím giữa, giống như có chuột 3 phím.

Màn hình đón chào thứ hai sẽ hướng dẫn bạn thêm về trình tự cài đặt cũng như đăng ký với RedHat để nhận sự giúp đỡ trực tuyến. Việc này đòi hỏi bạn phải mua chương trình cài đặt từ RedHat, tức là không dành cho các bản tải nạp miễn phí.

Minh hoạ 3.7 : Màn hình đón chào 2

Chú ý rằng các màn hình của RedHat luôn có phần trợ giúp nằm ở phía bên trái. Bạn có thể tắt chúng bằng cách nhấp chuột vào ô "Hide Help".

3.2.2 Tuỳ chọn cài đặt

Minh hoạ 3.8 : Màn hình "Tuỳ chọn cài đặt"

Màn hình "Tuỳ chọn cài đặt" xác định việc cài đặt là để máy bạn sẽ đóng vai trò gì. Bạn có các tuỳ chọn sau đây:

- **Workstation**: hãy chọn vai trò này khi bạn mới làm quen với Linux, muốn sử dụng máy của bạn một cách độc lập tại nhà hay xem nó chỉ như một trạm làm việc (Workstation) trong một mạng có sẵn. Bạn có thể gặp rắc rối nho nhỏ nếu cài đặt trên máy đã có dữ liệu (Windows 9x hay NT, 2000...).

- **Server**: có thể chọn vai trò này nếu bạn muốn thực hành như một người quản trị mạng đang tự xây dựng máy chủ (Server). Chú ý rằng với tuỳ chọn này tất cả dữ liệu đang có trên đĩa của bạn sẽ bị xoá sạch.

- **Laptop**: tương tự như Workstation, nhưng vai trò của tuỳ chọn này chủ yếu là để tối ưu hóa cho việc cài đặt một máy tính xách tay (laptop).

- **Custom**: có thể chọn vai trò cài đặt này khi bạn muốn cài đủ thứ trên máy của mình. Hãy sử dụng vai trò này khi bạn rất quen thuộc Linux và khi bạn cần một sự linh động tối đa trong việc cấu hình máy theo Linux.

- **Upgrade**: việc nâng cấp hệ thống hiện hành sẽ dễ dàng hơn khi bạn chọn vai trò này. Nó sẽ giữ lại dữ liệu và các cài đặt cũ của bạn, chỉ cập nhật các gói phần mềm và kernel của Linux một cách nhanh chóng và an toàn. Các cách cài đặt khác không đảm bảo việc bảo lưu dữ liệu cho bạn.

Xem "Nâng cấp các gói phần mềm với RPM".

Ghi chú: Muốn xem chương trình cài đặt đang thực hiện thao tác nào, bạn bấm <Alt-F3> để chuyển sang terminal ảo.

3.2.3 Phân vùng đĩa cứng

Tiếp theo bạn phải phân vùng ổ đĩa cứng, hoặc chí ít cũng phải chọn lựa các phân vùng trước đó bạn đã tạo ra. Chương trình cài đặt sẽ hiển thị hộp thoại Disk Setup như ở hình dưới.

Bạn có thể sử dụng chương trình **fdisk** hoặc chương trình Disk Druid hoặc bạn cho Linux tự động phân vùng cho bạn. Nếu sử dụng **fdisk**, bạn di chuyển đến dòng có chữ **fdisk** và bấm <Next>. Máy sẽ chuyển sang chương trình **fdisk** để phân vùng ổ đĩa cứng mà bạn đã chọn.

Muốn dùng chương trình Disk Druid thì bạn bỏ qua dòng **fdisk** để đến dòng "Using Disk Druid" (Sử dụng Disk Druid).

Minh hoạ 3.9 : Màn hình "Phân vùng đĩa"

Ngoài ra bạn cũng có thể để cho chương trình cài đặt tự phân vùng ổ đĩa cứng khi chọn tuỳ chọn đầu tiên.

3.2.3.1 Sử dụng fdisk của Linux

Khi dùng chương trình **fdisk** như một chuyên gia, trước hết bạn sẽ phải chọn ổ đĩa cần phân vùng (máy PC thông thường có 1 ổ đĩa cứng như thí dụ minh hoạ dưới đây).

Lưu ý: Bạn nhớ dùng chương trình **fdisk** của chính Linux và nên cẩn thận bởi vì nó khác với những **fdisk** của các hệ điều hành khác, chẳng hạn như DOS, Windows 98, OS/2 hoặc Windows 2000. Bạn không thể sử dụng lẫn lộn những chương trình đó, thí dụ không thể dùng **fdisk** của Linux để sắp xếp một phân vùng do DOS tạo ra.

*Minh hoạ 3.10 : Màn hình " Phân vùng bằng **fdisk**"*

Bạn có thể gõ lệnh ***m*** để liệt kê danh sách các lệnh của chương trình **fdisk** :

Lệnh	Command action (tiếng Anh)	Ý nghĩa
a	toggle a bootable flag	Bật, tắt cờ ***bootable*** (khởi động được)
c	toggle a DOS-compatible flag	Bật, tắt cờ tương thích DOS
d	detete a partition	Xoá một phân vùng
l	list known types	Liệt kê các kiểu phân vùng đã biết
m	print this menu	Hiển thị bảng này
n	add a new partition	Bổ sung một phân vùng
p	print the partition table	Hiển thị bảng phân vùng
q	quit without saving changes	Ra khỏi mà không ghi lại sự thay đổi
t	change a partition's system id	Thay đổi một ***system id*** của phân vùng
u	change display/entry units	Thay đổi các đơn vị hiển thị/nhập
v	verify the partition table	Kiểm tra bảng phân vùng
w	write table to disk and **exit**	Ghi bảng phân vùng vào đĩa và ra khỏi
x	extra functionality (expert only)	Chức năng phụ (dành cho chuyên gia)

*Bảng 3.3: Danh sách các lệnh của chương trình **fdisk***

Khi bắt đầu phân vùng, bạn nên gõ lệnh p (bấm <p> <Enter>) để hiển thị bảng phân vùng của ổ đĩa cứng đã được thực hiện trước đó bằng chương trình **fdisk** của DOS.

Thí dụ dưới đây cho thấy màn hình sau khi thực hiện lệnh p hiển thị gì :

```
The number of cylinders for this disk is set to 2482
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 2482 cylinders
Unit = cylinders of 16065 * 512 bytes
```

```
Device Boot      Start     End      Blocks   Id  System
/dev/hda1           1         6       48163+  83  Linux
/dev/hda2           7       159     1228972+  82  Linux swap
/dev/hda3         160      2482    18659497+  83  Linux
```

Minh hoạ 3.11.: Màn hình hướng dẫn của lệnh FDISK

Khung trên cho thấy các phân vùng đã được xác định ở đĩa nào, cùng với vị trí khởi đầu và kết thúc của phân vùng, kích cỡ của các block và các loại phân vùng. Bảng 3.4 sau đây tóm tắt những loại phân vùng quen biết của Linux:

Loại phân vùng	Giá trị	Loại phân vùng	Giá trị
Empty	00	Novell Netware 386	65
DOS 12-bit FAT	01	PIC/IX	75
XENIX root	02	Old MINIX	80
XENIX usr	03	Linux/MINUX	81
DOS 16-bit <=32M	04	Linux swap	82
Extended	05	Linux native	83
DOS 16-bit >=32	06	Linux extended	85
OS/2 HPFS	07	Amoeba	93
AIX	08	Amoeba BBT	94
AIX bootable	09	BSD/386	a5
OS/2 Boot Manager	0a	Open BSD	a6
Win95 FAT32	0b	NEXTSTEP	a7
Win95 FAT32(LBA)	0c	BSDI fs	b7
Win95 FAT16(LBA)	0e	BSDI swap	b8
Win95 Extended (LBA)	0f	Syrinx	c7
Venix 80286	40	CP/M	db
Novell	51	DOS access	e1
Microport	52	DOS R/O	e3
GNU HURD	63	DOS secondary	f2
Novell Netware 286	64	BBT	ff

Bảng 3.4 : Các loại phân vùng đã biết của Linux

Bảng trên trình bày tất cả các loại phân vùng có thể định nghĩa bằng chương trình **fdisk** của Linux, trong đó những loại phân vùng sơ cấp thường dùng là Linux Native và Linux Swap. Nếu dùng lệnh l của **fdisk** thì bạn cũng sẽ thấy kết quả tương tự như thế. Vì không cần phân vùng lại ổ đĩa cứng cho DOS cho nên bạn không phải xoá bỏ bất kỳ phân vùng nào của Linux. Bạn chỉ phải thêm phân vùng vào. Trong thí dụ sau,

chúng ta cùng tạo lại phân vùng như kết quả trên. Giả sử đĩa cứng của bạn đã tạo được phân vùng thứ 1 (hda1), bạn tạo thêm phân vùng 2 mới bằng cách gõ lệnh n như sau:

```
Command Action : n
e extended
p primary (1-4)
```

Bấm <p> <Enter>, **fdisk** sẽ hỏi bạn số phân vùng. Bạn gõ vào theo ý mình và bấm <Enter>. Nếu bạn cho một số phân vùng đã được dùng, **fdisk** sẽ thông báo việc này và hỏi bạn xem có xoá phân vùng ấy không, trước khi thử thêm nó vào bảng phân vùng. Với thí dụ này, bạn nhập số 2 để thêm vào một phân vùng sơ cấp thứ hai, mang dòng tham khảo là /dev/hda2.

```
Partition number (1-4):2
First cylinder (7-2482, default 7):
Using default value 7
Last cylinder or +size or +sizeM or +sizeK (7-2482, default 2482): 159
```

Tiếp theo, máy sẽ hỏi vị trí của cylinder (trụ đĩa) đầu tiên là gì. Thông thường đó là cylinder đầu tiên có sẵn. Trên thực tế **fdisk** hiển thị một giá trị mặc định cho bạn chọn, trong thí dụ sau giá trị mặc định là 7:

```
First cylinder (7-2482, default 7):
```

Bạn thấy rằng phân vùng đầu tiên kết thúc ở cylinder 6 và đĩa có cylinder cuối cùng là 2482.Vì thế khoảng trống do **fdisk** cung cấp cho phép bạn khởi đầu phân vùng kế tiếp bất kỳ ở điểm nào trong tầm 7-2482. Việc không đặt phân vùng ở bất kỳ chỗ nào trong ổ đĩa cứng là một ý hay. Bạn nên chọn điểm kế tiếp nào còn trống. Trong trường hợp này đó là cylinder 7. Gõ 7 và bấm <Enter>.

Ghi chú: Nếu khởi động Linux từ các phân vùng khởi đầu ở các cylinder cao hơn 1024 thì có thể bị rắc rối. Trong trường hợp bất đắc dĩ phải tạo phân vùng ở những vị trí cao hơn cylinder 1024, có thể bạn phải khởi động Linux từ đĩa mềm. Cuối chương này sẽ có hướng dẫn bạn tạo đĩa mềm khởi động. Khởi động Linux từ đĩa mềm sẽ lâu hơn khởi động từ ổ đĩa cứng.

Đến đây **fdisk** muốn bạn chỉ định dung lượng cho phân vùng mới. Bạn có thể chỉ định kích thước (size) bằng số cylinder hoặc bằng số byte (+size), số kilobyte (+sizeK), hoặc megabyte (+sizeM).

Bởi vì bạn đã biết kích thước gần đúng của tệp hoán chuyển, cho nên bạn hãy chỉ định phân vùng này trước, sau đó chừa phần còn lại của ổ đĩa cứng cho các phân vùng chương trình Linux. Ở thí dụ này, máy của bạn có 8 MB RAM, do đó bạn chỉ định phân vùng 16 MB bằng cách trả lời như sau:

```
Last cylinder or +size or +sizeM or +sizeK (42-1023): 159
```

Sau đó bạn dùng lệnh p để xem bảng phân vùng mà bạn vừa chỉ định. Ở thí dụ này, bảng phân vùng mới sẽ có dạng như sau:

```
root@mail linux-2.2.12]#/sbin/fdisk/dev/hda
The number of cylinders for this disk is set to 2482. There is nothing
wrong with that, but this is larger than 1024,
```

```
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., LILO)
2) booting and Partitioning software from other OSs (e.g., DOS FDISK, OS/2 FDISK)
Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 2482 cylinders
Units = cylinders of 16065*512 bytes
Device Boot  Start   End      Blocks       Id      System
/dev/hda1    1       6        48163+       83      Linux
/dev/hda2    7       2482     1228972+     83      Linux
```

Theo mặc định, **fdisk** sẽ định dạng phân vùng mới như là loại Linux Native. Muốn đổi nó thành phân vùng hoán chuyển (swap partition), bạn dùng lệnh t. Gõ t, sau đó gõ số phân vùng bạn muốn đổi (gõ số 2 trong thí dụ này). Chương trình **fdisk** sẽ yêu cầu bạn gõ giá trị thập lục phân của loại phân vùng theo bảng 2.7 (nếu không có sẵn bảng ấy, bạn chỉ gõ 1 và có ngay danh sách mã). Ở đây, bởi vì bạn muốn có một phân vùng hoán chuyển nên cần gõ số 82 tại dấu nhắc.

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes) : 82
Changed system type of Partition 2 to 82 (Linux swap)
```

Như bạn thấy, **fdisk** thông báo loại phân vùng mới, tuy nhiên bạn vẫn có thể dùng lệnh p để kiểm tra lại lần nữa xem giờ đây phân vùng 2 có phải là phân vùng hoán chuyển của Linux nay không.

Giờ bạn có thể thêm vào các phân vùng Linux của mình. Ở thí dụ này bạn chỉ cần thêm vào một phân vùng, song nếu muốn có nhiều hơn thì cứ thêm vào. Muốn thêm, bạn bấm <n>, chỉ định p cho một phân vùng sơ cấp khác và sau đó chỉ định số hiệu cho phân vùng ấy (trong thí dụ là 4). Để tránh tình trạng rải rác nhiều phân vùng khắp ổ đĩa cứng, bạn nên khởi đầu phân vùng sau cùng tại địa điểm mà phân vùng áp chót vừa kết thúc (trong thí dụ là cylinder 160). Đối với cylinder cuối cùng, vì bạn muốn sử dụng dung lượng còn lại cho hệ Linux, cho nên bạn có thể chỉ định cylinder cuối thay vì phải nói chính xác là bao nhiêu byte. Do đó bạn gõ 2482 như sau:

```
Command (m for help) : n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (160-2482, default 160):
Using default value 160
Last cylinder or +size or +sizeM or +sizeK (160-2482, default 2482):
Using default value 2482
```

Đến đây bạn nên gõ lệnh p để kiểm tra phân vùng mới. Nếu muốn thay đổi tiếp thì hãy thực hiện cẩn thận như trên.

Trong thí dụ về phân vùng trên ta thấy rằng khi khai báo là primary partition, số lượng tối đa chỉ là 4. Còn nếu khai báo là extended, ta có thể có số lượng phân vùng là tuỳ ý.

Khi đã hài lòng với tổng thể các phân vùng, bạn dùng lệnh w để ghi thông tin bảng phân vùng vào ổ đĩa cứng. Một khi chưa sử dụng lệnh w thì những thay đổi sẽ không được lưu trữ. Vì vậy trong khi thao tác nếu cảm thấy đã lỡ thay đổi điều gì đó không đúng ý, bạn dùng lệnh q để thoát ra mà không thay đổi gì trong bảng phân vùng. Nếu gõ w, Linux sẽ báo cho bạn biết rằng bảng phân vùng đã được đổi, kế tiếp Linux cập nhật các đĩa cho khớp với bảng phân vùng mới. Nếu làm đến đây mà máy bị treo, hãy khởi động lại bằng đĩa cài đặt và đĩa root cho đến khi bạn gặp dấu nhắc # trở lại.

Cẩn thận: Đừng dùng chương trình **fdisk** của Linux để tạo ra hoặc chỉnh sửa các phân vùng của những hệ điều hành khác. Điều này có khả năng làm cho cả hai hệ điều hành đều không sử dụng được ổ đĩa cứng nữa.

Tạo phân vùng hoán chuyển: Một vài bản phát hành Linux có cho phép việc tự động tạo ra và kích hoạt tệp swap trong tiến trình cài đặt. Song ở một vài bản phát hành khác, người sử dụng phải tạo và kích hoạt tệp ấy trước khi tiếp tục tiến trình cài đặt.

Ghi chú: Nếu trong tiến trình cài đặt bạn bị báo lỗi "out of memory" thì bạn phải tăng thêm dung lượng tệp swap.

Với RedHat 6.x trở lên, ta có thể bỏ qua mục này vì RedHat đã tự kích hoạt phân vùng hoán chuyển thay cho bạn.

Muốn tạo ra khoảng swap, bạn dùng lệnh **mkswap** sau đó cho máy biết dùng phân vùng nào và dung lượng cho RAM ảo là bao nhiêu. Thí dụ muốn tạo khoảng swap trên phân vùng/dev/hda3, tại dấu nhắc # bạn gõ lệnh như sau:

```
#mkswap -c /dev/hda3 16447
```

Số 16447 có nghĩa là 16 MB vào có thể nhìn thấy ở cột block trên màn hình khi gõ chương trình **fdisk**. Tuỳ chọn –c báo cho **mkswap** biết để kiểm tra những bad section (đoạn section bị hỏng) trên phân vùng. Sau đó bạn kích hoạt hệ thống swap bằng lệnh swapon:

```
#swapon /dev/hda3
```

Nhắc lại lần nữa rằng nếu đang sử dụng CD RedHat Linux để cài đặt, bạn không phải lo lắng gì về việc kích hoạt hệ thống swap khi bạn tạo ra phân vùng cho một người dùng. Trong khi cài đặt, chương trình sẽ phát hiện phân vùng hoán chuyển và tự động khởi động hệ thống để cài đặt.

Sau khi tạo các phân vùng trên những ổ đĩa cứng khác nhau và trở về hộp thoại phân vùng đĩa, bạn bấm ô chữ "Done" để tiếp tục tiến trình cài đặt.

Tạo khoảng swap xong, chương trình sẽ hiển thị hộp thoại Select Root Partition. Root Partition (phân vùng gốc) là hệ thống tệp chính của máy bạn dành cho Linux, nơi mà tất cả các tệp boot được lưu trữ. Từ hộp liệt kê danh sách, bạn hãy chọn tên ổ đĩa cứng để làm Root Partition và bấm <Enter>. Từ hộp thoại phân vùng đĩa cứng bạn có thể thiết lập các phân vùng khác.

Từ đây bạn có thể lập bất kỳ hệ thống tệp DOS hoặc OS/2 nào để sau đó có thể truy cập bằng Linux. Từ hộp liệt kê danh sách, bạn chọn phân vùng nào cần chỉnh sửa sau đó bấm <Enter>. Từ hộp thoại Edit Mount Point bạn xác định một điểm lắp ghép (**mount point**), nghĩa là một thư mục, nơi bạn muốn lập phân vùng ấy.

Sau khi bạn chọn xong **mount point** và root (điểm gốc) cho các phân vùng, chương trình sẽ tạo định dạng (format). Muốn xác định phân vùng nào cần format, bạn chọn tại hộp thoại Format Partition.

Xem "Tháo lắp các hệ thống tệp"

3.2.3.2 Sử dụng Disk Druid

Minh hoạ 3.12: Phân vùng đĩa bằng DiskDruid

Bấm ô Disk Druid ở hộp thoại Disk Setup, bạn sẽ thấy màn hình phân vùng đĩa như ở minh hoạ 3.12. Disk Druid giúp bạn tạo ra phân vùng, thiết lập **mount point**, kích thước phân vùng, loại hệ thống tệp và biết được thông tin về thuộc tính. Dưới đây sẽ liệt kê các ô bấm khác nhau trên màn hình chính của Disk Druid, cùng với chức năng của từng ô. Các ô bấm của chương trình Disk Druid cho phép bạn tiến hành phân vùng một cách trực quan:

- **New**: Dùng để tạo một phân vùng mới. Một hộp thoại sẽ xuất hiện như minh hoạ.

- **Edit**: Dùng để điều chỉnh thuộc tính của phân vùng được chọn (phân vùng được chọn là phân vùng được đánh dấu trong vùng Partition trên màn hình). Hộp thoại chỉnh sửa sẽ xuất hiện, cấu trúc của nó cũng giống như của hộp thoại Tạo mới.

Bạn cũng có thể tạo một phân vùng mới bằng cách Edit vùng "Free Space" của vùng Partition.

- **Delete**: Dùng để xoá phân vùng được đánh dấu trong vùng "Curent Disk Partitions".

- **Reset**: Dùng để hoàn lại tình trạng gốc trước khi sửa đổi.

- **Make RAID**: Có thể dùng để cung cấp dịch vụ RAID cho bất kỳ phân vùng nào.

Chúng tôi hy vọng là bạn không sử dụng chức năng RAID, trước khi hiểu rõ nó thông qua tài liệu "Official RedHat Linux Customization Guide" hay tương tự. Chức năng này hỗ trợ trong chừng mực nào đó việc bảo đảm tính toàn vẹn và bền vững của dữ liệu trên đĩa, tránh những hư hỏng có thể xảy ra.

Qua hộp thoại tạo phân vùng mới, bạn nhập một giá trị tương ứng vào trường **Mount Point**, chẳng hạn như giá trị phân vùng gốc (/) hoặc phân vùng var (/var). Sau đó bạn cho biết dung lượng của phân vùng tính bằng đơn vị megabyte, đồng thời xác định xem có muốn sau này phân vùng ấy có thể phình ra theo yêu cầu hay không (phân vùng có thể phình ra khi bạn thêm bớt những phân vùng khác).

Minh hoạ 3.13: Tạo phân vùng mới

Tiếp theo bạn chọn từ hộp liệt kê " FileSystem Type" để quyết định xem loại hệ thống tệp nào sẽ hiện diện trên phân vùng ấy. Cuối cùng, từ danh sách " Allowable Drive" bạn chọn xem ổ đĩa cứng vật lý nào sẽ chứa phân vùng.

Ghi chú: Nếu bạn chỉ định một kích cỡ lớn hơn khoảng trống còn lại trên ổ đĩa cứng đã chọn, Disk Druid sẽ báo lỗi và đề nghị bạn giảm dung lượng đăng ký. Disk Druid cũng báo cho bạn biết những vấn đề có thể xảy ra, đồng thời cung cấp các giải pháp khả dĩ.

3.2.3.3 Phân vùng tự động

Khi bạn mới làm quen với Linux, bạn có thể để Linux tự động phân vùng đĩa cứng (minh hoạ 3.14). Bạn có thể chọn một trong 3 tuỳ chọn:

- Remove all Linux Partition on this system - chỉ xoá những phân vùng Linux cũ, không xoá các phân vùng khác.

- Remove all Partition on this system – xoá tất cả các phân vùng trên các đĩa cứng, kể cả các phân vùng tạo bởi hệ điều hành khác (như Windows 95/98/NT/2000). ***Chú ý:*** Tuỳ chọn này cho phép chương trình cài đặt xoá hết mọi dữ liệu bạn đang có trên đĩa.

- Keep all Partition and use existing free space - bạn cần giữ lại các dữ liệu hiện hành của bạn. Tuy nhiên, bạn cũng cần đủ không gian trống cần thiết cho việc cài đặt mới.

Minh hoạ 3.14: Phân vùng tự động

3.2.4 Cài đặt chương trình khởi động

Bạn cần chọn một chương trình khởi động khi muốn khởi động hệ Linux của bạn mà không cần đĩa mềm. Bạn có thể chọn GRUB (Grand Unified Bootloader), một chương trình khởi động mới của RedHat, hoặc phổ biến nhất là LILO (Linux Loader), chương trình khởi động chuẩn của Linux (minh hoạ 3.15).

Trường hợp bạn đã có một hệ điều hành khác trên máy, bạn có thể không cần cài đặt trình khởi động nào cả, nhưng thay vào đó, bạn không được quên việc tạo đĩa khởi động (boot disk) ở cuối chương trình cài đặt.

Việc xác định nơi cài đặt trình khởi động rất quan trọng: nếu như bạn chỉ định sai, rất có thể hệ điều hành đang hoạt động sẽ bị "biến mất" hay máy tính của bạn sẽ bị treo và việc chỉnh sửa trở lại khá phức tạp.

Bạn có thể cài đặt vào MBR (Master Boot Record), một vùng đặc biệt và duy nhất trên máy của bạn, được BIOS nạp vào tự động và là điểm đầu tiên mà trình khởi động tác động đến trong quá trình khởi động. Nếu bạn cài đặt trình GRUB hay LILO vào MRB, chúng sẽ hiện dấu nhắc khởi động. Tuỳ theo cách cấu hình sau này, bạn có thể sử dụng chúng để nạp các hệ điều hành khác nhau.

Minh hoạ 3.15: Các chương trình khởi động

Nếu hệ điều hành đang có trên máy bạn cũng đang tác động đến MRB như OS/2, Window 2000... bạn có thể chỉ cần cài GRUB hay LILO vào vùng "First sector of Root Partition". Khi đó, chương trình khởi động của hệ điều hành hiện hành sẽ quản lý việc khởi động chung. Bạn sẽ phải định lại cấu hình cho chương trình khởi động ấy để có thể nạp GRUB hay LILO khi bạn muốn khởi động hệ Linux.

Nếu hệ điều hành trên máy bạn chỉ là Linux, hay Windows 9x, bạn nên cài đặt LILO hay GRUB vào MRB.

Chú ý: Bạn không cần lo khi lần đầu khởi động mà không thấy xuất hiện mục khởi động Windows XP hay Windows 2000 trên trình khởi động GRUB. Các phân vùng của Windows vẫn tồn tại. Nhưng bạn phải bổ sung tệp boot (sẽ mô tả thêm trong phần sau).

Bạn có thể thêm các tham số cho việc khởi động từ LILO hay GRUB bằng cách thêm vào ô "Kernel parameters". Chúng ta sẽ trở lại vấn đề này ở các chương sau.

Tuỳ chọn "Force use of LBA32" cho phép bạn vượt qua giới hạn 1024 cylinder của phân vùng "boot". Nếu hệ thống của bạn hỗ trợ LBA32 mà trình cài đặt không tự phát hiện được phần mở rộng này của BIOS, bạn nên đánh dấu vào tuỳ chọn này.

3.2.4.1 Các chương trình khởi động khác

Ta có thể dùng đĩa mềm khởi động, Loadlin, Syslinux và các phần mềm thương mại khác như System Commander, Partition Magic, Boot Start... Việc sử dụng Loadlin khá phức tạp, đòi hỏi có một bản sao của phần Linux kernel trên một DOS Partition. Ta sẽ khởi động bằng một chương trình khởi động nào đó rồi sao chép phần kernel sang một DOS Partition.

3.2.4.2 Mật khẩu cho GRUB

Nếu không chọn GRUB hay LILO, bạn có thể chuyển đến mục 3.3 " Thiết lập cấu hình mạng" ở dưới đây (minh hoạ 3.17).

Bạn cần cung cấp mật khẩu cho chương trình khởi động GRUB để ngăn ngừa các vụ xâm phạm tính bảo mật từ phía người dùng.

Minh hoạ 3.16: Thiết lập mật khẩu khởi động

3.3 Thiết lập cấu hình mạng

Chương trình cài đặt tiếp tục với việc thiết lập cấu hình mạng cho máy của bạn. Nếu có ý định kết nối với Internet, cài đặt các thành phần làm việc trên mạng và có một bìa giao tiếp mạng (NIC) thì bạn có thể sẽ gặp một màn hình như minh hoạ 3.17.

Nếu máy có nhiều NIC, bạn sẽ nhận được nhiều tab trong màn hình trên. Linux có thể phát hiện được nhiều loại NIC khác nhau. Còn nếu không có một NIC nào, chương trình cài đặt sẽ bỏ qua và đi tiếp đến mục "Firewall Configuration" (xem mục 3.3.2).

Nếu nhận diện được bìa mạng trong máy bạn, chương trình cài đặt sẽ hướng dẫn bạn thiết lập mạng TCP/IP (minh hoạ 3.17).

Lưu ý: Ethernet là giao diện mạng thông dụng nhất cho Linux hiện nay. Các công nghệ khác như Token Ring, ISDN hoặc ATM có thể dùng được, song chưa hoàn toàn thích hợp cho Linux. Nhiều chương trinh điều khiển các thiết bị dùng những công nghệ vừa kể trên có thể đang còn ở giai đoạn thử nghiệm trong môi trường Linux và rất tuỳ thuộc vào phần cứng của người sản xuất gốc (OEM).

3.3.1 Thiết lập cấu hình mạng TCP/IP

Bạn khai báo thông tin về TCP/IP qua hộp thoại Network Configuration. Quản trị viên mạng hoặc nhà cung cấp dịch vụ Internet sẽ cung cấp cho bạn các thông tin sau đây:

IP number, netmask, network address và broadcast address.

Minh hoạ 3.17: Thiết lập cấu hình mạng

Sau đó chương trình sẽ thiết lập cấu hình mạng máy bạn. Bạn khai báo vào hộp thoại Network Configuration các thông tin như tên miền và host name của hệ thống. Tên miền là hai phần cuối của địa chỉ Internet. Thí dụ nếu địa chỉ là www.citynet.com, thì citynet.com là tên miền, còn www là host name. Tuy nhiên ở đây bạn cần ghi tên đầy đủ (FQDN=Fully qualified Domain Name) vào ô "hostname".

Tiếp theo quản trị viên mạng sẽ cho bạn giá trị của cổng kết nối (Gateway) mặc định và máy chủ tên miền chính (Primary Name Server). Nếu mạng của bạn có nhiều máy chủ tên miền, bạn hãy điền giá trị của các server này vào nơi thích hợp.

Ghi chú: Nên thận trọng khi đặt tên host, bởi vì tên này sẽ xuất hiện ở câu nhắc mặc định, ở các tin nhắn và ở các báo cáo ký sự (log).

Ngay cả khi máy bạn không thực sự nối vào mạng, bạn cũng có thể đặt tên cho máy của mình. Còn không thì máy của bạn sẽ được coi như có tên là localhost.

3.3.2 Cấu hình bức tường lửa

Việc tích hợp bức tường lửa là một tính chất mới của các phiên bản RedHat 7.x và cho phép ta bảo vệ hệ thống (xem minh hoạ 3.18). Bạn có thể chọn các mức bảo mật sau đây tuỳ theo yêu cầu của mình:

- **High**: hệ thống bạn sẽ không chấp nhận các kết nối mà bạn chưa khai báo tường minh. Một cách mặc định, chỉ có các kết nối hồi âm DNS (DNS replies) và DHCP là được cho phép (tất nhiên các kết nối sau đây sẽ không được phép: FTP, IRC, RealAudio™, Remote X Window client, v.v.).

- **Medium**: bức tường lửa của bạn sẽ cấm các cuộc truy cập từ xa đến tài nguyên trên máy bạn. Một cách mặc định, mọi kết nối đến các cổng TCP có số hiệu nhỏ hơn 1023 và các cổng NFS server (2049), cổng X Font server, v.v. đều sẽ bị cấm.

- **No Firewall**: chế độ bảo mật không được đặt ra, bất cứ kết nối nào đến máy bạn cũng sẽ không bị cấm. Bạn chỉ nên sử dụng tuỳ chọn này khi bạn đang ở trong một mạng đáng tin cậy. Tuy nhiên nếu bạn mới tìm hiểu về Linux, bạn nên chọn chế độ này để có thể sử dụng ngay một số dịch vụ. Sau đó ta sẽ xem xét lại chế độ Bức tường lửa.

- **Customize**: tuỳ chọn này cho phép bạn chủ động hơn trong việc xác định các thiết bị, kết nối, dịch vụ nào là đáng tin cậy. Thí dụ: bạn nối vào mạng riêng qua NIC (Ethernet bìa – eth0) và dùng modem qua giao thức PPP để kết nối với mạng Internet (ppp0). Khi đó, bạn có thể xem các kết nối qua eth0 là đáng tin (Trusted devices = eth0) còn giao tiếp pp0 là cần kiểm soát.

Bạn cũng có thể thông qua tuỳ chọn "Allow Incoming" để quyết định có nên giới hạn hay không giới hạn các dịch vụ.

Chú ý rằng nếu ở RedHat 6.x, sau khi cài tự động, bạn có thể sử dụng ngay máy của mình làm một máy chủ Mail, Web,... Còn ở RedHat 7.x, do có Bức tường lửa cho nên các kết nối từ bên ngoài sẽ bị cấm và bạn cần cấu hình lại Bức tường lửa.

Minh hoạ 3.18: Cấu hình bức tường lửa

3.4 Các thiết lập khác

3.4.1 Hỗ trợ ngôn ngữ

RedHat Linux cho phép cài đặt và hỗ trợ nhiều ngôn ngữ trong hệ thống của bạn. Bạn có quyền chọn cùng lúc nhiều ngôn ngữ nhưng phải quy định một trong số đó làm ngôn ngữ mặc định. Thông thường ngôn ngữ mặc định là ngôn ngữ bạn chọn lúc bắt đầu cài đặt, tuy nhiên khi chọn thêm nhiều ngôn ngữ thì bạn vẫn sẽ có khả năng thay đổi ngôn ngữ mặc định sau khi cài đặt (minh hoạ 3.19).

Việc chỉ cài một ngôn ngữ sẽ tiết kiệm một lượng đáng kể khoảng trống của đĩa cứng, nhưng bạn cũng chỉ có thể dùng duy nhất ngôn ngữ đó mà thôi.

Minh hoạ 3.19: Thiết lập hỗ trợ ngôn ngữ

3.4.2 Thiết lập cấu hình thời gian

Qua hộp thoại "Time Zone Configuration" (minh hoạ 3.20), bạn có thể tự quy định sẽ sử dụng giờ địa phương hay giờ GMT. Bạn sẽ chọn múi giờ bằng cách dùng chuột di chuyển con chạy trên bản đồ đến thành phố thích hợp và sau khi kích hoạt, một chữ X đỏ sẽ đánh dấu vị trí được chọn đó. Bạn cũng có thể chọn bằng cách cuộn thanh cuốn (scroll bar).

Bạn vẫn sẽ có thể quy định lại Time Zone sau khi cài đặt bằng cách thực hiện lệnh /usr/sbin/timeconfig.

Minh hoạ 3.20: Lựa chọn múi giờ

3.5 Thiết lập trương khoản người dùng

Các hệ điều hành Linux/UNIX quy định một user có quyền hạn tối cao, do đó mang tên Superuser hay còn gọi là root. Vì nắm trong tay những quyền hạn như của quản trị viên hệ thống, superuser có thể thực hiện nhiều việc hay hoặc dở.

Mục "Account Configuration" trước hết cho phép bạn thiết lập mật khẩu cho trương khoản root. Mật khẩu của root là chìa khoá cuối cùng cho hệ thống của bạn, do đó hãy chọn một mật khẩu dễ nhớ nhưng đảm bảo sự an toàn. Hộp thoại Root Password sẽ mời bạn gõ vào hai lần để xác nhận mật khẩu.

Minh hoạ 3.21: Thiết lập trương khoản người dùng

Nếu các user quên mật khẩu của họ thì bạn còn giúp được bằng quyền hạn của root. Trong đa số trường hợp quên mật khẩu root thì bạn buộc phải cài đặt lại toàn bộ hệ thống ; tuy nhiên bạn vẫn có khả năng khởi động lại từ đĩa mềm và chỉnh sửa tệp mật khẩu để phục hồi, hoặc khởi động Linux để vào chế độ Single, sau đó sẽ đặt lại mật khẩu mới cho root.

Xem "Xử lý an ninh mật khẩu"

Cũng trong màn hình này, bạn có thể tạo các trương khoản khác cho các user cần thiết, bằng cách bấm vào ô <Add>. Một hộp thoại (minh hoạ 3.22) sẽ hiện ra cho phép nhập tên user (User name – 8 ký tự) mật khẩu (password) và nhắc bạn khẳng định lại mật khẩu (Confirm).

Bạn cũng có thể bỏ qua, không tạo thêm user ngay mà để sau khi cài đặt xong sẽ thực hiện.

Xem "Quản lý trương khoản người dùng"

3.6 Thiết lập cấu hình xác thực

Màn hình "Authentication Configuration" (cấu hình xác thực) sẽ giúp bạn thiết lập quy tắc xác thực trên máy của mình. Nhưng nếu bạn đã quy định từ khi cài đặt rằng máy của mình là máy trạm hay máy chủ thì màn hình nói trên sẽ không hiện ra.

Bạn có thể chọn sử dụng các tiêu chuẩn chứng thực trên mạng như NIS, LDAP, Kerberos, hay SMB, bằng cách đánh dấu vào ô tương ứng và mở thẻ (tag) liên quan để điền những thông tin thích hợp.

Minh hoạ 3.23: Thiết lập cấu hình xác thực

- Enable MD5 password: cho phép bạn tạo các mật khẩu dài (tối đa 256 ký tự) thay cho mật khẩu chuẩn chỉ có 8 ký tự.

- Enable shadow password: cung cấp cho bạn một phương pháp lưu trữ mật khẩu an toàn. Mật khẩu thật sẽ không lưu trữ trong tệp /etc/password như các hệ thống chuẩn, mà lưu trữ trong tệp /etc/shadow để chỉ có root mới đọc được.

- Enable NIS: cho phép bạn sử dụng chung một mật khẩu trên một nhóm máy tính trong cùng một vùng NIS (Network Information Service). Bạn cần cung cấp thêm các thông tin về tên của vùng NIS và tên của một máy chủ NIS cụ thể. Bạn cũng có thể chọn để kích hoạt khả năng phổ biến thông điệp đến mọi máy trong mạng cục bộ hầu tìm một máy chủ NIS thích hợp.

- Enable LDAP (Lightweight Directory Access Protocol): cho phép sử dụng LDAP cho một vài hay toàn bộ giao dịch chứng thực. LDAP là một giao thức truy cập thông tin danh bạ điện tử trong hệ thống mạng. Bạn cần cung cấp địa chỉ IP của máy chủ LDAP (LDAP server) trong hệ thống, chọn cách nhận biết thông tin về user qua tên riêng DN (LDAP Base DN – Distinguished Name) và cho phép bạn gửi thông tin mật khẩu của user đã được mã hoá đến một máy chủ LDAP trước khi xác thực thông qua TLS (Transfer Layer Security).

- Enable Kerberos: Kerberos là một hệ thống bảo mật cung cấp các dịch vụ chứng thực qua mạng. Có 3 tuỳ chọn để bạn lựa, bao gồm: Realm, KDC, Admin Server.

- Enable SMB Authentication: cài đặt PAM để sử dụng một máy chủ SMB cho việc xác nhận các user. Cần cung cấp các thông tin về máy chủ SMB và về Nhóm SMB.

3.7 Chọn các gói phần mềm cài đặt

Đến đây thì hệ thống về căn bản đã sẵn sàng cài đặt. Tuy nhiên bạn còn phải chọn những gói phần mềm (package) muốn cài đặt, sau đó thiết lập cấu hình cho chúng.

Minh hoạ 3.24: Chọn các gói phần mềm cài đặt

Chương trình cài đặt sẽ liệt kê trong hộp thoại " Components to Install" các thành tố có thể cài đặt và bảng 3.5 giới thiệu một số gói phần mềm như vậy.

Thành phần	Mô tả
Printer Support	Cho phép in từ hệ Linux
Classic X Window System	Cung cấp môi trường đồ họa GUI cho mọi máy UNIX/Linux, tương tự như MS Windows
X Window System	Cung cấp hệ thống X Window (quản lý cửa sổ đồ hoạ)
Laptop Support	Cung cấp các ứng dụng chạy trên Laptop và hỗ trợ cho các Laptop chạy Linux
GNOME	Cung cấp các ứng dụng theo giao diện kiểu GNOME
KDE	Cung cấp các ứng dụng theo giao diện kiểu KDE
Sound and Multimedia Support	Cung cấp hỗ trợ âm thanh và multimedia trong môi trường X Window
Network Support	Cung cấp các ứng dụng và công cụ để gỡ lỗi và quản trị mạng, bao gồm các dịch vụ SNMP
Dialup Support	Giúp truy cập Internet qua modem điện thoại
Messaging and Web Tools	Cung cấp chương trình sử dụng thư điện tử, lướt duyệt Web, dọc và đưa tin lên Usenet
Graphics and Image Manipulation	Cung cấp các chương trình đồ hoạ chẳng hạn như các phần mềm GhostView, Screen Snapshot và GIMP.
News Server	Cho phép hệ thống hoạt động như một server Diễn đàn thảo luận, có thể nhận và phổ biến ý kiến trao đổi của những người sử dụng.
NFS Server	Giúp hệ thống có thêm hai chức năng là xuất khẩu và ghép kèm vào các hệ thống tệp khác trên mạng.
Windows File System	Cung cấp các dịch vụ SMB cho cả client và server MS Windows
Anonymous FTP Server	Cung cấp phần mềm máy chủ FTP vô danh (wu-ftp)
SQL Database Server	Quản tri các Cơ sở dữ liệu dùng SQL (MySQL, PostgreSQL...)
Web Server	Apache, phần mềm Web server rất phổ biến

Router/FireWall	Phần mềm quản lý định tuyến và bức tường lửa.
DNS Name Server	Cung cấp phần mềm cần thiết để chạy server tên miền riêng trên mạng với hệ điều hành Linux
Network Managed Workstation	Cung cấp công cụ quản lý máy trạm
Authoring and Publishing	Cung cấp các ứng dụng cho việc sáng tác và xuất bản
Emacs	Cài đặt phần mềm soạn thảo văn bản emacs, cung cấp X Window front-end cho emacs
Utilities	Cung cấp các công cụ tiện ích
Legacy Application Support	Hỗ trợ các ứng dụng bằng tương thích ngược (kế thừa các ứng dụng cũ)
Software Development	Cung cấp các trình biên dịch và công cụ lập trình (GNU **gcc**, các trình dịch Python, Perl...)
Kernel Development	Cung cấp các công cụ phát triển kernel
Windows Compatibility/ Interoperability	Cung cấp ứng dụng phỏng tạo/liên tác môi trường MS Windows
Games and Entertainment	Cung cấp các trò chơi chạy dưới X Window hoặc chế độ văn bản.
Everything	Cài đặt tất cả những gì có trên CD: Cần có khoảng 3.0 GB trống, chưa kể khoảng trống dành cho các tệp dữ liệu.

Bảng 3.5: Các gói phần mềm cài đặt chủ yếu

Ghi chú: Bạn có thể chọn từng gói phần mềm bằng cách đánh dấu mỗi ô tương ứng trên hộp thoại, hoặc cài đặt toàn bộ khi đánh dấu tuỳ chọn ấy. Muốn chọn một gói phần mềm để cài đặt, bạn chỉ cần di chuyển đến mục mình muốn và nhấn thanh Spacebar. Sau khi chọn xong tất cả những thành tố phần mềm muốn cài đặt, bạn bấm OK và bấm <Enter>.

Lưu ý: Sau này bạn có thể dùng chương trình RPM được mô tả trong giáo trình để cài đặt các gói phần mềm.

GNOME và KDE là 2 kiểu giao diện GUI của Linux (tương tự giao diện Windows 9x, 2000, XP). Bạn có thể chọn cả hai với điều kiện là đĩa cứng của bạn đủ lớn (khoảng gần 2 GB).

Minh hoạ 3.25: Chọn chi tiết cài đặt các phần mềm

Sau khi cài đặt, một hộp thoại khác báo cho bạn biết nơi chứa thông tin về các tệp sẽ cài đặt là /tmp/install.log. Bấm <Enter> để tiếp tục cài đặt.

Nếu bạn đánh dấu vào ô “Select individual package”, bạn có thể chọn lựa lại từng gói phần mềm mà bạn muốn hay không muốn cài đặt.

Minh hoạ 3.26: Mối tương quan giữa các phần mềm cài đặt

Có thể chọn một trong 2 cách liệt kê các gói phần mềm theo hình cây hoặc dàn ngang (Tree View hoặc Flat View) để dễ theo dõi và lựa chọn. Khi nhấp chuột lên một gói phần mềm bất kỳ, bạn sẽ đọc được các giới thiệu về nó ở phần dưới của màn hình. Bạn cũng có thể theo dõi mối quan hệ giữa các phần mềm sẽ được cài đặt khi bật chế độ “ Dependencies” (liên quan). Nếu bạn loại bỏ một số phần mềm liên quan cần thiết cho một số phần mềm khác thì RedHat sẽ thông báo ngay rằng bạn cần cài đặt lại các phần mềm đó để bảo đảm các phần mềm đang chọn hoạt động tốt.

3.8 Thiết lập cấu hình X Window

Lưu ý: Cố gắng chọn bìa điều khiển màn hình cho thật phù hợp, bởi vì trong số những thiết bị ngoại vi thì bìa điều khiển màn hình và màn hình là những thứ mà phần mềm có thể dễ dàng làm hỏng nhất. Nếu chọn sai loại bìa điều khiển màn hình, màn hình của bạn có rủi ro bị cháy.

Đến đây, hệ thống sẽ cài đặt server XFree86 thích hợp cho phần cứng máy bạn.

Xem “ Cài đặt hệ thống XFree86”

Ở bước tiếp theo màn hình sẽ hỏi bạn về các chíp đồng hồ (clockchip) trên bìa điều khiển màn hình. Các chíp này dùng để điều khiển tín hiệu video thông qua bìa màn hình. Nếu không được đồng bộ hoá, các tín hiệu có khả năng làm cháy màn hình của bạn. Bạn phải cung cấp tham số chính xác và nếu không có những thông tin ấy bạn nên dùng tham số mặc định cho Clockchip, nghĩa là No Clockchip Setting, sau đó bấm OK.

Tiếp theo hệ thống sẽ tự động trắc nghiệm (autoprobe) và thiết lập cấu hình X Window. Máy bạn có thể bị treo trong tiến trình autoprobe. Nếu máy treo, bạn chỉ cần khởi động lại máy và tiếp tục cài đặt. Bạn có thể bỏ qua phần tự động trắc nghiệm và tiếp tục cài đặt.

Trường hợp autoprobe thành công, máy sẽ yêu cầu bạn chọn lựa độ phân giải. Bạn có thể chọn lựa nhiều độ phân giải tuỳ vào khả năng xử lý của bìa điều khiển màn hình và màn hình. Cuối cùng chương trình cài đặt sẽ hướng dẫn bạn cách khởi động và đình chỉ hệ thống X Window.

Minh hoạ 3.27: Cấu hình giao diện đồ hoạ X Window

3.9 Cài đặt các gói phần mềm

Bây giờ đến phần khó khăn và lâu nhất: chờ đợi cho máy chuyển dữ liêu từ dạng nén để bung ra khoảng 3.0 GB chương trình. Thoạt tiên, chương trình thiết lập (setup) sẽ định dạng cho các phân vùng (nếu bạn có chỉ định) và cài hệ thống tệp vào các phân vùng, kế tiếp máy sẽ bắt đầu cài đặt phần mềm. Vừa cài đặt, hệ thống vừa báo cho bạn biết tiến trình đã đến đâu qua hộp thoại “ Tình hình cài đặt” (Install Status). Thời lượng tiến hành tuỳ thuộc vào những gì bạn muốn cài đặt và tốc độ của máy bạn.

Minh hoạ 3.28: Tiến trình cài đặt (1)

Ghi chú: Bạn có thể chọn các công việc tự động chạy khi khởi động máy bằng cách chỉnh sửa thủ công các tệp rc.d. (xem mục rc.d trong giáo trình), hoặc gõ lệnh /usr/sbin/ntsysv để hiển thị hộp thoại Services và chọn lại mọi thứ.

Minh hoạ 3.29: Tiến trình cài đặt (2)

3.10 Tạo đĩa mềm khởi động

Để tạo đĩa mềm khởi động, bạn cần chuẩn bị sẵn một đĩa mềm trắng và đưa nó vào trong ổ đĩa mềm của máy tính.

Minh hoạ 3.30: Tạo đĩa mềm khởi động

Đây là một việc rất cần thiết. Đĩa mềm này sẽ giúp bạn khởi động hệ Linux của mình khi mà các chương trình GRUB, LILO và các chương trình quản lý khởi động của các nhà sản xuất khác không hoạt động được.

Bạn cũng có thể tạo đĩa khởi động sau khi cài đặt xong, nghĩa là bạn có thể bỏ qua bước này.

3.11 Kiểm tra cấu hình X Window

Như đã giới thiệu, các hệ Linux và UNIX đời mới đã có một giao diện đồ hoạ gọi là X Window, gọi tắt là X.

Minh hoạ 3.31: Cấu hình giao diện X Window

Việc cấu hình giao diện X để bạn làm việc thoải mái như trong môi trường MS Windows lại không dễ dàng lắm. Bạn cần nắm được chính xác các tham số về hệ thống đồ hoạ của mình, bao gồm: bìa đồ hoạ (Video Card), màn hình (Monitor), hoặc các tần số tín hiệu đồng bộ dọc/ngang (Vertical/Horizontal Sync). Các tham số này thường có sẵn trong tài liệu đi kèm màn hình của bạn.

Trong trường hợp không thấy loại màn hình của mình có tên trong danh sách, bạn nên chọn loại màn hình " phổ quát" (Generic) tương đương và chỉnh lại tần số đồng bộ dọc/ngang.

Minh hoạ 3.32: Cấu hình căn chỉnh

Bạn cũng cần chọn thêm các tham số về độ phân giải kích thước và độ phân giải màu trong màn hình " Custom configuration". Khi đó, bạn nên chạy kiểm tra (**test**) để xem cấu hình có hoạt động được không. Nếu máy bị treo, nhiều lúc bạn có thể dùng tổ hợp 3 phím Ctrl+Alt+Backspace để thoát khỏi chế độ kiểm tra. Tuy nhiên không phải lúc nào cũng thoát được.

Bạn cũng cần chọn giao diện GNOME hoặc KDE làm giao diện mặc định.

Minh hoạ 3.33: Màn hình giao diện GNOME

3.12 Khởi động lại

Sau khi hoàn tất phần thiết lập và đặt xong cấu hình cho hệ thống Linux, bạn cần khởi động lại. Việc này do chương trình cài đặt tự thông báo, trước hết bạn chỉ cần lấy ra các đĩa mềm và đĩa CD đang còn nằm trong máy mình.

Khởi động lại Linux không giống như khởi động lại DOS bằng cách tắt rồi mở công tắc điện. Nếu làm như thế với Linux, hệ thống và kết cấu tệp có thể bị hỏng, vì vậy trong khi chạy Linux bạn đừng đột ngột tắt điện.

Xem "Sao lưu và phục hồi tệp"

3.13 Cài đặt Linux ở chế độ văn bản

Các phiên bản mới của RedHat cho phép bạn cài đặt Linux theo cả chế độ đồ hoạ (sử dụng chuột) và chế độ văn bản (bàn phím).

3.13.1 Cấu hình phần cứng cơ bản

Việc cơ bản cần chuẩn bị là nắm rõ các tham số phần cứng của máy tính:

- Đĩa cứng: các tham số về loại, kích thước, kiểu đĩa. Bạn cần phân biệt rõ các đĩa cứng đã có trong máy đang được cài đặt như đĩa chính (Master) hay đĩa phụ (Slave) và với giao diện nào: IDE 0, IDE 1 hay SCSI...

- Bộ nhớ: tổng lượng RAM đã cài trên máy tính.

- Ổ CD: loại ổ CD với giao diện IDE, SCSI hay một loại khác.

- SCSI: tên của bìa giao diện SCSI và đời (model) của nó.

- NIC (Network Interface Card – bìa giao diện mạng): tên nhà sản xuất và đời. Ngoài ra cũng cần biết một số chi tiết về cấu hình mạng của nơi bạn đặt máy. Trong trường hợp máy không nối mạng, bạn có thể tự chọn một cấu hình nào đó hợp lý.

- Chuột: loại chuột (3 phím hay 2 phím) và cổng giao tiếp (PS/2, cổng COMx, USB)...

Thông thường chương trình cài đặt có thể nhận ra được các phần cứng trong máy, nhưng việc chuẩn bị sẵn các thông tin trên sẽ giúp bạn điều chỉnh lại các nhận biết tự động của chương trình cài đặt.

3.13.2 Các màn hình ở chế độ văn bản

Dù Linux đang chạy ở chế độ văn bản, các cấu hình mà bạn cần cài đặt được hiển thị trên các màn hình có giao diện trực quan và cũng gần giống các màn hình sử dụng môi trường đồ hoạ. Thí dụ: minh hoạ 3.34 mô tả phần cấu hình mạng còn minh hoạ 3.35 chính là màn hình cấu hình phân vùng đĩa cứng ở chế độ văn bản.

Minh hoạ 3.34: Cấu hình mạng ở chế độ văn bản

Minh hoạ 3.35: Phân vùng đĩa cứng ở chế độ văn bản

3.13.3 Dùng bàn phím để di chuyển

Nên chú ý rằng trong các màn hình ở chế độ văn bản, bạn chỉ sử dụng bàn phím để di chuyển từ phím nhấn này đến phím nhấn khác nhằm chọn cấu hình hay hành động thích hợp. Bạn sẽ sử dụng các phím mũi tên, Tab và Alt+Tab để di chuyển. Cần thật cẩn thận không bấm nhầm phím.

3.13.4 Cài đặt ở chế độ văn bản từ đĩa CD

Bạn có thể cài đặt chế độ văn bản từ đĩa CD nếu máy của bạn hỗ trợ việc này và chọn bất kỳ một phím chức năng nào đó để chấm dứt việc cài đặt chế độ đồ hoạ như mặc định. Bạn chọn tiếp:

```
boot: text
```

Hay :

```
boot: text expert
```

để bắt đầu.

Trong một vài trường hợp, bạn có thể cần nhập thêm tham số cho việc khởi động:

```
boot: text mem=128M
```

Vì theo mặc định. Linux thường chỉ tận dụng được 64 MB RAM (trong đa số trường hợp với các bản RedHat 7.0 trở về trước).

3.14 Cài đặt qua mạng

3.14.1 Cài đặt từ máy chủ

Vì phiên bản RedHat 7.3 cho phép cài đặt Linux từ nhiều đĩa CD nên nếu muốn thực hiện việc cài đặt từ các máy chủ NFS, FTP hay HTTP, bạn phải chép các thư mục RedHat trên các CD vào đĩa cứng của máy chủ tương ứng. Bạn làm như sau (trên máy chủ):

Đưa đĩa CD#1 và thực hiện các lệnh sau:

```
mount /mnt/cdrom
cp-var /mnt/cdrom/RedHat/ location/df/disk/space
```

Trong đó /location/of/disk/space là thư mục bạn sẽ tạo (thí dụ /mirror/RedHat/i386). Sau đó **unmount** đĩa CD#1 để có thể chép đĩa CD#2:

```
unmount /mnt/cdrom
```

Đưa đĩa CD#2 vào và thực hiện những lệnh sau:

```
mount /mnt/cdrom
cp-var /mn/cdrom/RedHat /mirror/RedHat/i386
```

Trong đó mirror/RedHat/i386 là thư mục bạn sẽ tạo (thí dụ /export/7.2/). Sau đó **unmount** đĩa CD#2.

```
umount /mnt/cdrom
```

Thực hiện tương tự với đĩa CD#3 (nhiều CD tức là càng ngày Linux càng ôm thêm vào mình các ứng dụng). Tiếp theo, bạn đặt thư mục trên thành sử dụng được cho việc cài đặt bằng cách kết xuất nó.

```
export /mirror/RedHat/i386
```

Đưa dòng lệnh này vào tệp /etc/export và chạy lệnh:

```
# exportfs –r
```

3.14.2 Cài đặt bằng NFS

Hộp thoại NFS sẽ chỉ hiện ra nếu bạn cài đặt từ một máy chủ NFS sau khi đã khởi động máy từ mạng LAN hay từ một đĩa có giao diện PCMCIA và đã chọn tuỳ chọn NFS Image trong hộp thoại " phương pháp cài đặt". Hãy điền một tên miền đầy đủ (fully-qualified domain name – FQDN) hay địa chỉ IP của máy chủ NFS. Thí dụ, nếu bạn cài đặt từ máy tính có tên eastcoast trong miền RedHat.com, bạn hãy điền eastcoast.RedHat.com trong vùng "NFS Server field".

Kế đó điền thư mục đã được kết xuất ở trên (thí dụ như /mirror/RedHat/i386) mà trong đó có chứa thư mục của RedHat.

Nếu máy chủ NFS đã xuất ra một ảnh của cây thư mục cài đặt RedHat, hãy điền tên thư mục đó (bạn hãy hỏi người quản trị hệ thống về địa chỉ đó). Thí dụ nếu máy chủ NFS có thư mục /mirrors/RedHat/i386/RedHat, hãy điền /mirrors/RedHat/i386.

Minh hoạ 3.36: Cài đặt Linux bằng NFS

3.14.3 Cài đặt bằng FTP

Hộp thoại FTP sẽ hiện ra nếu bạn chọn cách cài đặt bằng giao thức FTP (từ một máy chủ FTP). Hộp thoại này cho phép bạn chỉ định máy chủ FTP sẽ được sử dụng. Hãy điền tên máy chủ FTP hay địa chỉ IP của nó rồi điền tên thư mục chứa bản cài đặt RedHat.

Thí dụ, nếu máy chủ FTP chứa thư mục /mirrors/RedHat/i386/RedHat, hãy điền /mirrors/RedHat/i386.

Nếu mọi việc đều chính xác, sẽ có một thông báo hiện ra báo rằng base/hdlist đã được khôi phục. Sau đó sẽ là hộp thoại đón chào (Welcome).

Minh hoạ 3.37: Cài đặt Linux bằng FTP

3.14.4 Cài đặt bằng HTTP

Hộp thoại HTTP chỉ xuất hiện khi bạn chọn cài đặt từ một máy chủ HTTP.

Điền tên hay địa chỉ IP của máy chủ HTTP đang chứa bản cài đặt của RedHat và tên thư mục chứa bản cài đặt này vào các ô cần thiết. Thí dụ máy chủ chứa bản cài đặt RedHat trong thư mục pub/mirrors/RedHat/i386/RedHat trong thư mục wwwroot của Web server, ta sẽ điền /pub/mirrors/RedHat/i386.

Nếu tất cả chi tiết bạn điền là chính xác thì một thông báo sẽ hiện ra cho biết base/hdlist đã tìm thấy.

Sau đó, màn hình Welcome xuất hiện và bạn tiếp tục việc cài đặt tương tự như trong chế độ đồ hoạ.

Minh hoạ 3.38: Cài đặt Linux bằng HTTP

Bảng 3.8 cho bạn thấy sự tương đương giữa chế độ văn bản và đồ hoạ trong cài đặt.

Dạng văn bản	Tham khảo phần cài đặt ở chế độ đồ hoạ
Màn hình ngôn ngữ sử dụng	Mục 3.3.1
Màn hình bàn phím	Mục 3.3.1
Phương pháp cài đặt	Mục 3.3.1
Màn hình xác định đĩa chứa phần mềm cài đặt	Mục 3.15 (không có phần tương ứng trong cài đặt ở chế độ đồ hoạ)
Cài đặt qua mạng	Mục 3.15
Chọn chuột	Mục 3.3.1
Màn hình đón chào	Mục 3.3.1
Kiểu cài đặt	Mục 3.3.2
Cập nhật	Mục 3.16
Phân vùng đĩa	Mục 3.3.3
Tự động phân vùng	Mục 3.3.3
Disk Druid	Mục 3.3.3
Fdisk	Mục 3.3.3
Cài đặt chương trình khởi động	Mục 3.3.4
Mật khẩu cho GRUB	Mục 3.3.4
Cấu hình chương trình khởi động	Mục 3.3.4
Cấu hình tên máy (hostname)	Mục 3.4.1
Cấu hình bức tường lửa	Mục 3.4.2
Cấu hình mạng	Mục 3.4.1
Chọn sự hỗ trợ về ngôn ngữ và ngôn ngữ mặc định	Mục 3.5.1
Chọn múi giờ	Mục 3.5.2
Mật khẩu cho root	Mục 3.6
Cấu hình trương khoản người dùng	Mục 3.6
Cấu hình xác thực	Mục 3.7
Chon các gói phần mềm	Mục 3.8
Cấu hình bìa điều khiển màn hình	Mục 3.9

Cài đặt các gói phần mềm	Mục 3.10
Tạo đĩa khởi động	Mục 3.11
Cấu hình màn hình	Mục 3.12
Căn chỉnh X	Mục 3.12
Hoàn thành cài đặt	Mục 3.13

Bảng 3.6: Tham khảo chéo giữa các chế độ cài đặt ở dạng văn bản và đồ hoạ

3.15 Nâng cấp Linux

3.15.1 Khi nào cần nâng cấp?

Đó là khi bạn muốn cập nhật hệ điều hành Linux và/hay các phần mềm đang có trên máy từ những phiên bản RedHat trước đây mà không muốn xoá các căn chỉnh trong những phần mềm thành phần mà bạn đã hoàn tất việc cài đặt.

Đó là khi bạn cũng muốn nâng cấp phần kernel (lõi) của Linux mà không muốn cài đặt từ đầu.

Khi nâng cấp, chương trình cài đặt sẽ lưu lại các cấu hình cũ (các tệp cấu hình *.cf, *.conf...) bằng cách chép lại và thêm vào phần mở rộng .rpmsave (thí dụ sendmail.cf.rpmsave). Đồng thời nó sẽ ghi nhận các thay đổi đó vào tệp /tmp/upgrade.log. Bạn sử dụng các tệp cấu hình cũ để so sánh với cấu hình mới hầu giữ cho hệ thống hoạt động tựa như trước.

Khi bạn cập nhật các gói phần mềm, cần chú ý sự phụ thuộc vào nhau của các gói phần mềm.

3.15.2 Cập nhật Linux

Khi hiện màn hình về chọn kiểu cài đặt (mục 3.3.2) bạn hãy chọn kiểu " nâng cấp hệ thống".

Minh hoạ 3.39: Cập nhật hệ thống

3.15.3 Cập nhật hệ thống tệp

Nếu trên máy bạn đang phân vùng đĩa cứng theo kiểu hệ thống tệp là ext2, chương trình cài đặt sẽ hỏi bạn có cập nhật lên chuẩn ext3 hay không. Đây chỉ là một khuyến nghị chứ không bắt buộc.

Nếu bạn chọn chuyển qua chuẩn ext3, dữ liệu hệ thống hiện hành không thay đổi gì.

ext2 hỗ trợ chuẩn hệ thống tệp UNIX cơ bản, nhưng có bổ sung thêm tính năng hỗ trợ tên tệp dài (đến 255 ký tự).

ext3 hỗ trợ thêm tính năng journaling (nhật ký). Nhờ tính năng này, việc chữa lỗi khi hệ thống bị treo (crash) sẽ nhanh chóng và nghiêm chỉnh hơn, bớt phải chạy chương trình **fsck** như trước (nếu bị lỗi nặng và lỗi phần cứng, ta vẫn phải chạy **fsck**).

3.15.4 Cấu hình cho việc nâng cấp

Sau màn hình về hệ thống tệp, bạn được quyền chọn chế độ nâng cấp: căn chỉnh theo ý riêng hay để chương trình cài đặt tự động.

Lúc này bạn được quyền chọn những gói phần mềm thành tố nào cần nâng cấp.

Minh hoạ 3.40: Cấu hình cho việc nâng cấp

3.15.5 Về chương trình khởi động

Bạn cũng có thể chọn lại chương trình khởi động theo chuẩn mới GRUB hay theo chuẩn LILO.

Bạn có thể chọn lại nơi sẽ tạo chương trình khởi động ở MBR (Master Boot Record) hay ở sector đầu tiên của phân vùng gốc (Root Partition).

Mặc dù có thể không cần cài chương trình khởi động của Linux (GRUB hay LILO) lên MBR, nhưng theo khuyến cáo của các chuyên gia, nếu bạn chỉ muốn dùng Linux, hay dùng nó với Windows 9x trên cùng một máy, thì bạn nên chọn vùng MBR để nạp chương trình khởi động.

Ngay cả với Windows NT hay Windows 2000, chúng tôi cũng đề nghị bạn cài GRUB vào vùng MBR thay cho NTLDR của Windows.

3.16 Gỡ bỏ Linux

Để gỡ bỏ Linux ra khỏi máy tính, bạn cần gỡ bỏ các thông tin về GRUB và LILO từ MBR, nếu bạn đã chọn việc cài đặt chúng trên đó.

Với Windows9x/NT/DOS bạn chỉ cần chạy lệnh:

```
FDISK /mbr
```

Trong đó /mbr là một tuỳ chọn “bí mật” (undocumented, không được giải thích trong các tài liệu chính thức của Microsoft).

Lưu ý: bạn phải khởi động bằng DOS (DOS) và dùng FDISK (cũng viết **fdisk**) của DOS chứ không dùng **fdisk** của Linux hay Unix.

Sau đó, nếu muốn gỡ bỏ luôn Linux trên đĩa cứng, bạn dùng tuỳ chọn xoá NON-DOS Partition của chương trình FDISK.

Ghi chú: bạn cũng có thể xoá bỏ các phân vùng bằng cách tiến hành cài đặt từ CD rồi từ màn hình phân vùng (DiskDruitd hay FDISK), bạn chọn “ xoá phân vùng”. Bạn nhớ lưu lại các giá trị bằng lệnh Write (w) để cập nhật bằng phân vùng đĩa. Sau đó, bạn bấm Ctrl+Alt+Del để khởi động lại.

3.17 Hỏi-đáp

Sau khi khởi động lại, màn hình GRUB hay màn hình LILO hiện ra tuỳ theo cách chọn của bạn. Nếu hệ điều hành cũ của bạn là Windows 2000 (hay Windows NT SP5 trở lên) và bạn đã chọn cài GRUB hay LILO lên MBR, bạn sẽ không thấy tuỳ chọn cho hệ điều hành cũ đó. Bạn không nên lo lắng vì chỉ cần sửa một chút trong Linux là bạn lại có thể chọn hệ điều hành cũ hay Linux tuỳ ý mà không cần phải dùng một chương

trình quản lý khởi động thứ ba (thí dụ như Partition Magic, BootStart, DriverStart..) nào cả.

Nếu bạn đã nạp chương trình khởi động GRUB vào MBR và đang có hệ điều hành Windows 2000 ở ổ đĩa thứ nhất (Primary Disk on IDE 1), bạn chỉ cần thêm các dòng sau vào tệp grub.conf (/boot/grub/grub.conf):

```
rootnverify (hd0,0)
makeactive
chainloader +1
```

Ý nghĩa của các dòng như sau:

Dòng 1 nêu lên tên mục chọn trên danh sách của GRUB

Dòng 2 cho biết chương trình khởi động NTLDR đang nằm trên phân vùng thứ 1 của ổ đĩa thứ 1 (ổ đĩa chủ ở khe cắm (slot) IDE thứ 1).

Dòng 3 chỉ định chương trình CRUB phải kích hoạt phân vùng này lên.

Dòng 4 chuyển vai trò chương trình khởi động lại cho chương trình khởi động trên phân vùng vừa được kích hoạt.

Bạn cũng căn chỉnh được chương trình khởi động mặc định khi thay đổi chỉ số xx trong dòng lệnh, với xx nhận giá trị 0 cho mục chọn thứ 1 :

```
default=xx
```

Bạn thử khởi động hệ điều hành cũ xem được hay không (trên nguyên tắc là phải được, nếu hệ điều hành ấy vẫn còn trên đĩa cứng). Nếu hệ đó là DOS, bạn bấm <Shift> và gõ ký tự nào bạn thường dùng để nhận diện phân vùng DOS lúc cài đặt LILO. Nếu lỡ gõ vào một ký tự không đúng, bạn bấm <Tab> để chọn lựa trong danh sách các hệ điều hành hợp lệ. Nếu bạn gặp vấn đề ở điểm này, nên đưa đĩa khởi động DOS vào để khởi động lại.

Trên nguyên tắc bạn phải khởi động được từ đĩa cấp cứu. Khi hệ thống chạy được DOS rồi thì hãy chạy thử đĩa cấp cứu Linux mà bạn đã tạo ra trong khi cài đặt. Nếu đĩa này không chạy được, bạn phải cài đặt lại toàn bộ Linux. Những vấn đề đầu tiên cần phải xem lại là kernel và phần cứng. Trước khi bắt đầu bạn phải chắc chắn rằng mình có các loại phần cứng tương thích. Nếu trong khi cài đặt bạn có ghi chép, hãy so sánh phần kernel với phần cứng xem chúng có tương thích không.

Hỏi: tôi có thể sử dụng ổ đĩa cứng có hơn 1023 cylinder được hay không?

Đáp: Được. Bạn có thể cài đặt Linux trên các phân vùng có số cylinder cao hơn cylinder thứ 1023, song để khởi động Linux, bạn NÊN cài đặt thư mục root và đặc biệt là thư mục /boot trên ổ đĩa cứng đầu tiên có dưới 1024 cylinder.

Hỏi: làm cách nào để bổ sung tham số tại dấu nhắc LILO?

Đáp: Một vài loại phần cứng đòi hỏi bổ sung nhiều tham số vào kernel trước khi kernel nhận ra phần cứng. Bạn có thể thực hiện việc này hoặc bằng cách chỉnh sửa tệp /etc/**lilo**.cof để cung cấp các tham số cần thiết, hoặc bằng cách thử có thêm nhiều thí dụ về tham số LILO.

Hỏi: Tại sao LILO lại treo ở LI?

Đáp: đây là một triệu chứng của vấn đề cylinder 1023 đã nói ở trên. Nếu bạn cài đặt hệ thống khởi động cao hơn 1023, LILO sẽ không thể khởi động máy. Bạn thử sử dụng đĩa mềm cấp cứu được tạo ra trong tiến trình cài đặt, hoặc bạn phân vùng lại ổ đĩa cứng và cài đặt Linux lại. Đôi khi cũng là do BIOS của bạn không hỗ trợ LBA32, bạn nên cấu hình lại phân vùng chứa **mount point**/boot.

Hỏi: Máy không tìm ra bìa SCSI

Đáp: Để chữa lỗi này bạn phải thêm vào một tham số về thời gian khởi động như sau:

```
LILO: linux qlogicafas=0x230,11,5
```

Bạn xem thêm tuỳ chọn append về cấu hình LILO tại trang **lilo**.confman.

Hỏi: Làm cách nào để tháo gỡ (uninstall) LILO?

Đáp: Muốn gỡ bỏ LILO và cài đặt lại boot record nguyên thuỷ, hãy sử dụng lệnh sau:

```
lilo –u/dev/hda
```

Lệnh này đại diện cho boot record của ổ IDE đầu tiên. Tuỳ theo máy bạn, các tham số có thể thay đổi, chẳng hạn nếu ổ đĩa cứng đầu tiên là SCSI, bạn sẽ gõ /dev/sda.

Hỏi: Có thể sử dụng LILO và Windows 95 trên cùng một máy tính hay không?

Đáp: Được, bạn cài đặt Windows 95 trước và Linux sau. Trong khi cài đặt, bảo Linux lưu LILO vào MBR. Bạn cũng có thể sử dụng một chương trình thương phẩm, chẳng hạn như System Commander. Bạn cũng có thể cài đặt Windows 98, Windows 2000, Linux và cả Solaris trên cùng một máy tính. Điều này phụ thuộc nhiều vào cấu hình phần cứng của bạn và sự cẩn thận thực hiện chính xác các bước cài đặt theo hướng dẫn của các mini HOW-TO như:

- Linux + NT-Loader
- Multiboot with LILO
- Multiboot with GRUB
- Linux + Win95
- Loadlin + Win95-98-ME
- Linux + DOS_Win95+OS/2
- Linux + WinNT

Hỏi: Cài CD như thế nào?

Đáp: Khi cài đặt RedHat sẽ đưa các mục ghi thích hợp vào tệp /etc/fstab như sau:

```
LABEL=/ /   ext3  defaults    1     1
LABEL=/boot/boot  ext3  defaults    1     2
none /dev/pts devpts    gid=5,mode=620    0     0
/dev/hdc2   /mhome      vfat  defaults    0     0
none  /proc proc  defaults    0     0
none  /dev/shm    tmpfs defaults    0     0
/dev/hda3   swap  swap  defaults    0     0
```

```
/dev/cdrom /mnt/cdrom iso9660 owner, exec, dev, suid, ro, noauto, kudzu 0 0
/dev/cdrom1 /mnt/cdrom1 iso9660 noauto, owner, kudzu, ro 0 0
/dev/fd0 /mnt/floppy auto noauto, owner, kudzu 0 0
```

Lưu ý việc sử dụng noauto cho mục ghi CD. Không có tham số này Linux sẽ cố gắng tự động cài CD khi khởi động. Điều này không gây ra vấn đề gì miễn là lúc ấy có CD trong ổ.

Nếu tệp fstab không có mục ghi, bạn có thể hoặc chỉnh sửa /etc/fstab hoặc dùng dụng cụ Control Panel của X Window để thêm vào thông tin thích hợp cho việc cài CD. Ngoài ra phải chắc chắn rằng **mount point** (điểm lắp ghép) mnt/cdrom là có thực. Nếu mục ghi đúng, bạn có thể chuyển thư mục (**cd**) đến **mount point** và ra lệnh sau:

```
mount /mnt/cdrom
```

Hỏi: Tôi có RedHat 5.0 và đã nâng cấp lên gói id.so RPM như liệt kê trong phần erata, tuy nhiên ứng dụng libc5 vẫn tạo ra seg fault. Có điều gì không ổn?

Đáp: Vấn đề libc5 bị sập là do nhiều nguyên nhân.Trước hoặc sau khi nâng cấp, có thể bạn đã cài đặt một phiên bản khác của libc mà tiến trình nâng cấp không ghi nhận là bản cũ, hoặc các thư viện libc5 được đặt vào chỗ nào đó gây ra xung đột. Nếu rơi vào trường hợp này, bạn gõ lệnh:

```
rpm-ga | grep libc
```

Màn hình sẽ hiển thị như sau:

```
glibc-devel-2.0.5c-12
libc-5.3.12-24
glibc-debug-2.0.5c-12
rpm-2.4.10-1glibcglibc-profie-2.0.5c-12
glibc-2.0.5c-12
```

Nếu nhìn thấy những thứ như libc-debug-5.3.12-18 hoặc libc-5.4.44-2, bạn sẽ phải gỡ bỏ các gói phần mềm ấy (bằng lệnh **rpm** –e libc-debug) và chạy ldconfig –v.

Tệp /etc/ld.so.conf đã thay đổi, không giống như tuỳ chọn mà bạn đã đặt ra trước. Để chạy phần nạp tệp cho tốt, bạn nên đặt tệp /etc/ld.so.conf theo thứ tự sau:

```
/usr/i486-linuxaout/lib
/usr/i486-linux-libc5/lib
/usr/openwin/lib/usr/x11r6/lib
```

Hỏi: nếu có một vài ứng dụng cụ bị sai về thời gian thì sao ?

Đáp: Một vài libc5 apps đòi hỏi /usr/lib/zoneinfo. Hoặc là bạn soạn lại cho libc6, hoặc bạn cung cấp một symlink bằng lệnh sau đây:

```
ln –s.../share/zoneinfo /usr/lib/zoneinfo
```

Hỏi: Đã cài đặt đầy đủ mọi bản nâng cấp song chương trình vẫn bị sai về thời gian thì sao?

Đáp: Xem lại phần thiết lập tại /etc/sysconfig/clock như thế nào. Nếu phần ấy có dạng như sau:

```
UTC=true
```

```
ARC=false
```

Thì có nghĩa là Linux cho rằng đồng hồ BIOS trong máy bạn được thiết lập theo múi giờ UTC hoặc GMT. Trong khi đó có thể đồng hồ thực tế lại theo múi giờ địa phương, do đó bạn nên đổi lại UTC như sau:

```
UTC=false
```

Hỏi: Khi khởi động, máy báo có một phần cứng PCI lạ thì sao?

Đáp: lỗi "Unknown PCI device" có nhiều nguyên nhân. Nguyên nhân đầu tiên và ít tai hại nhất là PCI ấy không phản hồi theo cách Linux có thể hiểu được, song Linux vẫn tiếp tục hoạt động. Thông thường khi gặp lỗi PCI thì cả hệ thống bị treo và đòi hỏi lắp card bus cho PCI. Vì đây là vấn đề phần cứng trong kernel, RedHat không thể làm gì hơn là chỉ dẫn bạn đến với người bảo trì (maintainer) của đoạn ấy trong kernel.

Người bảo trì có thể cung cấp thông tin và tìm hiểu phần cứng trong máy bạn để trong tương lai sẽ giúp bạn tốt hơn. Bạn có thể gặp người bảo trì tại địa chỉ: mailto:linux-pcisupport@cck.uni.kl.de

Ghi lại thông tin ở /proc/pci, vì đây là phần mô tả chính xác phần cứng của máy bạn. Thử tìm xem thiết bị nào mà hệ thống không thể nhận ra. Có thể đó là chipset của bìa mạch chính, hoặc là cầu nối PCI-ISA. Nếu không tìm thấy thông tin trong sách cẩm nang phần cứng, bạn thử đọc các tham số trên con chíp gắn ở bìa.

Hỏi: Linux không phát hiện ra bìa mạng tương thích NE2000 thì sao?

Đáp: đã có trường hợp vài bìa mạng tương thích NE2000 chạy được với các kernel cũ nhưng không làm việc được với các kernel từ 2.0.x trở về sau. Đối với một số trường hợp, có thể đối phó bằng cách buộc bìa phải làm việc với những thiết lập như sau:

```
insmod ne io = 0XXXX irq=Y
```

(Bạn hãy thay thế XXXX và Y bằng địa chỉ IO và số ngắt IRQ. Phần lớn các giá trị phổ thông của địa chỉ IO là 0x300 và 0x310, còn IRQ thì ở trong khoảng 0-15). Sau đó dùng ifconfig hoặc netfg để cấu hình bìa. Đôi khi mặc dù bìa được Linux nhận ra nhưng lại không đủ khả năng truyền các gói tin TCP/IP.

Nếu các thiết lập như trên chạy được, bạn thêm chúng vào /etc/conf.modules. Máy sẽ hiển thị đại loại như:

```
alias eth0 ne
options eth0 io=0xXXX irq=Y
```

Hỏi: Tôi đã cài đặt xong Linux và có vẻ như máy bắt đầu khởi động được, song khi đến phần gì đó gọi là sendmail thì hình như máy bị treo. Điều gì đã xảy ra và tôi phải làm sao?

Đáp: Sau khi cài đặt xong nếu máy có vẻ như bị treo khi chạy đến một vài tiến trình nào đó như là sendmail, apache, hoặc SMB, thì ắt hẳn phải có vấn đề về mạng. Nguyên nhân phổ biến nhất là vì Linux không tìm ra tên máy của bạn (dù bạn đã thiết lập hệ thống dịch vụ tên miền DNS). Cái máy đó hiện đang trong tình trạng tạm ngưng để chờ network timeout của phần dò tìm DNS và sau đó sẽ hiển thị dấu nhắc đăng nhập. Khi dấu nhắc này hiện ra, bạn hãy đăng nhập như là "root" và dò tìm thủ phạm.

Nếu bạn đang trực tiếp ở trên mạng với một server DNS, hãy kiểm tra lại tệp /etc/resolv.conf xem có những giá trị đúng với server DNS cho bạn hay không. Hãy hỏi quản trị viên mạng để có các giá trị đúng.

Nếu bạn sử dụng Linux trên mạng không có server DNS (hoặc nếu cái máy này sẽ là server DNS), bạn phải chỉnh sửa tệp /etc/hosts để có hostname và địa chỉ IP giúp cho việc dò tìm được chính xác.

Tệp /etc/hosts có hình thức như sau: với mymachine là tên của máy tính được lấy làm thí dụ:

```
127.0.0.0 localhost localhost.localdomain
192.168.200.1 mymachine mymachine.mynetwork.net
```

Chương 4. Cài đặt Caldera OpenLinux

Chương này nói về việc cài đặt bản phát hành Caldera OpenLinux. Tựa như Linux RedHat và Slackware, OpenLinux là bản phát hành đầy đủ của một hệ điều hành đa nhiệm và nhiều người sử dụng, có cùng lõi kernel Linux 2.x.

OpenLinux là một hệ thống gồm 2 phần mềm riêng biệt: OpenLinux Server và OpenLinux Workstation (tương tự các phiên bản của Windows 2000 như Data Center, Advanced Server, Server, Professional).

Những chủ đề chính sẽ được đề cập trong chương này gồm có:

- *Những yêu cầu khi cài đặt OpenLinux*
- *Cài đặt*
- *Chuẩn bị*
- *Chuẩn bị các đĩa mềm cài đặt*
- *Cài đặt OpenLinux*
- *Cài đặt các phần mềm*
- *Thiết lập cấu hình hệ thống*
- *Cài đặt LILO*
- *Trở về khởi điểm*
- *Giải quyết vấn đề*

4.1 Yêu cầu về cấu hình hệ thống

4.1.1 CPU

OpenLinux và các hệ đời mới - dựa trên kernel 2.4 - chỉ có thể chạy trên các CPU có hỗ trợ chuẩn PAE (Physical Address Extensions). Đó là các CPU sau đây:

- Intel Celeron
- Intel Pentium Pro
- Intel Pentium II
- Intel Pentium III
- Intel Pentium 4
- ADM Athlon
- ADM Duron
- ADM Thunderbird

OpenLinux Server không hỗ trợ các CPU sau (OpenLinux Workstation thì được):

- Intel Pentium
- Intel Pentium MMX
- AMD K6 and K6.2

4.1.2 Dung lượng đĩa cứng

Khác với Windows, OpenLiux không chỉ là một hệ điều hành mà trong nó còn chứa khá nhiều phần mềm dịch vụ ứng dụng, do đó yêu cầu về dung lượng đĩa cứng khá cao. Cụ thể như sau:

Bảng 4.1. Dung lượng đĩa cứng cần thiết cho máy trạm

Tuỳ chọn cài đặt	Tiếng Anh	Dung lượng đĩa
Dịch vụ tối thiểu	Minimum Server	1.0 GB
Dịch vụ thông dụng	Recommended Server	1.5 GB
Tối đa	All Packages	1.7 GB

Bảng 4.2. Dung lượng đĩa cứng cần thiết cho các máy chủ

Tuỳ chọn cài đặt	Tiếng Anh	Dung lượng đĩa
Tối đa	All Packages	1700 MB
Dịch vụ web	Web Server	550 MB
Quản lý tệp/in ấn	File/Print Server	550 MB
Dịch vụ mạng	Network Server	700 MB
Dịch vụ tối thiểu	Minimum Server	550 MB

4.1.3 Các yêu cầu khác

Giống như đã trình bày trong chương 2, ta cần ghi nhớ các thông số cơ bản về màn hình, độ phân giải tối đa của màn hình, loại bìa video, bìa mạng, modem, máy in, bìa âm thanh...

4.1.4 Nâng cấp từ các phiên bản trước

Với OpenLinux 3.1 (Server hay Workstation) ta đều có thể cài đặt như là một nâng cấp từ các phiên bản trước như 2.3, 2.4, 2.2...

4.2 Chọn chế độ cài đặt

Ta có thể bắt đầu việc cài đặt bằng cách khởi động máy tính bằng đĩa CD-ROM cài đặt hay từ đĩa mềm khởi động (không phải là đĩa khởi động của DOS).

Khi màn hình tuỳ chọn chế độ cài đặt xuất hiện, ta có thể chọn một trong các tuỳ chọn chế độ cài đặt sau:

- Standard Install Mode: chế độ cài đặt chuẩn là nội dung chính của chương này.

- ISA Hardware Support Mode: cũng giống như chế độ cài đặt chuẩn nhưng trình cài đặt sẽ có dò tìm và sử dụng các bìa PnP ISA.

- VESA Install Mode: sử dụng khi bạn chắc chắn rằng bìa đồ hoạ của bạn không nhận được sự hỗ trợ của Caldera Linux, khi đó bạn sẽ làm việc với giao diện VEGA chuẩn (640x480, 16 màu).

Bạn cũng có thể hoán đổi phím bấm và phỏng tạo phím giữa nếu chuột của bạn chỉ có 2 phím.

Trong trường hợp chuột của bạn không hoạt động, bạn có thể chọn lại kiểu (Mouse type), loại (model) và cổng giao tiếp (Port) bằng cách bấm <Tab> để di chuyển và bấm thanh <Space bar> để chọn.

Ghi chú: Bạn có thể xem lại chương 2, 3 để hiểu hơn về các xác định Port.

Minh hoạ 4.3: Chọn chuột

Các tổ hợp phím sau đây sẽ giúp bạn lựa chọn mà không dùng phím Tab

Tổ hợp phím	Lựa chọn
Alt+T	Chọn kiểu chuột
Alt+M	Chọn loại chuột
Alt+P	Chọn cổng giao tiếp
Alt+L	Cho hoán đổi phím bấm
Alt+E	Phỏng tạo phím giữa

4.6 Cấu hình bàn phím

Màn hình cấu hình bàn phím cho phép bạn chọn loại bàn phím bạn đang sử dụng (Minh hoạ 4.4).

Thông thường, user chọn loại Generic-104 keys (loại phổ quát, có 104 phím).

Bạn cũng cần chọn sơ đồ bàn phím thích hợp (layout). Nếu không, các phím bạn bấm sẽ có khả năng bị hệ điều hành hiểu sai.

Minh hoạ 4.4: Chọn kiểu bàn phím

Khác với RedHat, bạn không thể sử dụng các phím đánh dấu (Dead-key).

4.7 Lựa chọn màn hình và chế độ đồ hoạ

Nếu bạn chọn chế độ cài đặt là Cautious, Expert, VESA thì màn hình chọn bìa đồ hoạ mới hiện ra. Bạn có thể chọn lại các thông số nếu các xác định của LIZARD không chính xác. Bạn cũng có thể chọn "Do not install graphics support" để không sử dụng các gói phần mềm X.

Minh hoạ 4.5: Chọn Bìa đồ hoạ.

Bạn cũng chỉ thấy màn hình chọn "Màn hình" khi chọn chế độ cài đặt như trên.

Minh hoạ 4.6: Chọn màn hình.

Bạn chọn màn hình thật đúng loại mà bạn đang có, hoặc hãy chọn loại Typical.

Bạn cần chú ý dãy tần số sử dụng của màn hình vì nếu bạn chọn không chính xác (quá cao, hay vượt ngưỡng giới hạn của màn hình), có thể màn hình của bạn sẽ bị hỏng.

Qua màn hình "Chế độ đồ hoạ", bạn sẽ chọn chế độ màu và tần số làm tươi thích hợp (Minh hoạ 4.7).

Bạn chú ý nên kiểm tra lại chế độ đã chọn bằng cách nhấp vào nút "Test this mode". Đây là một kiểm tra tổng hợp của các tuỳ chọn trên. Nếu việc kiểm tra này không được thực hiện hoàn hảo, bấm một phím bất kỳ để trở lại màn hình chọn. Bấm trên nút <Back> để trở về màn hình trước để chọn lại.

Ghi chú: Đôi khi máy bạn có thể bị treo khi thực hiện việc kiểm tra. Khi đó bạn hãy thử lại việc cài đặt theo chế độ VESA hay Cautious.

Minh hoạ 4.7: Chọn chế độ đồ hoạ.

4.8 Chuẩn bị đĩa

Cần thật cẩn trọng khi thực hiện các chọn lựa trên màn hình này, nhất là khi bạn muốn cài thêm OpenLinux trên một hệ thống đã chứa sẵn dữ liệu (Minh hoạ 4.8). Có 5 tuỳ chọn:

- Update: cập nhật phiên bản trước của OpenLinux, lưu được các cài đặt và các tệp cấu hình.

- Entire Hard Disk: xoá sạch đĩa và định dạng lại toàn bộ đĩa.

- Free Disk space: chỉ sử dụng phần đĩa trống chưa phân vùng. Không có dữ liệu nào bị xoá nhưng khi đĩa đã bị phân vùng hết thì tuỳ chọn này không dùng được.

- Prepared partitions: sử dụng các partitions đã cài đặt Linux.

- Customs: Chỉ khi bạn là chuyên gia hay bạn muốn thử làm chuyên gia. Kết quả của việc bạn sử dụng tuỳ chọn này là có thể là bạn xoá sạch đĩa của mình, hay tạo ra một đĩa không thể khởi động.

Minh hoạ 4.8: Chuẩn bị đĩa.

4.8.1 Tuỳ chọn "Update"

Bạn chọn Update. Màn hình Update hiện ra, hãy chọn lấy Phân vùng mà bạn dự định sẽ cài đặt OpenLinux. Nhấp chuột vào "Analyze Parrtition" để LIZARD xác định xem có thể cập nhật thành công? Kết quả hiện lên khung văn bản.

Minh hoạ 4.9: Cập nhật hệ thống phân vùng có sẵn.

Bấm vào <Next> trình LIZARD sẽ tiến hành việc cập nhật (Minh hoạ 4.9).

4.8.2 Tuỳ chọn "Installation Disk Drive"

Khi bạn đã chọn tuỳ chọn này, nếu bấm <Next>, màn hình chọn ổ đĩa sẽ hiện ra để bạn định sẽ sử dụng ổ đĩa nào để cài đặt.

Minh hoạ 4.10: Chọn ổ đĩa cài đặt.

Chọn ổ đĩa và bấm vào nút <Prepare selected disk **for** Linux>

Minh hoạ 4.11: Sử dụng phần đĩa trống.

Bấm <Next> để tiếp tục với màn hình khác.

4.8.3 Tuỳ chọn “Free Disk Space”

Chọn ổ đĩa đang còn vùng đĩa chưa phân vùng. Sau đó bấm <Next> để tiếp tục.

4.8.4 Tuỳ chọn “Prepared Partition”

Minh hoạ 4.12: Sử dụng phân vùng đã chuẩn bị.

Bạn cần chọn phân vùng gốc (root partition – **mount point** là /). Rồi bấm <Next> để chọn các phân vùng cần định dạng lại (Minh hoạ 4.12).

Minh hoạ 4.13: Phân vùng đĩa tuỳ chọn.

4.8.5 Tuỳ chọn “Customs”

Màn hình “Select partition” sẽ hiện ra cho phép bạn chọn và xác định **mount point** cũng như kích thước của các partitions (Minh hoạ 4.13).

Bạn có thể xoá một phân vùng cũ, định lại kích thước, **mount point** bằng cách bấm trên các nút ổ cột bên phải màn hình (delete, edit...).

Sau khi đã chọn hoàn chỉnh, bấm <Write> để lưu lại hoặc bấm <Reset> để trở về cấu hình cũ.

Sau khi lưu lại, bấm <Next> để tiếp tục định dạng đĩa. Màn hình “Format Partitions” sẽ lại xuất hiện. Bấm “Format chosen partitions” để thực hiện việc định dạng. Bấm <Next> để tiếp tục, sau khi định dạng xong (Minh hoạ 4.14).

Minh hoạ 4.14: Định dạng phân vùng

4.9 Lựa chọn và cài đặt phần mềm

Tuỳ theo bạn đang cài đặt cho một server (máy chủ) với OpenLinux Server hay cho một máy trạm với OpenLinux Workstation, mà màn hình chọn lựa các thành tố của phần mềm sẽ hiện ra tương ứng.

4.9.1. Server

Có các tuỳ chọn như sau: (Minh hoạ 4.15)

- All Packages: Cài đặt tất cả các gói phần mềm, nói cách khác là biến máy của bạn thành một Máy chủ tổng hợp.

- Web Server: Cài các thành phần cần thiết cho một máy chủ Web.

- File Printer Server: cài đặt các phần cần thiết cho một máy chủ quản lý tệp và máy in.

- Network Server: tập hợp các thành phần cơ bản của một máy chủ quản lý mạng: DHCP, DNS, NNTP, Sendmail, FTP...

- Minimum Server: cấu hình tối thiểu của một máy chủ OpenLinux.

Và một check box “Refine Selection” cho phép bạn chọn các thành tố tuỳ ý.

Minh hoạ 4.15: Tuỳ chọn phần mềm cho server.

4.9.2. Workstation

Ta có các tuỳ chọn sau: (Minh hoạ 4.16)

- All Packages: cài đặt toàn bộ các gói phần mềm có trên đĩa.

- Minimum: chỉ cài tối thiểu phần mềm đủ để khởi động và chạy Linux. Ta có thể cài đặt thêm các gói phần mềm về sau.

- Recommended: cài đặt toàn bộ môi trường phát triển (các trình biên dịch, diễn dịch, ngôn ngữ...)

Và một check box "Refine selection" cho phép cài đặt theo yêu cầu và tuỳ chọn.

Minh hoạ 4.16: Tuỳ chọn phần mềm cho Workstation.

4.9.3. Tuỳ chọn "Refine selection"

Khi này màn hình "Refine Selection" hiện ra (Minh hoạ 4.17).

Minh hoạ 4.17: Định nghĩa lại sự lựa chọn.

Cây "Collection" ở bên trái sẽ cho biết các hạng mục phần mềm.

Danh sách các gói phần mềm tương ứng với hạng mục trên hiện ra ở bên phải.

Khi bạn chọn một phần mềm bên phải, nội dung của gói phần mềm đó sẽ hiện lên ở vùng "Package Details".

Các mục đã chọn màu đỏ là các gói phần mềm đặc biệt quan trọng cho hệ thống hoạt động bình thường nên không thể gỡ bỏ.

Trong một mục chọn còn có cả các tiểu mục, khi có một tiểu mục nào chứa được chọn, mục chọn đó sẽ có ô chọn màu xám.

Nên chú ý đến kích thước của đĩa khi cài đặt thêm hay bớt các gói phần mềm.

4.10 Tạo user

Màn hình "Set Login mane" yêu cầu bạn phải xác định mật khẩu đăng nhập của root và tạo thêm user cho hệ thống (Minh hoạ 4.18).

Minh hoạ 4.18: Tạo user và mật khẩu.

Như bạn đã biết, việc đăng nhập vào hệ thống với account là root là không khuyến khích vì dễ phá huỷ hệ thống, nhất là khi bạn mới tiếp xúc với Linux.

Bạn nhập thông tin về user mới (không phải là root) vào ô "Real Name", đồng thời nhập "Login Name" của user này. Bấm <Add> để thêm user ấy vào danh sách.

Bạn có thể chú ý rằng ở phía dưới của màn hình, có một dòng thông báo các tiến trình đang chạy ("Packages"). Đó là trình LIZARD đang ngầm cài các gói phần mềm mà bạn đã chọn. Điều này giúp cho tiến trình cài đặt được nhanh hơn.

4.11 Lập cấu hình mạng

Bạn cấu hình mạng cho máy thông qua màn hình "Set up Network". Nếu bạn đang cài đặt máy trong một hệ thống mạng sử dụng địa chỉ IP cố định, bạn cần có các thông tin về địa chỉ IP được cấp, Sub Netmask và Gateway.

Minh hoạ 4.19: Cấu hình mạng.

Bạn cũng cần điền các thông tin về tên máy với các chi tiết về tên miền hay máy chủ tên miền. Bạn phải sử dụng tên máy đầy đủ (FQDN-Fully Qualified Domain Name) chứ không dùng tên tắt hoặc **alias**.

Khi bạn không có bìa mạng, bạn hãy chọn Disable để hệ thống không mất thời gian kiểm tra khi khởi động.

4.12 Cấu hình quản lý mồi

Tương tự như trường hợp cài đặt RedHat, bạn có thể chọn cài đặt trình quản lý mồi (Boot Loader) vào MBR (Master Boot Record) hay không. Trong trường hợp máy bạn đang có nhiều hệ, trên màn hình sẽ hiện đủ các phân vùng có chứa hệ điều hành để bạn có thể chọn hệ nào sẽ được nạp theo mặc định.

Minh hoạ 4.20: Cấu hình quản lý mồi

4.13 Cấu hình modem

Nếu trình LIZARD phát hiện được chính xác modem mà bạn có, nhất là khi bạn mở sẵn Modem, màn hình cấu hình Modem sẽ không hiện ra.

Chọn loại Modem thông qua danh sách model, nếu không thấy đúng loại, hãy chọn loại Generic tương ứng.

Chọn Device thích hợp với cổng giao tiếp giữa Modem với máy chủ tính theo quy ước sau:

Cổng trong DOS/Windows	Thiết bị trong OpenLinux
COM1	/dev/ttyS0
COM2	/dev/ttyS1
COM3	/dev/ttyS2
COM4	/dev/ttyS3

4.14 Cấu hình máy in

Bạn chọn máy in theo danh sách "Model", tuỳ theo loại máy in mà bạn có thể vùng Variant sẽ được bật lên để chọn cho thích hợp.

Nếu máy in của bạn đang nối trực tiếp với máy tính và đang được bật, trình LIZARD sẽ phát hiện tự động. Với các máy in từ xa, bạn cần cấu hình sau khi cài đặt xong (Minh hoạ 4.21).

Xem thêm "In ấn"

Bạn nhấn vào nút <Add> để thêm máy in vào danh sách.

Minh hoạ 4.21: Chọn máy in.

4.15 Chọn múi giờ

Bạn sử dụng danh sách hay bản đồ để xác định múi giờ thích hợp. Bạn cũng có thể sử dụng giờ quốc tế GMT hay giờ địa phương theo tuỳ chọn.

Minh hoạ 4.22: Chọn múi giờ.

4.16 Hoàn thành việc cài đặt.

Trong khi tiến trình cài đặt ”Packages” chưa đạt 100%, bạn có thể chuyển qua màn hình “Game” để giải trí với trò chơi “Solitaire” (tương tự như trò chơi Solitaire trong Windows).

Khi trình LIZARD đã chấm dứt việc chép các gói phần mềm vào đĩa, bạn sẽ chấm dứt việc cài đặt bằng cách bấm <Next> để qua màn hình tạo đĩa cấp cứu.

4.17 Tạo đĩa cứu hộ

Caldera cho phép bạn tạo đĩa mềm cứu hộ (Rescue disk) để có thể truy nhập vào hệ thống OpenLinux khi bạn không thể khởi động được từ đĩa cứng vì một lý do nào đấy.

Bạn chỉ cần đưa vào ổ đĩa mềm một đĩa trắng và bấm <Write Disk>

Sau khi đĩa cấp cứu được tạo xong, bấm <Finish>, lấy đĩa và cất kỹ vào nơi an toàn. Nhớ ghi nhãn rõ ràng để không bị sử dụng nhầm.

4.18 Một vài thủ tục cài đặt khác

4.18.1 Các tham số khởi động

Để sử dụng dòng lệnh trong việc khởi động (boot), ta tiến hành theo các bước sau:

- Khởi động từ CD-ROM hay đĩa mềm.

- Tại màn hình khởi động, di chuyển vệt sáng đến mục chọn “Standard Installation” hay “Unattended Installation” rồi bấm phím <e>.

- Di chuyển con trỏ đến dòng bắt đầu bằng kernel, lại bấm <e> lần nữa.

- Tại cuối dòng, bắt đầu bằng từ kernel, thêm tham số khởi động như trong bảng dưới đây và bấm <Enter>. Bạn có thể ghi nhiều tham số, các tham số cách nhau bằng dấu phẩy <,>.

- Bấm <b> để khởi động với các tham số này.

Chế độ	Tham số khởi động
Nhắc LUI khởi động từ một máy chủ cài đặt (installation server) thông qua đường dẫn mặc định	lizard=auto
Nhắc LUI khởi động từ đĩa mềm	lizard=floppy

Yêu cầu khi cài đặt cần dùng đến đĩa mềm chứa module (modules floppy)	install er=modules
Không cần dò tìm bìa mạng	install er=noether
Không cần dò tìm bìa PCMCIA	install er=nopcmcia
Cấu hình mạng đặc biệt cho một máy chủ NFS (NFS installation server)	install er=asknet

4.18.2 Nạp các trình điều khiển trong khi cài đặt

Đây là một ưu điểm của trình LIZARD trong việc cho phép người dùng cài đặt các trình điều khiển cho các phần cứng mà OpenLinux 3.1 chưa hỗ trợ.

Trước hết bạn cần có một đĩa chứa trình điều khiển cập nhật. Trên đĩa này cần có một cấu trúc thư mục đặc biệt.

- Đầu tiên cần có thư mục /lib/modules/override/

- Các thư mục dưới đây sẽ được tạo ra bên dưới thư mục trên khi cần thiết:

Bảng 4.3. Cấu trúc thư mục của /lib/modules/override

Thư mục	Mô tả
atm	Asynchronous Transfer Mode (ATM) drivers
block	Block device drivers
cdrom	Proprietary CDROM drivers (not IDE or SCSI)
fc4	Fiber Channel SCSI drivers
ode	IDE/ATAPI drivers
ieee1394	High performance serial driver (aka Firewire)
ospv	IPv4 technology drivers (RARP, IP masquerading)
ipv6	Experimental IPv6 drivers
misc	Miscellaneous drivers (irDA, tape drives)
net	Network bìa/chipset drivers
pcmcia	PCMCIA/CardBus drivers
scsi	SCSI bìa/chipset drivers
sound	Kernel sound drivers
usb	USB drivers
video	Video4Linux drivers

Bất kỳ trình điều khiển nào trong bất kỳ một thư mục con nào đều có thể chép đè lên trình điều khiển cũ hay thêm vào trong thư mục chứa trình điều khiển cũ.

Đĩa cài đặt này phải được định dạng theo ext2. Bạn có thể sử dụng tiện ích explore2fs (http://uranus.it.swin.edu.au/~jn/linux/explore2fs.htm) hay sử dụng các biến thể UNIX có hỗ trợ filesystem của Linux, như Free BSD hay Open BSD.

Bạn chỉ có thể dùng một đĩa cài đặt duy nhất cho mỗi lần cài đặt.

Nếu bạn cài đặt và khởi động bằng đĩa mềm, bạn sẽ được nhắc để hoán đổi đĩa cài trình điều khiển (modules disk) với thông báo "Mounting module disk".

Nếu bạn cài bằng CD, bạn sử dụng tham số khởi động như phần trên để chắc chắn sẽ nhận được thông báo nhắc đưa đĩa module vào.

4.18.3 Thực hiện cài đặt tự động

Có 3 cách cài đặt tự động (LUI: Lizard Unattendented Installation):

- Từ một máy chủ.

- Từ một phân vùng.

- Từ đĩa mềm.

Cần chú ý rằng LUI không chuyển các user không hệ thống (non-system users) từ một hệ thống đã cài đặt sang hệ thống mới.

Ghi chú: root và tty là những ví dụ về user hệ thống.

Nếu bạn chọn, bạn có thể chép các tệp /etc/password và /etc/shadow từ hệ thống cũ sang hệ thống mới, khi muốn sao chép các users không hệ thống từ một máy chủ đang sẵn chứa sang một hệ thống mới cập nhật. Tuy nhiên nếu bạn có cài thêm các ứng dụng xác nhận như PAM, ... thì sẽ có khó khăn.

4.18.3.1 Cài đặt từ một máy chủ

Bạn dùng phương thức này nhằm tập trung nguồn cài đặt trong một máy chủ phục vụ việc cài đặt (Installation Server) cho các máy con nằm trong mạng.

4.18.3.1.1 Cấu hình một máy chủ DHCP

Trên máy chủ cài đặt, tạo một cấu trúc thư mục con bên dưới Lizard trong thư mục nguồn. Khi đó, thư mục lizard của bạn sẽ tương tự như điểm **mount** của CD. Đó là vị trí mặc định để LIZARD truy cập thông tin.

Mục DHCP cần chỉ rõ vị trí này trong mục filename. Ta có một ví dụ như sau:

```
host phantom {
hardware ethernet 00:23:37:8B:05:92;
fixed-address 10.10.10.10;
option host-name "phantom.domainname.org";
next-server 10.10.10.127;
filename "/share/openLinux-3.1/cd1/";
}
```

Trong trường hợp này, thư mục nguồn sẽ như sau:

```
/share/OpenLinux-3.1/lizard
/share/OpenLinux-3.1/cd1
/share/OpenLinux-3.1/cd2
```

Nếu DHCP chưa được cài đặt trên hệ thống hay bạn muốn chép đè lên các cấu hình cũ, bạn cần sử dụng tham số er=asknet.

Còn khi cài đặt từ máy chủ cài đặt, bạn sẽ cần dùng tham số lizard=auto.

4.18.3.1.2 Cài đặt một máy chủ NFS

Ta thực hiện các bước sau:

- Cài đặt một máy chủ cài đặt NFS theo hướng dẫn của NFS-HOWTO

- Xoá hay thêm các gói phần mềm đã được cài đặt trong thư mục <cd1 base directory>/col/install/RPMS/ trên máy chủ đó. Chú ý cần xét đến các phụ thuộc giữa các gói phần mềm.

- Bắt đầu cài đặt từ máy chủ cài đặt này. Bạn cần chọn <All Packages>.

4.18.3.2 Cài đặt từ một phân vùng

Khi bạn không thể cài đặt từ CD-ROM hay từ một máy chủ NFS, bạn có thể cài đặt từ một phân vùng trên máy của bạn.

- Chép tất cả thư mục của các đĩa CD cài đặt vào thư mục gốc của phân vùng đó. Không được chép vào thư mục con.

- Tạo đĩa mềm khởi động (xem mục 4.18.3.3).

- Khởi động máy từ đĩa mềm này. Trình dò tìm phần cứng tự động sẽ phát hiện ra phân vùng có chứa các tệp cài đặt và bắt đầu cài đặt từ phân vùng đó.

4.18.3.3 Cài đặt từ đĩa mềm

4.18.3.3.1 Tạo đĩa mềm khởi động

Có thể tạo đĩa mềm khởi động cho OpenLinux từ DOS/WINDOWS hay từ trong Linux.

A- Từ DOS/WINDOWS: Bạn thực hiện các bước sau:

1. Đưa đĩa CD1 vào ổ đĩa CD-ROM (ở đây chúng tôi xem ổ đĩa CD Rom là ổ D:)

2. Đưa một đĩa mềm vào ổ đĩa mềm (ổ đĩa A:)

3. Thực hiện lần lượt các lệnh sau từ DOS/DOS Prompt:

C:\>d:

D:\> **cd** \co1\launch\floppy

D:\>..\..\tools\rawrite\rawrite3

Enter disk image source file name: install.144

Enter target diskette driver: a:

Insert a formatted diskette into drive A: and press Enter.

4. Lấy đĩa mềm ra, ghi nhãn là "Installation Diskette"

5. Đưa một đĩa mềm khác vào và thực hiện các lệnh gần như bước 3 :

D:\>..\..\tools\rawrite\rawrite3

Enter disk image source file name: modules.144

Enter target diskette driver: a:

Insert a formatted diskette into drive A: and press Enter.

6. Lấy đĩa mềm ra, ghi nhãn là "Modules Diskette".

Và bạn đã có đủ đĩa để có thể cài đặt từ đĩa mềm.

B- Từ Linux:

1. Đưa đĩa CD1 vào ổ CD-ROM

2. Đưa một đĩa mềm vào ổ đĩa mềm

3. Thực hiện các lệnh sau từ một cửa sổ terminal, hay từ console:

mount /mnt/ cdrom

cd /mnt/cdrom/co1/launch/floppy

dd if=install.144 of = /dev/fd0 bs=8192

4. Lấy đĩa mềm ra, ghi nhãn là "Installation Diskette".

5. Đưa một đĩa mềm khác vào và thực hiện các lệnh sau:

mount /mnt/ cdrom

cd /mnt/cdrom/co1/launch/floppy

dd if=modules.144 of=/dev/fd0bs=8192

6. Lấy đĩa mềm ra, ghi nhãn là "Modules Diskette".

Và bạn đã có đủ đĩa để có thể cài đặt từ đĩa mềm.

4.18.3.3.2 Tạo đĩa LUI

Từ hệ thống đã cài đặt OpenLinux 3.1, đăng nhập làm root, thực hiện lệnh sau:

```
/bin/mklizard – floppy
```

Lệnh này sẽ tạo ra trên đĩa mềm thư mục /etc/ và 2 tệp rules và fsinfo.cnf.

Thư mục /etc chứa tất cả các tệp dùng cho việc cài đặt, bao gồm cả tệp mẫu /etc/fsinfo.cnf.sample. Các tệp này chứa các thông tin cài đặt màn hình, lắp ghép (**mount**), trình quản lý khởi động, mật khẩu...

Tệp rules quản lý việc LIZARD thực hiện như thế nào tại từng thời điểm trong quá trình cài đặt.

fsinfo.cnf lại định nghĩa cách phân vùng cho LIZARD.

Ta có một tệp rules mẫu như sau:

```
# rules – the config file for LIZARD's Unattended Install (LUI) mode\
# (floppies>=360)
# Syntax:
# RULE_TOKEN MATCH_PATTERN START_SCRIPT PROFILE XCONFIG PKG_LIST\
# FSINFO_FILE FINISH_SCRIPT
# Ruleset 1: The "floppy" rules.
# To run LUI from floppy use the keyword "floppy" followed by\
# 127.0.0.1.
# All fields after 127.0.0.1 are subject to globbing with the\
# algorithm explained in the "LIZARD unattended install HOWTO"\
# available from http://www.caldera.com/support.
# Run unattended install (LUI) with a configuration floppy.
# floppy 127.0.0.1 bin/start.sh etc/profile etc/SF86Config\
# etc/pkgs.sel etc/fsinfo.cnf bin/finish.sh
# ====================================
# Ruleset 2: The "hostname" rules.
# Hostname entries start with keyword "hostname" followd by\
# IP address.
# They win over network entries if both match for our machine.
# All fields after your IP address are subject to globbing with the\
# algorithm explained in the "LIZARD unattended install HOWTO"\
# available from http://www.caldera/com/support.
# my workstation (adjust and remove '#' at beginning to enable\
# this rule).
# hostname 192.168.1.1 bin/start.sh etc / profile etc/XF86Config\
# etc/pkgs.sel etc/fsinfo.cnf bin/finish.sh
# ====================================
# Ruleset 3: The "network" rules
# Network entries start witch the keyword "network" followed by a\
# NET address.
# They should be used on pools of identical machines only.
# All fields after your NET address are subject to globbing with the
# \ algorithm explained in the "LIZARD unattended install HOWTO"\
# available from http://www.caldera.com/support.
# my network (adjust and remove '#' at beginning to enable this rule)
# network 192.168.1.0 bin/start.sh etc/profile etc /XF86Config\
# etc/pkgs.sel etc/fsinfo.cnf bin/finish.sh
```

Chương 5. Bắt đầu sử dụng Linux

Chương này sẽ giúp bạn thiết lập trương khoản người sử dụng (user account), giới thiệu một vài lệnh căn bản và tiện ích của hệ điều hành Linux mà bạn mới cài đặt xong lên máy PC. Bạn có một hệ điều hành đa nhiệm và đa người dùng của riêng mình nên cứ mạnh dạn bắt tay vào thao tác để có kinh nghiệm thực tế, vì có thể bạn không được gặp dịp may như thế trên những hệ UNIX thuần tuý. Ngoài ra, với Internet, bạn sẽ có cơ hội tải về hàng ngàn ứng dụng nguồn mở và miễn phí từ thế giới Linux.

Sau đây là các chủ đề chính của chương này:

- *Thiết lập trương khoản*
- *Quản lý người sử dụng*
- *Những lệnh căn bản*
- *Làm việc với các tệp DOS*
- *Đóng chương trình Linux*
- *Chạy các chương trình Linux*
- *Chơi trò chơi trên Linux*
- *Chạy các chương trình DOS*
- *Chạy các chương trình Windows*

5.1 Thiết lập trương khoản

Việc khởi động Linux mất gần một phút, cuối cùng dấu nhắc hiện lên màn hình và mời bạn đăng nhập vào hệ thống. Toàn bộ thông báo đại loại có thể như sau:

```
RedHat Linux Release 7.3 (Valhala)
Kernel 2.4.18-3 on an I686
Web login:
```

Bạn sẽ thấy các số hiệu khác nhau tuỳ theo phiên bản Linux được cài đặt.

Còn nếu đã cài đặt X Window, bạn có thể có màn hình login (xem minh hoạ 5.1).

Minh hoạ 5.1: Màn hình login

Đến đây bạn phải nhập vào tên người sử dụng (user name) và mật khẩu (password). Trương khoản người sử dụng giúp Linux phân biệt bạn với nhiều người khác mà nó phải phục vụ cùng lúc hoặc ở các thời điểm khác nhau.

Linux chấp nhận nhiều trương khoản, mỗi trương khoản cung cấp cho từng người sử dụng một thư mục mặc định, gọi là thư mục "nhà" (home directory).

Việc thiết lập trương khoản làm cho người sử dụng chỉ được thao tác trong phạm vi vài thư mục nào đó của hệ thống và với một số các câu lệnh nhất định mà thôi, bởi vì mục tiêu đầu tiên của trương khoản là để bảo vệ sự riêng tư của từng cá nhân.

5.1.1 Giao tiếp qua dòng lệnh

Bạn nhập dòng lệnh cho Linux cũng giống như cho DOS hoặc cho những hệ điều hành giao tiếp qua dòng lệnh với người dùng. Linux đòi hỏi sự chính xác với từng ký tự trong câu lệnh, kể cả việc phân biệt chữ thường với chữ hoa.

Nếu xảy ra trường hợp Linux không hiểu một câu lệnh nào đó, bạn nên kiểm tra xem mình viết đúng hay chưa. Đa số các câu lệnh sẽ được thực thi ngay sau khi bạn bấm phím <Enter>.

5.1.2 Lịch trình nhập lệnh

Nhiều shell Linux có lệnh history để xem lại lịch trình các câu lệnh được gõ vào. Lịch trình nếu không bị xoá sẽ được dùng như một thứ nhật ký của phiên sử dụng. Bạn có thể bấm phím <[redacted] để hiện lịch trình, rồi di chuyển con chạy trong đó và bấm <Enter> để kích hoạt một lệnh tự chọn mà không phải gõ lại nó.

Trong thí dụ sau, người sử dụng tên là Lan Anh dùng lệnh history và màn hình hiển thị một lịch trình các câu lệnh đã nhập, đại loại như:

```
[lan_anh@web~]$history
1 clear
2 adduser
3 history
```

Sau khi đã có lịch trình, bạn có thể chọn lại một lệnh trong đó bằng cách bấm phím <[redacted] và lướt con chạy cho đến khi gặp lệnh thích hợp. Hoặc bạn có thể bấm phím <!> rồi nhập số của lệnh mà bạn muốn Linux thi hành lại. Thí dụ khi muốn máy thi hành lại lệnh **adduser** trong lịch trình trên, bạn gõ như sau:

```
[lan_anh@web~]$!2
```

Bạn có thể định nghĩa số của từng dòng trong lịch trình ra lệnh tại tệp cấu hình profile ở trương khoản của người sử dụng.

Xem "Tìm hiểu các shell của Linux" để biết thêm về tệp cấu hình profile

Ghi chú: Linux có nhiều loại shell và một số shell không có tiện ích lịch trình.

5.1.3 Nhập lệnh bằng sao ghép

Nếu bạn đã cài đặt chuột và một chương trình tiện ích mang tên "selection", bạn có thể sao vài đoạn chữ từ các vùng khác nhau của màn hình để ghép thành một câu lệnh.

Muốn chọn một đoạn chữ, bạn bấm và giữ phím trái của chuột rồi kéo lê con chạy đi hết đoạn chữ (để nó đổi màu thành âm bản), sau đó bấm phím phải của chuột để sao đoạn chữ sang dòng lệnh.

Thao tác này rất có ích khi bạn muốn nhập một câu lệnh dài.

5.1.4 Tự động điền lệnh

Linux còn một cách khác để nhập lệnh. Bạn gõ vài ký tự đầu của tên lệnh rồi bấm phím <Tab>. Linux sẽ tìm trong thư mục lệnh xem tệp nào có tên bắt đầu giống như

những ký tự mà bạn vừa gõ vào, rồi tự động điền đủ tên tệp vừa được phát hiện. Trong trường hợp có vài lệnh mang các ký tự khởi đầu giống nhau, Linux sẽ phát ra âm thanh "bip" và tự động viết đủ tên lệnh cho đến ký tự chung cuối cùng mà những lệnh đó đều có.

Thí dụ bạn muốn chép một lệnh mang tên todo_Monday sang tệp todo_today. Bạn gõ **cp** to tại dấu nhắc rồi bấm <Tab>, Linux sẽ bíp và tự động viết đủ phần chung của tên tệp tại dòng lệnh như sau:

```
[lan_anh@web~]$sp todo_
```

Nếu vào lúc này bạn gõ M (nghĩa là ký tự đầu tiên của chữ Monday) và bấm <Tab>, Linux sẽ tự động điền đủ todo_Monday vào dòng lệnh.

5.2 Quản lý người sử dụng

Người chịu trách nhiệm gìn giữ các trương khoản của người sử dụng trong hệ thống được gọi là quản giá trị viên hệ thống. Quản giá trị viên hệ thống có nhiệm vụ thiết lập các trương khoản và một số các công việc khác mà bạn có thể tìm hiểu tại phần "Quản giá trị hệ thống" của giáo trình này.

Trên hệ Linux của bạn, bạn là quản giá trị viên hệ thống, do đó bạn chịu trách nhiệm thiết lập trương khoản cho chính mình, cho gia đình và bè bạn.

Muốn thêm vào trương khoản cho chính mình, bạn phải đóng vai trò quản giá trị viên hệ thống để tạo nó ra. Đôi lúc quản giá trị viên hệ thống còn được gọi là superuser bởi vì người này nắm quyền kiểm soát trên toàn bộ hệ thống. Để bắt đầu hành trình Linux, thoạt tiên bạn phải đăng nhập với tư cách là superuser qua trương khoản root.

5.2.1 Đăng nhập và đăng xuất

Muốn đăng nhập (log in) với quyền hạn của superuser, bạn gõ root tại dấu nhắc đăng nhập và Linux sẽ hỏi mật khẩu.

Mật khẩu ngăn cấm những người không thẩm quyền lén đăng nhập vào những trương khoản mà họ không có quyền dùng. Linux bảo vệ mật khẩu bạn gõ vào bằng cách không cho hiện lại (echo) các ký tự, nghĩa là không hiển thị những gì đang gõ vào, do đó bạn cố gắng gõ đúng mật khẩu.

Nếu gõ vào tên hoặc mật khẩu không hợp lệ, Linux sẽ báo lỗi như sau:

```
web login: lan_anh
Password: password
Login incorrect
web login:
```

Để đăng xuất (log out), bạn gõ logout. Lệnh này sẽ đưa bạn trở về dấu nhắc đăng nhập. Nếu lệnh này không hoạt động, bạn cần gõ lệnh **exit.**

5.2.2 Thêm người sử dụng trong Slackware

Trong Linux Slackware, sau khi đăng nhập với tư cách là root, bạn bổ sung người sử dụng mới vào hệ thống hiện hành bằng cách gõ lệnh **adduser**:

```
[root@web~] # adduser
Adding a new user. The user name should be not exceed 8 characters in
length, or you may run into problems later.
Enter login name for new account (^C to quit):
```

Bạn cần chú ý đến dấu nhắc, tức là chỗ mà ngay sau nó bạn phải gõ lệnh vào. Dấu nhắc bắt đầu bằng host name là cái tên mà bạn đã chọn trong khi cài đặt hệ thống và tiếp theo là ký tự ~ tức dấu sóng (tilde), kết thúc bằng dấu thăng #. Linux sử dụng ký tự ~ để biểu thị thư mục nhà (/home) của trương khoản hiện hành (sẽ giải thích ở phần sau). Tại thời điểm này host name là root@web và dấu ~ tượng trưng cho thư mục hiện hành, nghĩa là bạn đang ở trong thư mục ấy. Nếu bạn ra lệnh **adduser** từ thư mục /usr/bin, thì dấu nhắc sẽ có dạng:

```
[root@web /usr/bin] #
```

Ký tự cuối là dấu thăng (#), được Linux quy ước dùng để chỉ trương khoản của superuser. Các user bình thường thì có ký tự dollar ($) ở cuối dấu nhắc.

Tiếp theo, bạn có thể thấy cảnh báo nếu có sai chính tả và ngữ pháp tại các dấu nhắc ("should be not" hoặc "you may run" …). Những sai sót này không ảnh hưởng đến an toàn máy; chỉ nhắc bạn nhớ rằng Linux là một hệ thống chạy được và chạy tốt, song cũng có khác các thương phẩm.

Bạn hãy gõ vào tên một người sử dụng dài không quá tám (8) ký tự và bấm <Enter>. Sau đây là một thí dụ khi cô Lan Anh tạo lập trương khoản:

```
Enter login name for new account (^C to quit): lan_anh
Editing information for new user [lan_anh]
Full Name: Phan Lan Anh
GID[100]: <Enter>
Checking for an available UID after 500
501...
First unused uid is 502
UID[502]:<Enter>
Home directory: [/home/lan_anh] :<Enter>
Shell [/bin/bash]:<Enter>
Password: xxxxxxx
Information for new user [lan_anh]:
Home directory: [/home/lan_anh] Shell: [/bin/bash]
Password: [xxxxxxx] uid: [502] gid: [100]
Is this correct? [y/N]:y
Adding login [lan_anh] and making directory [/home/lan_anh]
Adding the files from the /etc/skel directory:
./.kermc -> /home/lan_anh/./.kermc
./.less -> /home/lan_anh/./.less
./.lessrc -> /home/lan_anh/./.lessrc
./.term/ -> /home/lan_anh/./.term
```

```
./.term/.termrc ->/home/lan_anh/./.termrc
./.emacs -> /home/lan_anh/./.emacs
[root@web~] #
```

Ghi chú: Bạn cần gõ tên đầy đủ của người sử dụng để về sau nhận diện được trương khoản ấy, rồi nhập số định danh nhóm (group ID) và số định danh người sử dụng (user ID), nhưng vào thời điểm này bạn đừng lo âu về những chi tiết đó. Linux dùng chúng để xác định các thư mục và các tệp mà bạn có quyền truy cập. Bạn có thể chấp nhận các giá trị mặc định (nằm bên trong ngoặc vuông) bằng cách bấm phím <Enter> sau mỗi lần máy hỏi.

Tiếp theo Linux nhắc bạn nhập tên home directory cho người sử dụng, tức thư mục sẽ dành cho người sử dụng khi đăng nhập. Đây là vùng để trương khoản của người sử dụng ấy dùng để lưu các tệp và làm việc. Linux cung cấp một thư mục mặc định dựa vào tên của người sử dụng. Nếu thư mục mặc định ấy được bạn chấp nhận thì bấm <Enter>. Nếu không, bạn gõ vào một tên khác và bấm <Enter>. Tạm thời bạn nên chấp nhận các mặc định do lệnh **adduser** đề nghị.

Đến đây Linux yêu cầu bạn xác định loại shell cho người sử dụng. Shell là một chương trình diễn dịch (interpreter) tiếp nhận dòng lệnh được gõ vào và thực hiện một số lệnh nhất định. Kể từ đầu đến giờ, bạn đang dùng một shell gọi là bash. Vào thời điểm này, bạn chỉ cần chấp nhận tuỳ chọn mặc định bash.

Xem "Tìm hiểu về Shell"

Tham số quan trọng cuối cùng là mật khẩu của trương khoản. Tốt nhất, đối với mỗi trương khoản bạn nên có một mật khẩu khác nhau.

Tiếp theo, Linux sẽ hiển thị tất cả mọi thông tin được nhập vào và hỏi bạn xem đã chính xác chưa. Nếu chưa, bạn gõ n (hoặc bấm <Enter>, bởi vì No là lựa chọn mặc định) sau đó trở lại chỉnh sửa những chỗ cần thiết. Chỉ khi mọi thứ đã chính xác, bạn mới gõ y.

Linux hiển thị một loạt tệp chép từ thư mục /etc/skel sang thư mục "nhà" của người sử dụng. Đấy là những tệp cấu hình cho một số mục như terminal của người sử dụng và cách hoạt động của một số chương trình chẳng hạn như **emacs** và **less.**

Sau khi thêm trương khoản vào hệ thống, bạn kiểm tra sự hiện diện của nó bằng một trong hai cách. Cách nhanh nhất là dùng tiện ích **finger** để xem người sử dụng có trương khoản hay chưa. Dạng tổng quát của câu lệnh này là :

```
finger tên_user
```

Thí dụ:

```
[root@web~] # finger lan_anh
Login: lan_anh
Name: Phan Lan Anh
Directory: /home/lan_anh
Shell:/bin/bash
Never logged in.
No Mail.
```

```
No Plan.
[root@web~] #
```

Nếu người sử dụng có một trương khoản, Linux sẽ hiển thị tình trạng ấy; nếu không thì sẽ thông báo là không có trương khoản.

Cách kiểm tra thứ hai một trương khoản là đăng nhập thật sự vào trương khoản ấy để thử xem Linux có cho vào hay không. Bạn có thể thử bằng nhiều cách:

- Đăng xuất sau đó lại đăng nhập bằng tên người sử dụng mới.

- Sử dụng lệnh **su.**

- Dùng lệnh login.

- Sử dụng một trong sáu terminal ảo do Linux cung cấp để đăng nhập vào trương khoản mới (nên nhớ là Linux có đặc tính đa người dùng).

Lệnh	**Mô tả**
Logout	Đăng xuất khỏi trương khoản đang dùng và trở về dấu nhắc đăng nhập. Không thể truy cập vào trương khoản nói trên nếu không đăng nhập lại.
su tên_user	Chuyển sang trương khoản của người khác và sẽ nhập mật khẩu của trương khoản mới này. Nếu không khai tên_ user, thì mặc định là root. Người sử dụng cũ không bị đăng xuất và vẫn ở trương khoản cũ.
Login tên_ user	Cũng là chuyển người sử dụng, nhưng người sử dụng cũ bị đăng xuất. Nếu để trống tên_user thì hệ thống sẽ trở về dấu nhắc đăng nhập bình thường.
<Alt-Fx>	Cho phép sử dụng 1 trong 6 terminal ảo bằng cách bấm <Alt> và một phím chức năng từ F1 đến F6. Sẽ hiển thị một màn hình để có thể đăng nhập trương khoản mới. Tức là dù ở một trương khoản song vẫn có thể vào trương khoản khác và tổ hợp <Alt-Fx> giúp di chuyển qua lại giữa hai trương khoản ấy.

Bảng 5.1: Các cách kiểm tra một trương khoản

Ghi chú: Nếu định thêm người sử dụng mới vào một trương khoản mà bạn đã tạo ra từ trước và không phải là root, bạn sẽ không có quyền sử dụng lệnh **adduser**, một câu lệnh chỉ dành riêng cho superuser. Khi đó bạn hãy đăng nhập lại với tư cách root.

5.2.3 Thêm người sử dụng mới trong RedHat Linux

Phiên bản RedHat Linux có tự động hoá chức năng tạo thêm người sử dụng mới; từ dòng lệnh, bạn gõ:

```
[root@web/root] #adduser lan_anh
```

Thực chất, lệnh trên là một shell script nằm ở /usr/sbin; muốn ra lệnh đó, bạn phải là superuser. Giá trị của các thuộc tính sẽ có của người sử dụng mới đã được khai báo sẵn trong tệp /etc/default/**useradd** hay /home/etc/default/**useradd.**

Trình script trên chỉ là một tệp văn bản, nó tạo ra các thư mục và tệp cần thiết cho người sử dụng mới. Điều còn lại phải làm là thiết lập mật khẩu người sử dụng khi

người ấy đăng nhập. Việc thay đổi mật khẩu sẽ được bàn tới tại mục "Thay đổi mật khẩu".

Xem "Làm việc với shell script" và "Xem nội dung của một tệp"

5.2.4 Dùng bảng điều khiển RedHat để quản lý người sử dụng

Nếu cài đặt XFree86 chung với RedHat Linux, bạn có thể mở cửa sổ thiết lập cấu hình Quản lý người sử dụng và nhóm người sử dụng (User/Group Manager) trong bảng điều khiển (Control Panel) như minh hoạ 5.2 để thêm người sử dụng mới, xoá bỏ hoặc cho nghỉ một người sử dụng, hoặc thay đổi những thiết lập cho một người sử dụng. Bạn có thể dùng lệnh sau : từ cửa sổ shell (hay xterm, hay Run), gõ:

```
redhat-config-users
```

Minh hoạ 5.2: Trình quản lý user/group

Muốn thao tác một trương khoản của người sử dụng, bạn chọn người sử dụng từ hộp thoại sau đó chọn nút thích hợp. Bảng 5.2 mô tả chức năng từng nút.

Minh hoạ 5.3: Màn hình người sử dụng

Nút	**Mô tả**
Add	Hiện hộp thoại Add User, giúp thiết lập các thuộc tính từng người sử dụng, chẳng hạn như thư mục "nhà" và mật khẩu.
Deactivate	Tạm treo trương khoản của một user để sau này còn sử dụng. Lý do treo có thể là user ấy nghỉ phép hoặc bị kỷ luật. Có thể nén các tệp của user ấy để tiết kiệm chỗ trống trên ổ đĩa cứng và chờ đến khi tái kích hoạt.
Reactivate	Tái kích hoạt trương khoản của người sử dụng.
Remove	Xoá một user ra khỏi hệ thống. Các thư mục và tệp của user ấy sẽ bị xoá sạch. Linux cho phép sao lưu chúng trước khi xoá hẳn.
Edit	Chỉnh sửa các chi tiết trương khoản của user như mật khẩu (trong trường hợp quên mật khẩu), shell và nhóm của user.
Exit	Thoát khỏi RH Linux User/Group Manager.

Bảng 5.2: Các nút quản lý user/group của RH Linux

Nhấp chuột vào nút Add sẽ hiển thị hộp thoại Add User như ở minh hoạ 5.2, bạn thiết lập cấu hình trương khoản cho từng người sử dụng bằng cách điền vào các trường của hộp thoại. Bảng sau mô tả các trường cùng với chức năng tương ứng.

Trường	**Mô tả**
Username	Tên của người sử dụng dùng để đăng nhập hệ thống
Password	Mật khẩu của người sử dụng đang đăng nhập. Muốn phân phối mật khẩu người sử dụng, phải dùng lệnh Edit từ menu; màn hình sẽ hiển thị một hộp thoại khác để nhập mật khẩu mới cho người sử dụng. Tổ hợp này cũng cho phép bỏ trống trường mật khẩu bằng

	cách chọn giá trị "none" hoặc khoá mật khẩu.
UID	Đây là một trường do hệ thống sinh ra. Xem thêm chi tiết tại chương "Quản lý trương khoản người sử dụng".
Primary Group	Nhóm sơ cấp người sử dụng. Trường này giúp đưa những người sử dụng vào thành từng nhóm, với từng quyền hạn riêng biệt.
Full Name	Tên họ đầy đủ của người sử dụng.
Home	Thư mục "nhà" của người sử dụng. Thông thường vị trí này ở bên trong thư mục /home.
Shell	Shell mặc định, vị trí làm việc ban đầu của một trương khoản. Một hộp thoại cho phép chọn bất kỳ shell nào mà RedHat Linux sẵn có.

Bảng 5.3: Các tuỳ chọn của hộp thoại Add User

5.2.5 Thay đổi mật khẩu

Bạn thường phải đổi mật khẩu (hoặc đặt mật khẩu cho một trương khoản mới), nhất là cho trương khoản gốc (root) vì nó cần được bảo vệ thật cẩn thận. Để thay đổi mật khẩu, bất kỳ phiên bản Linux hoặc UNIX nào đều dùng lệnh passwd, nó đòi nhập cả mật khẩu cũ lẫn mới và kiểm tra mật khẩu mới vừa nhập vào. Nếu bạn chưa có mật khẩu cho trương khoản (thậm chí quên mất mật khẩu), hãy sử dụng lệnh passwd để thay đổi. Việc thay đổi mật khẩu diễn ra như sau:

```
[lan_anh@web~]$passwd lan_anh
Changing password for lan_anh
Enter old password: old-password
Enter new password: new-password
Re-type new password: new-password
```

Nếu bạn làm không đúng, Linux sẽ thông báo là mật khẩu chưa được thay. Linux cũng yêu cầu mật khẩu phải bao gồm ít nhất sáu ký tự.

Cẩn thận: Đừng quên mật khẩu! Nếu quên mật khẩu của người sử dụng, bạn phải thay đổi thông tin về trương khoản ấy. Nếu quên mật khẩu trương khoản root, bạn phải sử dụng đĩa mềm khởi động (được tạo ra trong tiến trình cài đặt) để khởi động lại hệ thống và đổi mật khẩu. Bạn cũng có thể thay đổi mật khẩu của root bằng cách khởi động lại hệ thống ở chế độ Single.

Bạn có thể để mật khẩu trống bằng cách chọn "none" trong hộp thoại RH Add/Edit User, sau đó cho người sử dụng tự lập mật khẩu mới bằng lệnh passwd. Bạn cũng có thể chỉnh sửa tệp /etc/passwd và gỡ bỏ mật khẩu đã mã hoá từ bản ghi của người sử dụng (trong trường hợp không có khai báo shadow). Thông thường RedHat xác định số ký tự tối thiểu của password là 6, tuy thế ta có thể thay đổi được, thí dụ ta dùng công cụ linuxconf như ở minh hoạ 5.4.

Minh hoạ 5.4: Màn hình quy định việc quản lý mật khẩu

5.3 Sử dụng các lệnh cơ bản

Những mục sau đây sẽ hướng dẫn bạn sử dụng các lệnh cơ bản để điều khiển hệ thống, trong đó có một số lệnh thực sự là những trình tiện ích mà Linux dùng để mở rộng tập hợp các câu lệnh của mình. Những chương trình ấy nằm trong các thư mục /bin, /sbin và /usr/bin.

5.3.1 Dùng man để tìm trợ giúp cho câu lệnh

Muốn nhận được trợ giúp trực tiếp cho các lệnh Linux, bạn gõ:

```
man xyz*
```

Linux sẽ hiển thị từng trang thông tin liên quan đến từng lệnh bắt đầu bằng các ký tự xyz. Bạn có thể xem lần lượt các lệnh này bằng cách bấm phím q, khi đó nội dung từng lệnh sẽ hiện ra. Đặc điểm này của **man** chỉ có từ RedHat 7.2.

Nếu chưa biết chắc chắn mình sẽ dùng lệnh nào, bạn thử gõ tham số -k và nhập một từ khoá liên quan đến chủ đề bạn đang quan tâm. Lúc ấy **man** sẽ tìm trong các tệp trợ giúp (được gọi là trang **man**, hoặc trang manual) có chủ đề chứa từ khoá ấy. Linux cũng cung cấp một **alias** (bí danh) cho lệnh ấy, gọi là **apropos.**

Nếu bạn nhập lệnh **man ls,** Linux hiển thị trợ giúp cho lệnh **ls**, bao gồm tất cả các tham số. Lệnh **man** –k **cls** cung cấp danh sách các câu lệnh có chữ **cls** trong tệp trợ giúp. Lệnh **apropos cls** cũng giống như **man** –k **cls.**

5.3.2 Sử dụng các lệnh can thiệp vào thư mục

Cũng giống như những hệ điều hành khác mà bạn có dịp dùng qua, Linux có nhiều câu lệnh để tạo ra, xoá bỏ, di chuyển thư mục và hiển thị thông tin của thư mục.

5.3.2.1 Chuyển đổi thư mục hiện hành bằng lệnh cd

Cũng như DOS và các hệ điều hành khác, Linux chứa các tệp trong một cấu trúc cây gọi là thư mục. Bạn đi đến một tệp qua đường dẫn từ thư mục gốc bằng ký tự /. Do đó tệp cấu hình emacs cho người sử dụng tên lan_anh có thể được xác định như sau:

```
/home/lan_anh/.emacs
```

Nếu trước nay ban từng quen thuộc với hạn chế của DOS là tám ký tự cho tên tệp và ba ký tự cho cái đuôi, nay bạn sẽ thích thú hơn vì Linux không hạn chế số ký tự cho tên tệp.

Xem "Tìm hiểu tên tệp và tên đường dẫn"

Linux cũng sử dụng khái niệm về một home directory (thư mục nhà), thư mục này được xác định khi thêm một trương khoản vào hệ thống. Thông thường, thư mục "nhà" của một người sử dụng được xác định bằng ký tự tilde (ký tự sóng ~). Khi người sử dụng muốn chép một tệp từ thư mục hiện hành /usr/home/lan_anh vào thư mục "nhà" của mình, bạn có thể dùng ký tự tilde thay vì tên của thư mục.

```
cp.emacs ~
```

Muốn di chuyển trong cấu trúc Linux, bạn dùng lệnh chuyển thư mục **cd**. Nếu bạn gõ **cd** vào mà không kèm tham số nào, Linux đưa bạn về thư mục "nhà" của bạn. Muốn

chuyển từ thư mục này sang thư mục khác, bạn dùng lệnh **cd** như khi bạn sử dụng DOS, nghĩa là **cd** thư_mục_mới. Linux cũng dùng dấu chấm (.) để đại diện thư mục hiện hành và dấu chấm chấm (..) đại diện cho thư mục mẹ. Thực ra chính DOS mới phỏng tạo UNIX, chứ UNIX/Linux không phỏng tạo DOS.

Ghi chú: Hãy cẩn thận khi dùng dấu sổ ngược ở đường dẫn thư mục. Trong khi DOS dùng dấu sổ ngược xuống (\) thì Linux dùng dấu chéo lên (/). Dấu chéo xuống được Linux dùng để nối tiếp một câu lệnh trên một dòng khác.

Và ngược lại với DOS, Linux quan tâm đến việc bạn phải dùng khoảng trắng khi xác định các tham số "." và ".." Linux không hiểu lệnh **cd...**, nhưng nếu bạn viết **cd ..** thì Linux sẽ hiểu. Nói tóm lại, Linux cần khoảng trắng ở giữa câu lệnh và tham số.

5.3.2.2 Liệt kê các tệp và thư mục bằng lệnh ls

ls viết tắt cho list (danh sách) và Linux dùng lệnh này để liệt kê danh sách tệp. Lệnh này cũng tương tự như DIR của DOS. (Linux chấp nhận lệnh **dir** để liệt kê danh sách tệp trong một thư mục). Lệnh **ls** của Linux liệt kê tất cả các tệp chính bằng màu sắc. Theo mặc định, màu xanh lơ biểu thị thư mục và xanh lục biểu thị các chương trình thi hành được. Muốn thay các màu mặc định, bạn chỉnh sửa tệp /etc/DIR_COLORS.

Xem "Liệt kê các tệp"

ls sử dụng nhiều tham số để thay đổi cách liệt kê tệp và loại tệp phải liệt kê. Tham số phổ biến nhất là –la, ra lệnh cho máy liệt kê thông tin của tệp theo dạng dài. Lệnh **ls**-la liệt kê tất cả thông tin của từng tệp trong thư mục hiện hành. Lệnh **ls**.emacs liệt kê các tệp bắt đầu bằng .emacs, trong khi **ls**-l.emacs liệt kê tất cả thông tin của các tệp bắt đầu bằng .emacs.

Tuỳ chọn –l**tar** (được dùng như là **ls**-ltar) liệt kê thông tin giống như lệnh **ls** vừa kể, có khác chăng là các thông tin được trình bày theo thứ tự từ lâu nhất đến mới nhất.

5.3.2.3 Tạo thư mục mới bằng lệnh mkdir

Tương tự như lệnh MD của DOS, **mkdir** của Linux tạo thư mục mới và bạn cung cấp tên của thư mục ấy như ở thí dụ sau:

```
mkdir backup
```

Ghi chú: Trong trường hợp bạn quá quen với lệnh MD của DOS và không thích dùng **mkdir**, Linux có một cách để tạo bí danh cho tên các lệnh.

Xem "Đặt bí danh cho lệnh"

5.3.2.4 Xoá bỏ thư mục bằng lệnh rmdir

Lệnh **rmdir** xoá các thư mục Linux với điều kiện thư mục ấy phải rỗng. Thí dụ nếu thư mục /backup có thư mục thứ cấp thì lệnh:

```
rmdir /backup
```

không hoàn thành công tác được.

Lệnh:

```
rmdir /backup/lan_anh/*
```

xoá tất cả các tệp trong thư mục /backup/lan_anh và sau đó

```
rmdir/backup/lan_anh
```

mới có thể xoá bỏ thư mục /backup/lan_anh bây giờ đã rỗng.

Cẩn thận: Vì lệnh **rmdir** không thể xoá thư mục chưa rỗng, bạn có thể dùng tham số -r với lệnh **rm**. Thí dụ: **rm** –r * sẽ xoá sạch mọi thứ từ thư mục hiện hành và tất cả các thư mục thứ cấp bên dưới. Do đó bạn phải cẩn thận khi ra lệnh này, vì một khi đã xoá là không phục hồi được. Nhớ sao lưu trước.

5.3.3 Sử dụng các lệnh thao tác tệp

Linux xử lý tệp và thư mục cũng như nhau.

5.3.3.1 Chép các tệp bằng lệnh cp

Lệnh **cp** tương tự như copy của DOS. Bạn dùng lệnh này để chép một hoặc nhiều tệp từ thư mục này sang thư mục khác. Cú pháp của **cp** như sau:

```
cp tệp_nguồn tệp_ đích
```

Bạn thay thế hai tham số tệp_nguồn tệp_đích bằng tên hai tệp mà bạn chọn. Nếu muốn giữ nguyên tên tệp, hãy dùng thư mục sẽ chứa tệp thay vào chỗ tham số tệp_đích. Ở DOS, bạn có thể bỏ trống tệp_đích nếu chép về thư mục hiện hành.

Lệnh **cp** lananh1 lananh1.old sẽ chép tệp lananh1 sang một tệp sao lưu mang tên lananh1.old, trong khi lệnh **cp** ~/lananh1.old /backup/lan_anh sẽ chép tệp lananh1.old từ thư mục "nhà" sang thư mục /backup/lan_anh. (Ký tự ~ đại diện cho thư mục "nhà" của người sử dụng).

5.3.3.2 Chuyển tệp bằng lệnh mv

Lệnh này tương tự như MOVE của DOS để bạn di chuyển tệp từ thư mục này sang thư mục khác. Khi ra lệnh di chuyển tệp, nghĩa là bạn đã chép tệp ấy sang chỗ mới và xoá tệp ở chỗ cũ. Lệnh **mv** không sao chép tệp.

Cú pháp của lệnh **mv** giống như lệnh **cp**:

```
mv tệp_nguồn tệp_đích
```

Lệnh **mv** lananh1 lananh1.old chép tệp lananh1 sang một tệp sao lưu mang tên lananh1.old, sau đó huỷ tệp lananh1 cũ, trong khi lệnh **mv** ~/lananh1.old /backup/lan_anh di chuyển tệp lananh1.old từ thư mục "nhà" sang thư mục /backup/lan_anh.

5.3.3.3 Xoá tệp bằng lệnh rm

Lệnh này nguy hiểm bởi vì khi đã huỷ thì bạn không thể khôi phục tệp được. Do đó để an toàn, bạn nên sử dụng hình thức sau đây của lệnh **rm**:

```
rm –i tên tệp
```

Tham số -i bắt máy phải hỏi lại người sử dụng xem có thực sự muốn xoá bỏ tệp hay không. Thí dụ lệnh **rm** lananh1 sẽ xoá tệp lananh1, trong khi **rm** –i lananh1 sẽ mời bạn khẳng định việc huỷ tệp.

Cẩn thận: Với Linux, một khi tệp bị huỷ thì coi như mất luôn chứ không thể lấy lại được (undelete) như với DOS. Khi xoá một tệp, hy vọng duy nhất của bạn là bản sao lưu của tệp ấy.

5.3.3.4 Hiển thị nội dung tệp bằng lệnh more

Lệnh **more** hiển thị tệp văn bản qua từng màn hình một. Bạn có thể xem một tệp văn bản mà không nhất thiết phải dùng đến phần mềm soạn thảo (edit), không cần in tệp ấy ra và cũng không phải tạm dừng thiết bị cuối (terminal) trong khi thiết bị ấy hiển thị tệp. Thí dụ, muốn hiển thị nội dung tệp .emacs, bạn gõ lệnh như sau:

```
more .emacs
```

Trong các phiên bản trước 6.0 của RH Linux, lệnh **more** khá bất tiện khi không cho bạn trở lui xem lại một trang màn hình đã qua. Tuy nhiên các tuỳ chọn dưới đây mới được bổ sung từ phiên bản 6.x cho phép trở lui k trang màn hình (mặc định là 1 trang): b hay <Ctrl-b>. Lệnh **less** sau đây có thể giải quyết vấn đề này tiện lợi hơn.

Ghi chú: Nếu dùng **more** với một tệp dữ liệu nhị phân, bạn có thể bực mình vì chẳng hạn máy sẽ bị treo. Trong trường hợp này, bạn thoát ra bằng cách bấm <Ctrl-q> hoặc <Ctrl-s>.

5.3.3.5 Sử dụng lệnh less

less hiển thị từng màn hình một. Giống như **more, less** có thể hiển thị màn hình thông tin của một tệp văn bản, song điểm khác biệt là bạn có thể di chuyển tới lui. Bạn thử dùng lệnh sau đây để duyệt tệp readme trong thư mục /info:

```
less /info/readme
```

Ghi chú: ở DOS, bạn dùng lệnh CLS để xoá màn hình ; ở Linux, đó là lệnh **clear.**

5.4 Xử lý các tệp DOS trong Linux

Trong khi cài đặt, bạn đã có dịp khai báo các partition (phân vùng) DOS để cho Linux hiểu được. Các partition này được đặt ở một thư mục mà bạn đã chọn trong khi thiết lập cấu hình, thí dụ như /dosc.

Nếu dùng lệnh **cp** để chép các tệp ấy sang đĩa mềm, bạn có thể gặp rắc rối vì UNIX và Linux xử lý tệp văn bản khác với DOS, nhất là khi phải xử lý phím xuống dòng. Mặt khác khi muốn tương tác với đĩa mềm, nếu dùng lệnh **cp**, bạn cần phải **mount** (lắp ghép) đĩa mềm trước và sau khi thực hiện lệnh sao chép **cp** xong, bạn còn phải **unmount** (tháo gỡ) đĩa mềm đó để lấy đĩa mềm ra. Để giải quyết vấn đề này, người ta đã soạn thảo một nhóm chương trình để xử lý các tệp DOS chạy ở môi trường UNIX. Đó là các câu lệnh thuộc họ m-, trong đó có những lệnh như **mcopy** và **mdir**. Lệnh **mcopy** hoạt động như lệnh COPY của DOS và **mdir** liệt kê thư mục. Như bạn đã nhận thấy, các lệnh này giống như đồng loại của chúng ở DOS, chỉ khác là chúng bắt đầu bằng ký tự m, do đó mới gọi là họ lệnh m-. Họ lệnh m- là thành phần của phần mềm chương trình **mtools**, vốn là bộ sưu tập các chương trình công cộng (phân phối tự do miễn phí) giúp cho UNIX tương tác với các tệp DOS dễ dàng hơn.

Các câu lệnh nói trên cũng giúp bạn chép tệp sang đĩa mềm dễ dàng hơn. vì lúc ấy bạn có thể sử dụng cách gọi tên của DOS, chẳng hạn như ổ đĩa A, thay vì phải gọi theo Linux là /dev/fd0. Muốn biết thêm chi tiết về họ lệnh m-, bạn gõ:

```
man mtools
```

Tuy nhiên trong lúc cài đặt, bạn cần chọn phần mềm "DOS emulation" thì mới sử dụng được **mtools**.

Lệnh	**Mô tả**
mattrib	Hiển thị thuộc tính của tệp được lựa chọn
mcd	Chuyển sang thư mục được nêu ở đường dẫn
mcopy	Chép các tệp sang đường dẫn được nêu
mdel	Xoá bỏ các tệp được chọn
mdir	Liệt kê thư mục
mformat	Format đĩa mềm
mlabel	Đặt tên nhãn cho file system DOS
mmd	Tạo thư mục mới
mrd	Xoá bỏ một thư mục (thư mục này phải rỗng, như đối với DOS)
mren	Đặt lại tên cho tệp DOS
mtype	Hiển thị nội dung văn bản của tệp DOS

Bảng 5.4: Các lệnh họ m-

5.5 Đóng tắt Linux

Khi bạn dùng máy PC chạy với DOS, làm việc xong bạn chỉ cần tắt công tắc điện. Nhưng với Windows làm như thế sẽ tổn hại đến các tệp.

Với Linux, mức độ rủi ro này còn cao hơn nữa, đặc biệt sẽ tổn hại cả phần cứng lẫn hệ thống tệp. Bạn phải đóng tắt Linux theo quy định, nếu không muốn lần làm việc tiếp theo sẽ gặp khó khăn khi khởi động máy.

Trước khi ghi dữ liệu vào đĩa cứng, Linux lưu rất nhiều thông tin về chính mình và thông tin của những tệp đang nằm trong bộ nhớ, tại những vùng đệm, còn gọi là bộ nhớ trung gian (buffer). Tiến trình này giúp cải thiện năng suất của cả hệ thống, đồng thời kiểm soát việc sử dụng phần cứng. Đây là việc mà các hệ điều hành đa nhiệm phải làm để một người sử dụng này không thể sử dụng một thiết bị nào đó đang phục vụ một người sử dụng khác. Nếu đột ngột tắt máy, các thông tin vừa kể sẽ mất đi và hệ thống tệp bị hỏng.

Xem "Khởi động và đóng tắt"

Là một hệ điều hành đa nhiệm và dành cho nhiều người sử dụng, Linux phải đảm bảo rằng từng thành viên phải ngưng phiên làm việc đúng quy cách và lưu lại công việc đang tiến hành trước khi đóng tắt cả hệ, như thế sẽ tránh được trường hợp mất dữ liệu và hỏng hệ thống tệp. Điều này cũng giúp cho những người khác đang cùng sử dụng

hệ Linux này có đủ thời gian đăng xuất. Muốn đóng tắt Linux cho đúng, bạn phải dùng lệnh **shutdown** với cú pháp như sau:

```
shutdown [-r] thời_gian_đóng_tắt [lời nhắn]
```

Tuỳ chọn –r có nghĩa là Linux phải lập tức khởi động lại sau khi đóng tắt. Điều này có ích khi bạn muốn thoát khỏi Linux để khởi động một hệ điều hành khác.

Thời_gian_đóng_tắt báo là tham số báo cho hệ điều hành biết khi nào có thể đóng tắt. Thời gian được tính theo hình thức 24 giờ, thí dụ nếu muốn hệ điều hành đóng tắt vào lúc 11 giờ đêm, bạn cần gõ lệnh:

```
shutdown 23:00
```

Tham số [lời nhắn] là thông báo chung cho tất cả những người sử dụng đang ở trong mạng. Từng người sẽ thấy lời nhắn này trên màn hình của mình.

Thí dụ khi muốn ngưng sử dụng máy để thực hiện công tác sao lưu hàng tuần, bạn gõ lệnh như sau để mọi người lo mà đăng xuất:

```
[root@web] /root] # shutdown –r 23:00 Đóng tắt hệ thống vào
lúc 11:00 pm để bảo trì hệ thống.
```

Lưu ý: Trên một số hệ thống đôi khi Linux có thể hiểu được nhóm phím khởi động lại <Ctrl-Alt-Del> và sẽ thực hiện thao tác đóng tắt đúng quy trình như khi bạn gõ lệnh **shutdown**. Tuy nhiên ở một số hệ thống khác, Linux không hiểu được nhóm phím ấy.

Nếu lỡ tắt hệ thống không đúng cách và làm hỏng kết cấu tệp, bạn có thể dùng lệnh **fsck** để thử sửa lại hệ thống tệp.

*Xem "Sử dụng lệnh **fsck**"*

5.6 Chạy các chương trình Linux

Khi đã quen các thao tác mở, tắt trong Linux và vài lệnh cơ bản, bạn có thể bắt đầu thử một số ứng dụng đã cài đặt khi thiết lập hệ thống. Những ứng dụng này bao gồm các tiện ích, từ một cái máy tính bỏ túi đơn giản cho đến những bộ biên dịch C và C++. Một vài ứng dụng như thế có giá trị lớn ; may thay nhờ vào giấy phép GNU, nhiều ứng dụng đã trở thành miễn phí.

Nhiều chương trình khác dành cho Linux cũng hiện diện miễn phí trên Internet và bạn có thể lấy được chúng nhờ vào một chương trình tải nạp đi kèm trong bản phát hành Slackware và RedHat. Ngoài ra nhiều cửa hàng có thể cung cấp cho bạn những đĩa CD-ROM với hàng trăm chương trình UNIX dưới dạng mã nguồn. Bạn có thể chọn lọc vài chương trình yêu thích từ CD-ROM rồi dùng các chương trình **gcc** và **g++** để biên dịch các chương trình ấy.

Những chương trình này chủ yếu làm việc ở chế độ văn bản, do đó không cần phải chạy hệ X Window.

5.6.1 Sử dụng chương trình CD Player

CD Player là một chươngr trình chơi nhạc từ đĩa CD, được cài đặt sẵn trong bản phát hành RedHat Linux. Bạn nên thử nó một lần xem, nếu máy bạn có một ổ CD-ROM

chấp nhận đĩa CD audio. Thực tế có thể CD Player không tương thích với tất cả các loại ổ CD-ROM có trên thị trường.

Chương trình này cho phép dùng bàn phím điều khiển CD, vì vậy bạn nhớ bật bàn phím sang chế độ <Num Lock>.

5.6.2 Sử dụng Gnumeric và KSpread

Trước khi phần mềm VisiCalc ra đời, các nhà lập kế hoạch phải dùng những tờ giấy kẻ ô gọi là spreadsheet. Là phiên bản điện tử của tờ spreadsheet, VisiCalc đã làm một cuộc cách mạng ở khâu tính toán và lập kế hoạch.

Ngày nay các chương trình như Microsoft Excel hoặc Lotus 1-2-3 đang tiếp nối truyền thống của VisiCalc. Trong Linux, Gnumeric và KSpread cũng làm chức năng đó.

Minh hoạ 5.6: Trình bảng tính Gnumeric

Gnumeric và KSpread đều có dạng bảng tính (spreadsheet calculator) gồm nhiều dòng và cột. Mỗi ô chứa một giá trị số học, một chuỗi ký tự, hoặc một biểu thức. Các chuỗi ký tự có thể dựa vào những ô khác để lập thành nhiều mối liên hệ phức hợp.

Minh hoạ 5.7: Trình bảng tính KSpread

Nếu đã từng làm việc với các chương trình bảng tính khác, bạn sẽ không gặp khó khăn với các lệnh của Gnumeric và KSpread.

5.6.3 Sử dụng bc Calculator

bc là một chương trình tính toán theo câu lệnh vì bản thân nó có một ngôn ngữ lập trình cho phép bạn thực hiện tương tác.

Sau khi bạn gõ lệnh, bc hiển thị vài dòng lưu ý về bản quyền tác giả và dấu nhắc sẽ nhấp nháy chờ lệnh. Bạn có thể ra lệnh làm hai phép tính cộng và trừ. Bạn cũng có thể ra lệnh nhân và chia, song phiên bản bc phát hành kèm với RH Linux lại xén bớt kết quả hai phép tính này.

bc rất tiện lợi cho các phép tính đơn giản. Một điều tiện lợi nữa là bc có khả năng lưu giá trị từ một phép tính này cho phép tính sau chỉ bằng một cú pháp đơn giản, đó là tên-của-biến = biểu-thức.

Thí dụ sau đây tính giá trị của 125*5, sau đó lưu kết quả vào biến var1. Kết quả phép tính được hiển thị bằng cách gõ tên của biến var1 và sẽ in giá trị (625) ở hàng kế tiếp. Thí dụ này còn lập biến var2 làm nơi chứa kết quả (125) của var1 chia cho 5.

```
var1 = 125*5

var1

625

var2 = var1/5

var2
125
```

5.6.4 Sử dụng chương trình minicom

Bạn có thể nối kết với thế giới nếu bạn có một modem và một phần mềm viễn thông. Linux cung cấp một phần mềm mang tên **minicom** và bạn chỉ cần nối modem với cổng nối tiếp COM trên máy PC.

Cũng giống như nhiều phần mềm Linux khác, **minicom** do một người viết ra nhưng được nhiều người khác trên Internet giúp đỡ hoàn chỉnh và phần mềm này chạy rất tốt, có khả năng cạnh tranh với nhiều ứng dụng thương mại khác. Để biết thêm các chức năng chi tiết của **minicom**, bạn xem trợ giúp bằng lệnh **man.**

Điều đầu tiên nên ghi nhớ là **minicom** dùng nhóm phím <Ctr-Shift-a> cho một số chức năng, chẳng hạn như auto-dial (tự động quay số điện thoại) và file downloading (tải tệp xuống). Đang ở trong **minicom**, nếu cần trợ giúp bạn bấm <Ctrl-a><z> để hiển thị màn hình tóm tắt các câu lệnh.

Phím	**Mô tả**
D	Truy cập thư mục quay số gọi
S	Gửi tệp đi
P	Liệt kê các tham số liên lạc
L	Bật/tắt việc ghi biên bản phiên làm việc vào một tệp
F	Gửi tín hiệu BREAK cho thiết bị bên kia đầu dây
T	Phỏng tạo terminal giữa các chế độ vt100, Minix, hoặc ANSI
W	Bật/tắt chế độ làm tròn dòng
G	Chạy một tệp script của **minicom**
R	Nhận một tệp
A	Thêm vào cuối dòng một ký tự chuyển dòng
H	Dừng liên lạc đường điện thoại
M	Khởi động modem
K	Chạy giao thức kermit
E	Bật/tắt chế độ bán song công
C	Xoá màn hình tại chỗ
O	Giúp bạn lập cấu hình **minicom**
J	Nhảy tắt đến một command shell khác
X	Thoát khỏi modem và khởi động lại modem
I	Chế độ cursor key
Z	Hiển thị màn hình trợ giúp
B	Cuốn ngược trở về cửa sổ terminal

*Bảng 5.6: Tóm tắt lệnh **minicom***

Khi đang ở trong cửa sổ trợ giúp, bạn chỉ việc gõ ký tự tương ứng để thi hành lệnh. Tuy nhiên nếu đang ở chương trình **minicom**, bạn phải bấm <Ctrl-a> trước ký tự đã chọn. **minicom** cho phép dùng bốn giao thức chuyển tệp: zmodem, ymodem, xmodem và kermit. Bạn nên thử chạy zmodem trước hết vì khả năng vượt trội của giao thức này về việc phục hồi khi có lỗi. Nếu các thiết bị cuối mà bạn định liên lạc lại không có zmodem, bạn nên thử dùng giao thức theo thứ tự vừa được kê ra. Điều này không có nghĩa rằng kermit là dở. Kermit không dở, nhưng chỉ chậm hơn các giao thức khác. Điểm mạnh của kermit là hầu hết các hệ thống khác đều chấp nhận giao thức này.

Điều thứ hai phải ghi nhớ khi sử dụng **minicom** là phần mềm viễn thông này có vài câu lệnh đặc biệt nguy hiểm. Đó là những câu lệnh giúp người sử dụng có quyền hạn như một superuser. Do vậy ai đó khi chạy **minicom** sẽ làm được một vài việc mà biết đâu bạn không muốn người ấy làm.

Xem "An ninh tệp"

5.7 Chạy các chương trình DOS trong Linux

Sau khi chạy các ứng dụng Linux một thời gian, có thể bạn sẽ muốn chạy vài chương trình DOS hoặc Windows. Một số phần mềm như DOSEMU và Wine cho phép làm việc này bằng cách mô phỏng nhiều hệ điều hành khác nhau.

DOSEMU giúp các chương trình dựa vào DOS (và những biến thể khác như PC-DOS) chạy được với Linux. DOSEMU có nghĩa là Mô phỏng DOS (DOS EMUlator). Còn phần mềm Wine thì mô phỏng Windows và sẽ được giới thiệu ở mục "Chạy các chương trình Windows với Linux".

Ghi chú: Một vài bản phát hành Linux có lệnh "simply dos" để khởi động một chương trình chỉnh sửa theo chế độ DOS. Những bản phát hành mang tính thương mại như RedHat Linux hoặc tương tự đều có lệnh này.

5.7.1 Cài đặt DOSEMU

Phần mềm DOSEMU (phiên bản thường dùng là 1.0.2) được lưu ở dạng nén trong thư mục /contrib/ cuả đĩa CD-ROM với tên dosemu_1.0.2.tgz hoặc dosemu_1.0.2.tar**.** Bạn sao tệp này vào thư mục /usr/src rồi bung ra bằng những lệnh giải nén thích hợp như sau đây:

```
[root@web src] # gzip – d dosemu_1.0.2.tgz
[root@web src] # tar – xvf dosemu_1.0.2.tar
```

Sau đó bạn tạo ra một số tệp bằng những lệnh khác:

```
[root@web src] # make config
[root@web src] # make depend
[root@web src] # make most
```

Từ đó các tệp DOSEMU sẽ hiện diện trong thư mục /var/lib/dosemu. Để thực hiện thao tác vừa qua, bạn phải đăng nhập như là root và máy bạn phải còn trống ít nhất 10 MB bộ nhớ ảo.

Ghi chú: Máy bạn phải được cài đặt gói phần mềm Development, một số công cụ và trình biên dịch để xây dựng ứng dụng mô phỏng DOS.

Bạn cũng có thể dùng dạng RPM của phiên bản này, gọi là dosemu_1.0.1-1.i386.rpm. Khi đó bạn chỉ cần gõ lệnh sau:

```
[root@web src] # rpm –i dosemu_1.0.1-1.i386.rpm
```

5.7.2 Lập cấu hình DOSEMU

DOSEMU được xây dựng để chạy một số lệnh DOS và chương trình DOS trong môi trường DOS phỏng tạo.

Sau khi xây dựng xong phần mô phỏng, bạn phải lập cấu hình cho hệ thống. Đầu tiên bạn tạo ra một đĩa mềm DOS khởi động, sau đó chép các tệp DOS sau đây vào đĩa: command.com, **fdisk**.exe và sys.com.

Tiếp theo, bạn chép các tệp DOSEMU từ thư mục dosemu vào đĩa: emufs.sys, ems.sys, cdrom.sys và exitemu.com. Bạn có thể dùng các lệnh –m được đề cập trước đây tại mục "Xử lý các tệp DOS với Linux" để sao chép các tệp từ phân vùng Linux sang đĩa mềm.

Ghi chú: Nếu khó tìm các tệp Linux, bạn hãy sử dụng lệnh find, thí dụ như lệnh sau đây sẽ hiển thị vị trí tệp ấy trên hệ thống máy của bạn:

```
find / -name emufs.sys-print
```

DOSEMU cần phải có tệp cấu hình dosemu.conf và global.conf để chạy được hoàn hảo. Tệp global.conf là tệp cấu hình chính của dosemu luôn được thực thi khi khởi động dosemu. Tệp dosemu.conf là tệp chứa các giá trị cho những cấu hình đã khai báo trong global.conf. Có thể sửa chữa chúng thông qua công cụ 'setup-dosemu'.

Bạn phải tuỳ chỉnh tệp dosemu.conf cho hợp với hệ thống máy mình nhưng không nên chỉnh sửa gì ở tệp global.conf. Để có một ý niệm ban đầu, bạn xem các tệp mẫu như danh sách 5.1 dưới đây hiển thị toàn bộ dosemu.conf và global.conf trong bộ cài đặt. Những nhận xét được đánh dấu bằng dấu thăng (pound #) và hầu hết các tuỳ chọn đều mang hình thức giá trị tham số. Nếu một tham số nào đó có hơn một giá trị, thì các giá trị ấy được đặt trong ngoặc móc ({}).

Các giá trị sẽ được xem là một chuỗi (string) hay một giá trị số (numeric) hoặc logic (boolean) tuỳ theo việc chúng được đặt giữa 2 dấu nháy kép hay giữa 2 ngoặc đơn.

Danh sách 5.1: Nội dung tệp dosemu.conf và global.conf điển hình

```
DOSEMU.CONF
#########################################################
# This file is /etc/dosemu.conf. included by <DOSEMU-LIB-DIR>/global.conf
#
# Linux DOSEMU configuration for Parser versions - 3 (dosemu-0.97.0.1)
#
# ./doc/README.txt (chapter 2.) contains a description of the syntax
# and the usage of dosemu.conf.
```

```
#
# Access rights are defined in
#
#     /etc/dosemu.users
#
############################################################
#
# Notes for editing this section:
#     In    $_xxx = (n) n is a numerical or boolean value
#                    =       =
#     In    $-zzz = "s" s is a string
#
# Please edit only bateen the brakets and quotes and keep the rest intact.
#     ^^^^^^^^^^^^^^^^^^^^^^  ^^^^^^^^^^^
$_debug -   "-a"  # same format as -D commandline option
      # (but without the -D in front)
$_features= ""    # list of temporary hacks, see release notes in
      # the file ChangeLog. e.g "0:1 2:0", which means
      # to set feature - 0 to 1 and feature-2 to 0.
$_mapping= ""     # prefered mapping driver, one of:
      # auto, mapself, mapfile, mapshm
$_ timint = (on)  # emulate INT08 type timer interrupts
$_mthco = (on)    # or off
$_cpu = "80386"   # CPU showing to DOS, valid values: "80[345]86"
      # or "emulated" for non-native CPU (386 in this case)
$ rdtsc = (on)    # if possible use Pentium cycle counter
$_cpuspeed = (0)# 0 = calibrated by dosemu, else given (e.g.166.666)
$_pci = (off)
$ xms = (1024) # in Kbyte
$_ems = (2048) # in Kbyte
$ _ems frame = (0xe000)
-     7
$_dpmi = (off)    # in Kbyte
$_ dosmem = (640) # in Kbyte, < 640
$_hardware_ram = "" # list of segment values/ranges such as
      # "0xc8000 range 0xcc000.0xcffff"
$-secure ="ngd"   # secure for: n (normal users), g (guest), d (dexe)
      # empty string: depending on 'restricted'
      # "0": always insecure (not recommended)
$_ odd hosts = -4 black list such as "lucifer.hell.com billy.the.cat"
```

```
$_diskless_hosts="" # black list such as 'hacker1 newbee gateway1''
$_ emusys = # empty or 3 char., config.sys      - config.XXX
$_emubat =  # empty or 3 char., autoexec.bat    - autoexec.XXX
$_emuini =  # empty or 3 char.. systern.ini     - system.XXX
$_hogthreshold = (1) # 0 == all CPU power to DOSEMU
$_irqpassing = "" # list of IRQ number (2-15) to pass to DOS such as
      # "3 8 10"
$_speaker = ""    # or "native" or "emulated"
$_term_char_set   ""    # Global code page and character set selection.
      #     automatic, else: ibm. latin, latin1, latin2
$_ - term - color (on) # terminal with color support
$_ - term_updfreq = (4) # time between refreshs (units: 20 == 1 second)
$_escchar = (30)# 30 == Ctrl-^, special -sequence prefix
$_rawkeyboard = (0) # bypass normal keyboard input, maybe dangerous
$_layout = "auto" # one of: finnish(-latin1). de(-latin1). be. it. us
      # uk. dk(-Iatin1), keyb-no, no-latin1. dvorak, po
      # sg(-latin1), fr(-latin1), sf(-latin1), es(-latin1)
      # sw, hu(-Iatin2), hu-cwi. keyb-user
      # hr-cp852, hr-latin2. cz-qwerty. cz-qwertz.
      # Or 'auto' (which tries to generate the table from
      # the current Linux console settings)
$_keybint = (on)# emulate PCish keyboard interrupt
$_X_updfreq = (5) # time between refreshs (units: 20 == 1 second)
$_X_title = "DOS in a BOX"# Title in the top bar of the window
$_X_icon_name = "xdos" # Text for icon, when minimized
$_X_keycode = (auto)    # on == translate keyboard via dosemu keytables
      # or 'off' or 'auto'
$_X_blinkrate = (8) # blink rate for the cursor
$_X_font = "" # basename from /usr/X11R6/lib/X11/fonts/misc/*
      # (without extension) e.g. ''vga''
$_X_mitshm = (on) # Use shared memory extensions
$_X_sharecmap = (off) # share the colormap with other applications
$_X_fixed-aspect = (on)# Set fixed aspect for resize the graphics window
$_X_aspect_43 = (on)    # Always use an aspect ratio of 4:3 for graphics
$_X_lin_filt = (off)    # Use linear filtering for >15 bpp interpolation
$_X_bilin_filt = (off) # Use bi-linear filtering for >15 bpp interpolation
$_X_model3fact = (2)    # initial size factor for video mode 0x13 (320x2OO)
$_X_winsize = ""# "x,y" of initial windows size (defaults to float)
$_X_gamma = (1.0) # gamma correction
$_X_vgaemu_memsize = (1024)
```

```
      # size (in Kbytes) of the frame buffer for emulated vga
$_X_lfb = (on) # use linear frame buffer in VESA modes
$_X_pm_interface = (on)# use protected mode interface for VESA modes
$_X_mgrab - key "Home" # KeySym name to activate mouse grab. empty == off
$_X_vesamode ""   # "xres,yres ... xres,yres"
      # List of vesamodes to add. The list has to contain
      # SPACE separated "xres.yres" pairs
$_video = vga"    # one of: plainvga. vga, ega. mda, mga. cga
$-console   (0)   # use 'console' video
$_graphics (0)# use the cards BIOS to set graphics
$_videoportaccess = (1)# allow videoportaccess when 'graphics' enabled
$_vbios_seg = (OxcOOO) # set the address of your VBIOS (e.g. OxeOOO)
$_vbios_size = (OxIO000)# set the size of your BIOS (e.g. MON)
$_vmemsize (1024) # size of regen buffer
$_chipset = ""    # one of: plainvga, trident, et4000, diamond, avance
      # cirrus, matrox. wdvga, paradise. ati, s3, sis
$_dualmon = (0)   # if you have one vga_plus_one hgc (2 monitors)
$_vbootfloppy = "" # if you want to boot from a virtual floppy:
      # file name of the floppy image under DOSEMU_LIB_DIR
      # e.g. "floppyimage" disables $_hdimage
      #     "floppyimage +hd" does - not _ disable $_hdimage
$_floppy_a ="threeinch"# or "fiveinch" or "atapi" or empty, if not existing
      # optionally the device may be appended such as
      # "threeinch:/dev/fdO"
$_floppy_b = ""         # dito for B:
$_hdimage = "freedos" # list of hdimages under DOSEMU_LIB_DIR
      # assigned in this order such as
      # "hdimage_c hdimage_d hdimage_e"
      # If the name begins with '/dev/', then partion
      # access is done instead of virtual hdimage such as
      # "/dev/hdal" or "/dev/hdal:ro" for readonly
      # Currently mounted devices and swap are refused.
      # Hdimages and devices may be mixed such as
      # "hdimage_c /dev/hdal /dev/hda3:ro"
      # Note: 'wholedisk' is - not_ supported.
$_hdimage_r = $_hdimage# hdimages for 'restricted access (if different)
$_aspi = " "      # list of generic SCSI devices to make available
# for the builtin aspi driver (format of an entry
# is 'device:type:mappedtarget' such as
# "sg2:WORM sg3:Sequential_Access:6 sg4:CD-ROM" or
```

```
# "sg2:4 sg3:1:6 sg4:5" (which are equal)
$_com1 = " "      # e.g. "/dev/mouse" or  "/dev/ttySO"
$_com2 = " "      # e.g. "dev/modem" or   "/dev/ttyS1"
$_com3 = " "      # dito      "/dev/ttyS2"
$_com4 = " "      # dito      "/dev/ttyS3"
$_ttylocks = " "        # Lock directory (e.g. "/var/lock")
# default (" ") is /usr/spool/uucp
$-mouse = " "     # one of: microsoft, mousesystems, logitech, mmseries
      # mouseman, hitachi, busmouse, ps2, jmps2
$_mouse_dev = " " # one of: com1, com2, com3, com4 or /dev/mouse
$_mouse_flags = " " # list of none or one or more of:
      # "emulate3buttons cleardtr"
$_mouse_baud = (0) # baudrate, 0 == don't set
$_printer = "lp"# list of /etc/printcap) printer names to appear as
      # LPT1, LPT2, LPT3 (not all are needed, empty for none)
$_printer_timeout = (20)# idle time in seconds before spooling out
$_ports = " "     # list of portnumbers such as "Ox1ce Ox1cf Ox238"
      # or "Ox1ce range Ox28O,Ox29f 310"
      # or "range Ox1aO,(Ox1aO+15)"
$_ipxsupport = (off)    # or on
$_novell_hack  = (off)
$_vnet = (off)    # 'on' for packet-multi (used by dosnet)
$_sound =   (off) # sound support on/off
$_sb_base   =     (Ox220)
$_sb_irq = (5)
$_sb_dma =  (1)
$_sb_dsp =  "/dev/dsp"
$_sb_mixer = "/dev/mixer"
$_mpu_base = "Ox330"
GLOBAL.CONF
#########################################################
# This file is global.conf, it is -allways_ executed as part of the
inititial
# configuration.
#
# Linux dosemu configuration for parser versions - 3 (dosemu-0.97.0.1)
#
# ./doc/README-tech.txt (chapter 2.) contains a description of the syntax
# and the usage. However, you normal won't edit this file
#
```

```
# NOTES:
#
# 1.  The file dosemu.conf (and optionally -/.dosemurc) contains variable
#     settings, that are included by global.conf for doing the
#     most reasonable configuration.
#     The file dosemu.conf (and optionally -/.dosemurc) is what gets
#     updated by the menu driven 'setup-dosemu' tool.
#
# 2.  We strongly recommend you to edit ONLY dosemu.conf.
#     If you change global.conf, you are at your own and could break
#     'setup-dosemu'. You really need to know a lot of DOSEMU
#     internals before you fiddle with editing global.conf.
#     However, some very special cases can only be handled in global.conf.
#
# 3. The file global.conf (this one) must either
#
#     -     be in DOSEMU LIB DIR (the default, changeable via dosemu.users
#           or --Flibdir)
#     -     or given via -F option (from root login, else only on
#           non-suid-root DOSEMU)
#     -     or be in $HOME/,dosemu/lib.
#
#     For the latter there must be given the following conditions:
#     -     in dosemu.users give the keyword 'private_setup' for this user
#     -     DOSEMU -must- be non-suid-root. except for user root or if you
#     -     give the keyword 'unrestricted' in addition to 'private-setup'.
#
# 4.  The only compiled-in path is /etc/dosemu.users (or if not found
#     /etc/dosemu/dosemu.users). however, this can be overridden by
#     --Fusers, if not running suid-root. All other pathes are configurable
#     and the dosemu binaries can reside everywhere in the system.
#
# This file (global.conf) may also serve as an example file for
#     dosrc ( old style user configuration file )
#     option -I   ( configuration via commandline, see man/dos.)
#
# Access rights for suid-root running DOSEMU sessions are defined in
#
#     /etc/dosemu.users
#
```

```
#####################################################
ifdef u_forceold
undef parser_version_3
define version_3_style_used
endif
ifndef parser_version_3
      # normally won't come here, because older DOSEMUs don't use this file
      # ... but if using -F option
      include "/etc/dosemu.conf"
else
      # we are on version 3 parser
# for security reason we restrict the PATH and restore it later
# to its original value
$_DOSEMU_ORIG_PATH = $PATH
$PATH = "/usr/local/bin:/usr/bin:/bin"
## we set some vital variable
if (!strlen($DOSEMU_LIB_DIR))
# be sure, because older DOSEMU won't set this
$DOSEMU_LIB_DIR = "/var/lib/dosemu";
$DOSEMU_CONF_DIR = "/etc";
$DOSEMU_HDIMAGE_DIR = $DOSEMU_LIB_DIR;
else
if (!strlen($DOSEMU_HDIMAGE_DIR))
$DOSEMU_HDIMAGE_DIR = $DOSEMU_LIB_DIR;
endif
# make sure we have absolute paths
$DOSEMU_LIB_DIR = shell("cd ", $DOSEMU_LIB_DIR, pwd -P");
$DOSEMU_LIB_DIR = strdel($DOSEMU_LIB_DIR, strlen($DOSEMU_LIB_DIR)-1, 1);
$DOSEMU_HDIMAGE_DIR = shell("cd ", $DOSEMU_HDIMAGE_DIR, "; pwd -P");
$DOSEMU_HDIMAGE_DIR              =              strdel($DOSEMU_HDIMAGE_DIR,
strlen($DOSEMU_HDIMAGE_DIR)-1, 1);
endif
$CONFIG_VERSION =( (0 << 24) | (97 << 16) | (2 << 8) | 0)
#                        ^     ^^    ^     ^
if ( $DOSEMU_VERSION_CODE < $CONFIG_VERSION
abort "
***sorry. your ", $DOSEMU_LIB_DIR, "/global.conf doesn't match this dosemu
version"
endif
$DOSEMU_VERSION =($DOSEMU_VERSION_CODE >> 24), '.',
(($DOSEMU_VERSION_CODE>>16) & 255), '.',
```

```
((($DOSEMU_VERSION_CODE>>8) & 255), '.',
($DOSEMU_VERSION_CODE & 255);
$LIST_DELIM = "," #delimiters for lists <tab>, blank, comma
# we include the global settings from /etc/dosemu.conf shell("test -f ",
$DOSEMU_CONF_DIR.,"/dosemu.conf")
if ( $DOSEMU_SHELL_RETURN)
abort "
***error: ", $DOSEMU_CONF_DIR, "/dosemu.conf not accessable ... giving up.
"
endif
      undef version_3_style_used    # unset it to have a valid check below
      $xxx = 'include "', $DOSEMU_CONF_DIR, '/dosemu.conf"';
      $$xxx;
      ifndef version_3_style_used
            abort "
***Your ", $DOSEMU_CONF_DIR, "/dosemu.conf is obviously an old style or a
too simple one.
            Please read Quickstart and README.txt on how to upgrade.
"
endif
## we check if we have an ${HOME}/. dosemurc (given by $DOSEMU_RC),
## if yes, we include it here 'under user scope', which means it has
## all c_* classes undefined and can only change enviroment variables in
## its own isolated name space (prefixed with 'dosemu_').
## We later take over only such variables, that we allow.
$_dosemurc = $DOSEMU_RC # we get that passed from 'parse-config()'
shell("test -f ", $ dosemurc)
if ( (!$DOSEMU_SHELL_RETURN) && ($DOSEMU_VERSION_CODE >= ( ((97<<16) |
(4<<8)      | 3))))
      # we have a .dosemurc and can include
      $_dosemurc = ' include " ', $_dosemurc, ' " ' ;
      enter_user_scope;
      $$_dosemurc;
      leave_user_scope;
define skip_dosrc:
## now we take over the allowed variables, only those which exist
## with 'dosemu_' prefixed will overwrite those without the prefix.
checkuservar $_debug,
      $_features, $_mapping,
      $_timint, $_mathco, $_cpu, $_pci, $_xms, $_ems, $_ems_frame,
      $_emusys, $_emubat. $_emuini
```

```
checkuservar
      $_term_char_set, $_term_color. $_term_updfreq, $_escchar, $_layout,
      $_keybint
checkuservar
      $_X_updfreq, $_X_title, $_X_icon_name, $_X_keycode, $_X_blinkrate,
      $_X_font, $_X_mitshm, $_X_sharecmap, $_X_fixed_aspect, $_X_aspect_43,
      $_X_Iin_filt, $_X_bilin_filt, $_X_mode13fact. $_X_winsize, $_X_gamma,
      $_X_vgaemu_memsize,    $_X_lfb,    $_X_pm_interface,    $_X_mgrab_key,
$_X_vesamode
ifdef guest
define restricted
      else
            checkuservar
            $_speaker,
            $_com1,   $_com2.   $_com3.   $_com4.   $_mouse,   $_mouse_dev.
$_mouse_flags,
$_mouse_baud,
            $_printer, $_printer_timeout, $_ports,
            $_ipxsupport. $_novell_hack, $_vnet,
            $_sound, $_sb_base, $_sb_irq, $_sb_dma, $_sb_dsp, $_sb_mixer.
$_mpu_base
      endif
ifndef restricted
      checkuservar   $_hogthreshold,   $_dpmi,   $_vbootfloppy,   $_aspi,
$_ttylocks
      $xxx = $ hdimage # remember the contents for later check
      checkuservar $_hdimage
      if ($xxx ne $_hdimage)
            # the user changed it in .dosemurc, have to check
            if (strlen($_hdimage) && ($DOSEMU_UID != $DOSEMU_EUID)
                  # we are running suid-root as user, don't allow partition
access
                  $yyy = $_hdimage
                  foreach $xxx ($LIST_DELIM, $yyy)
                        if (!strncmp($xxx, "/dev/", 4))
                              abort "
***sorry, your are not allowed to use ", $xxx, "in dosemurc
"
endif
done
endif
endif
```

```
endif
endif
## end of .dosemurc inclusion
if ( $DOSEMU_VERSION_CODE >= ( (99<<16) | (13<<8) | 2) )
      if (strlen($_mapping))
            mappingdriver $_mapping;
      endif
endif
if (strlen($_debug))
      debug $_debug ;
else
      debug {off}
endif
if ($_cpuspeed) cpuspeed $_cpuspeed endif
if ($_rdtsc) rdtsc on else rdtsc off endif
pci $_pci
dosbanner on
timint $_timint
mathco $_mathco
if (strlen($_cpu))
$_cpu = "cpu ", $_cpu;
$$_cpu
else
cpu 80386
endif
xms $_xms
if ($_ems) ems {ems_size $_ems ems_frame $_ems_frame} else ems off endif
dosmem $_dosmem
if ($_emusys ne " ")    emusys $_emusys endif
if ($_emubat ne " ")    emubat $_emubat endif
if ($_emuini ne " ")    emuini $_emuini endif
$DISABLE_MOUSE = (0)
$DISABLE_SERIAL = (0)
## terminal stuff, we check a lot to insure proper operation
$BEING_ON = " "
$USING_X = (0)
$REMOTE_HOST = " "
## Note GNU sh-utils>=2.0 changed the default behaviour such
#     that it doesn't display the peer for remote connections anymore.
#     We need -1 option to get this. As GNU sh-utils 1.16 doesn't
```

```
#     have this option, we detect this case via shell return value.
      $xxx = shell("who -1 am i 2>/dev/null")
      if ($DOSEMU_SHELL_RETURN)
            # we have GNU sh-utils < 2.0
            $xxx = shell("who am i")
      endif
      if ( ($DOSEMU_STDIN_IS_CONSOLE eq "1")
                        || ( strlen($xxx) && (strchr($xxx, "(" ) < 0 )))
            $BEING_ON = "console"
      else
            if (strstr($xxx, "(:") < 0)
                  $BEING_ON = "remote"
                  $REMOTE_HOST = strsplit($xxx. strchr($xxx,"(")+1,99)
      $REMOTE_HOST = strdel($REMOTE_HOST, strchr($REMOTE_HOST,")"),99)
else
      $BEING_ON = " "
endif
if ( (strchr($DOSEMU_OPTIONS,"X") >=0) && ($DISPLAY ne " ") )
      $USING_X = (1)
endif
if (strstr("xterm dtterm", $TERM) >=0)
      $BEING_ON = $BEING_ON, "_xterm"
else
      if (strstr("linux console", $TERM) < 0)
            # remote side must be some type of real terminal
$BEING_ON = $BEING_ON, "_terminal"
else
# remote side is a Linux console
      $BEING_ON = $BEING_ON. "_console"
endif
endif
endif
warn "dosemu running on ", $BEING_ON;
$xxx = $_term_char_set
if    ($xxx eq "")
      $xxx = "latin"
      if (strstr($BEING_ON, "_console") >=0)
            $xxx = "ibm"
      endif
endif
```

```
if    ( ($BEING_ON ne "console") && (!$USING_X))
$DISABLE_MOUSE = (1)
if ($_term_color) $_term_color = "on" else $_term_color "off" endif
terminal {charset $$xxx updatefreq $_term_updfreq escchar $_escchar
                  color $$_term_color
endif
if ( $DOSEMU_VERSION_CODE >= ( (97<<16) | (10<<8) | 5) )
      charset $$xxx
endif
## host checking. Naah, we won't let all in ;-)
## first: logged in hosts
if (strlen($REMOTE_HOST) && strlen($_odd_hosts))
foreach $xxx ($LIST_DELIM, $_odd_hosts)
if ( ! strncmp($xxx, $REMOTE_HOST. strlen($REMOTE_HOST)))
abort "Sorry, your host is on the blacklist"
endif
      done
endif
## second: hosts on diskless machines, that aren't allowed
if (strlen($_diskless_hosts))
      foreach $xxx ($LIST_DELIM, $_diskless_hosts)
if ( ! strncmp($xxx, $DOSEMU_HOST, strlen($DOSEMU_HOST)))
      abort "Sorry, your host is on the blacklist"
endif
done
endif
## X param settings
if ( $USING_X || (strchr($DOSEMU_OPTIONS."L") >0) )
      # running as xdos or DEXE (which may force X later)
      if ($_X_keycode == -1)  # we do not test for 'auto', because older
            # DOSEMU don't know it and we want older
            # versions also running with this global.conf
$_X_keycode = "keycode_auto"
else
if ($_X_keycode) $_X_keycode = "keycode" else $_X_keycode = " " endif
endif
if ($_X_sharecmap) $_X_sharecmap = "sharecmap" else $_X_sharecmap = ""
endif
if ($_X_aspect_43) $_X_aspect_43 = "aspect_43" else $_X_aspect_43 =     "  "
endif
if ($_X_lin_filt) $_X_Iin_filt = "lin_filt" else $_X_Iin_filt = " " endif
```

```
if ($_X_bilin_filt) $_X_bilin_filt = "bilin_filt" else $_X_bjlin_filt = " "
endif
      if ($ X mode13fact != 2)
            $_X_mode13fact = "mode13fact ", $_X_mode13fact
      else
            $_X_mode13fact = " "
      endif
      $_X_gamma = "gamma ",   (int($_X_gamma * 100))
      if (strlen($_X_font)) $_X_font = 'font " ', $_X_font, ' " ' endif
if (strlen($_X_winsize))
$Xxx = (strstr($_X_winsize,","))
            $_X_winsize = "winsize (", strdel($_X_wlnsize,$xxx,999), "), ("
                        strsplit($_X_wjnsize,$xxx+1,999), ")"
      endif
      $xxx = $_X_vesamode
      $_X_vesamode = ""
      if (strlen($xxx))
            foreach $yyy (" ", $xxx)
                  $zzz = (strchr($yyy,","))
                  $_X_vesamode   =   "vesamode   (",   strdel($yyy,$zzz.999),
"),(",
                              strsplit($yyy,$zzz+1,999), ") "
            done
      endif
      if (strlen($_X_mgrab_key))
            $_X_mgrab_key = 'mgrab_key " ', $_X_mgrab_key, ' " '
      endif
      X {
            title $_X_title icon_name $_X_icon_name
            updatefreq $_X_updfreq blinkrate $_X_blinkrate
fixed_aspect $_X_fixed_aspect vgaemu_memsize $_X_vgaemu_memsize
lfb $_X_lfb pm_interface $_X_pm_jnterface mitshm $_X_mitshm
$$_x keycode $$_X_sharecmap $$_X_aspect_43 $$_X_Iin_filt $$_X_bilin_filt
            $$_X_font $$_X_winsize $$_X_gamma
            $$_X vesamode $$_X_mode13fact $$_X_mgrab_key
}:
endif
ifdef guest
      ## /etc/dosemu.users defined 'guest' for this login
      define restricted # force restricted setting
      $_hogthreshold = (1)     # force 'nice'
```

```
      $_rawkeyboard = (off) # force normal keyboard
      if ($BEING_ON eq "console") video { vga } endif
      sound emu off
      $DISABLE_MOUSE = (1)
      $DISABLE_SERIAL = (1)
else
      ## other then guest
      # video settings
      if ($BEING_ON eq "console")
            if (!strlen($_video)) $_video = "plainvga" endif
            if ($_console) $_console = "console" else $_console = " " endif
            if ($_graphics)
                  $_graphics = "graphics"
                  $_console = "console"
                  $_vbios_seg = "vbios_seg ", ($_vbios_seg)
                  $_vbios_size = "vbios_size ", ($_vbios_size)
                  $_vmemsize = "memsize ", ($_vmemsize)
      if ($_videoportaccess) allowideoportaccess on endif
      else
      $_graphics = " "; $_vbios_seg = " "; $_vbios_size = " "; $_vmemsize =
" "
      endif
if (strlen($_chipset)) $_chipset = "chipset ",$_chipset endif
      if ($_dualmon) $_dualmon = "dualmon" else $_dualmon = " " endif
      video {
            $$_video $$_console $$_graphics $$_chipset
            $$_vbios_seg $$_vbios_size $$_vmemsize $$_dualmon
}
else
      if ($_dualmon) video {dualmon} endif
endif
## sound settings
if ($_sound)
sound_emu { sb_base $_sb_base sb_irq $_sb_irq sb_dma $_sb_dma
sb_mixer $_sb_mixer mpu_base $_mpu_base }
else
sound_emu off
endif
## serial
if (($DISABLE_SERIAL) && strlen($ ttylocks))
```

```
      ttylocks { directory $_ttylocks }
endif
if ((!$DISABLE_SERIAL) && strlen($_com1))
      if ($_mouse_dev eq "com1")
if (!$DISABLE_MOUSE) serial { mouse com 1 device $_Com1 } endif
$_mouse = " "
else
      if ($_com1 eq "virtual")
serial      { com 1 virtual }
else
serial { com 1 device $_com1 }
endif
endif
endif
if ((!$DISABLE_SERIAL) && strlen($_com2))
if ($_mouse_dev eq "com2")
if (!$DISABLE_MOUSE) serial { mouse com 2 device $_com2 } endif
$_mouse = " "
else
if ($_com2 eq "virtual")
serial { com 2 virtual }
else
serial { com 2 device $_com2 }
endif
            endif
      endif
      if ((!$DISABLE_SERIAL) && strlen($_com3))
            if ($_mouse_dev eq "com3")
                  if (!$DISABLE_MOUSE) serial { mouse com 3 device $_com3 }
endif
                  $_mouse = " "
            else
                  if ($_com3 eq "virtual")
serial      { com 3 virtual }
else
serial { com 3 device $_com3 }
endif
endif
endif
if ((!$DISABLE_SERIAL) && strlen($_com4))
```

```
      if ($_mouse_dev eq "com4")
if (!$DISABLE_MOUSE) serial { mouse com 4 device $_com4 } endif
$_mouse = " "
else
      if ($_com4 eq "virtual")
serial      { com 4 virtual }
else
serial { com 4 device $_com4 }
            endif
      endif
endif
## mouse settings
if ((!$DISABLE_MOUSE) && strlen($_mouse) && (strstr($_mouse_dev,"com")<O))
      if ($_mouse_baud)
      $_mouse_baud = "baudrate ", $_mouse_baud
else
$ mouse baud = " "
endif
if (strlen($_mouse_dev))
$_mouse_dev = "device ", $_mouse_dev, " internaldriver"
endif
mouse { $$_mouse $$_mouse_dev $$_mouse_flags $$_mouse_baud }
endif
endif # end 'not guest'
hogthreshold $_hogthreshold
## keyboard setting
If ($BEING_ON ne "console") $_rawkeyboard = (off) endif
if ($_layout eq "auto")
      keyboard { layout auto keybint $_keybint rawkeyboard $_rawkeyboard }
else
      if ( strstr($_layout, "load") <0 )
            # we use the builtin keytables
            if (strlen($_layout)) $_layout = "layout ".$_layout
            else $_layout = "layout us" endif
            keyboard    {    $$_layout    keybint    $_keybint    rawkeyboard
$_rawkeyboard }
      else
            # we have to load a keytable
            $yyy = " "
            foreach $xxx ($LIST_DELIM, $_layout)
```

```
                  if ($xxx ne "load")
                        $yyy = $xxx
                  endif
            done
            if (!strlen($yyy))
                  abort "no keytable name in $_layout"
            endif
            shell("test -f ", $DOSEMU_LIB_DIR, "/keymap/", $yyy)
            if ( $DOSEMU_SHELL_RETURN)
                  abort "keytable ",$yyy, "not found in, $DOSEMU_LIB_DIR,
"/keymap/*"
            endif
            $_layout 'include "keymap/', $yyy, ' " ' ;
            $$_layout
            keyboard { keybint $_keybint rawkeyboard $_rawkeyboard }
      endif
endif
secure off
$_disks = " "
ifdef restricted
      ## /etc/dosemu.users defined 'restricted' for this login
      if (!strlen($_secure)) secure on endif
      define c_normal
      if (strchr($_secure,"d")>=0) dexe { secure } endif
      ifdef guest
            if (strchr($_secure,"g")>=0) secure on endif
      else
            if (strchr($_secure,"n")>=0) secure on endif
            $_disks = $_hdimage_r
      endif
      if (strchr($_secure,"0")>=0) secure off endif
      dpmi off
      speaker emulated
      ipxsupport off
      # printer ( options "%s" command "lpr" timeout 20 }
else
      ## /etc/dosemu.users does allow full access for this login
dexe { allowdisk }
$_disks = $_hdimage
if (!strlen($_vbootfloppy))
```

```
      bootC       # default boot mode is 'from C:'
else
      bootA       # default boot mode is 'from A:'
      $yyy = " "; $zzz = " ";
foreach $xxx ($LIST_DELIM, $_vbootfloppy)
      if (strlen($yyy))
            $zzz = $xxx
      else
            $yyy = $xxx
endif
      done
      $_vbootfloppy = $DOSEMU_LIB_DIR, "/", $yyy
bootdisk { heads 2 sectors 18 tracks 80 threeinch file $_vbootfloppy }
      if (strlen($zzz))
$_disks = $_hdimage
      endif
endif
if (strlen($_floppy_a))
      $zzz = strsplit($_floppy_a, strstr($_floppy_a, ":"), 999):
      if (strlen($zzz))
$zzz = strsplit($zzz, 1, 999)
$_floppy_a = strdel($_floppy_a, strstr($_floppy_a, ":"), 999);
else
      $zzz = "/dev/fd0"
endif
$xxx = shell("test -r ". $zzz);
if ($DOSEMU_SHELL_RETURN)
warn "*** Warning: floppy ", $zzz, not accessable, disabled";
else
floppy { device $$zzz $$_floppy_a }
endif
endif
if (strlen($_floppy_b))
$zzz = strsplit($_floppy_b, strstr($_floppy_b, ":"), 999);
if (strlen($zzz))
$zzz = strsplit($zzz, 1, 999)
      $_floppy_b = strdel($_floppy_b, strstr($_floppy_b, ":"). 999);
else
      $zzz = "/dev/fd1"
endif
```

```
$xxx = shell("test -r ", $zzz);
if ($DOSEMU_SHELL_RETURN)
warn "**** Warning: floppy ", $zzz, " not accessable, disabled";
else
floppy { device $$zzz $$_floppy_b }
            endif
      endif
      # misc stuff
      dpmi $_dpmi
      if (strlen($_irqpassing))
            $yyy = "irqpassing { "
            foreach $xxx (" ", $_irqpassing)
                  $yyy = $yyy, "use_sigio ", $xxx
            done
            $yyy = $yyy, " } ";
            $$yyy
      else
            irqpassing off
      endif
      if (strlen($_hardware_ram))
            hardware_ram ( $$_hardware_ram }
      endif
      if (strlen($_speaker))
            $_speaker = "speaker ", $_speaker;
            $$_speaker
      else
            speaker off
      endif
      ipxsupport $_jpxsupport
      if ($_novell_hack) pktdriver novell_hack endif
      vnet $_vnet
      if (strlen($_printer))
            foreach $xxx ($LIST_DELIM, $_printer)
                  $xxx = " ' -p". $xxx, " %s' ";
                  printer   {   options   $$xxx   command   "lpr"   timeout
$_printer_timeout }
            done
      endif
      if (strlen($_ports)) ports { $$_ports } endif
endif
```

```
## setting up hdimages
$_mounted_devices = " "
if (strlen($_disks) && (!defined(c_dexerun)))
      foreach $xxxx ($LIST_DELIM, $_disks)
            $xu_pref = " "
            if ((strchr($xxxx, "*") +  1)&&(strlen($xxxx)  ==  strchr($xxxx,
"*")+ 1))
$xxx_pref = strdel($xxxx, strlen($xxxx) -1, 1);
$xxxx = shell("cd '/var/lib/dosemu/", $xxx_pref, " ' 2>/dev/null && echo -n
*")
if ($DOSEMU_SHELL_RETURN)
      $xxxx = strcat($xxx_pref, "*")
      $xxx_pref = " "
            endif
      endif
      foreach $xxx ($LIST_DELIM, $xxxx)
      $xxx = strcat($xxx_pref. $xxx)
      if (!strncmp($xxx, "/dev/", 4))
            if (!strlen($_mounted_devices))
                  # we first get all mounted devices
$_mounted_devices = shell(" df I awk '/\dev\//{print $1}' tr '\n' ' ' ");
      # we try to get the swap device too
$_mounted_devices = $_mounted_devices, shell("awk '!/^\#/   && /\/dev/ &&
/swap/{print $1}' /etc/fstab | tr '\n' ' ' ");
      endif
      $yyy = strdel($xxx, strstr($xxx, ":ro"), 999);
      $zzz = strsplit($xxx, strstr($xxx, ":ro"), 999);
      $yyyy = strcat($yyy. ' ');
      if (strstr($_mounted_devices, $yyyy) <0)
                  # device not in use
            if (strlen($zzz))
      disk { partition $yyy readonly }
else
disk { partition $yyy };
            endif
      else
            abort  "***  device  ",  $yyy,  "  in  use,  cannot  do  partition
access"
      endif
else
      $yyy = strdel($xxx, strstr($xxx, ":ro"), 999);
```

```
      $zzz = strsplit($xxx, strstr($xxx, ":ro"), 999);
      $yyy = $DOSEMU_HDIMAGE_DIR, "/", $yyy
      shell("test -d '", $yyy, "'")
      if (!$DOSEMU_SHELL_RETURN)
            if (strlen($zzz))
      disk { directory $yyy readonly };
else
disk { directory $yyy };
 endif
else
disk { image $yyy }
endif
endif
done
done
endif
## setting up ASPI devices
ifndef restricted
if (strlen($_aspi))
foreach $xxx ($LIST_DELIM, $_aspi)
$zz = (1);
$yy2 ""; $yy3 = (-1);
foreach $yyy (":", $xxx)
      $zzz = "$yy", $zz, " = $yyy";
      $zz = ($zz + 1);
      $$zzz
done;
      aspi { $yy1 devicetype $yy2 target $yy3 };
done
      endif
endif
## setting up the features list
if ( ($DOSEMU_VERSION_CODE >= ((98<<16) | (3<<8) | 3))
&& ($DOSEMU_VERSION_CODE < (99<<16)) )
      || ($DOSEMU_VERSION_CODE > ((99 << 16)    (5<< 8))) )
if (strlen($ features))
      foreach $xxx ($LIST_DELIM, $_features)
            $yyy = strdel($xxx, strstr($xxx, ":"), 999);
                  $zzz = strsplit($xxx, strstr($xxx, ":"), 999);
                  if (strlen($zzz))
```

```
                        $zzz = strsplit($zzz, 1, 999);
                  else
                        $zzz = (0);
                  endif
                  feature { $yyy = $zzz };
            done
      endif
endif
# ... here we are and need disallow 'shell' any further
# so we can restore PATH
$PATH = $_DOSEMU_ORIG_PATH
## define the allowed classes for subsequent .dosrc and -I parsing ifdef
restricted undef c-all
ifdef guest
undef c_normal
define c_dexe
define c_nice
define c_x
endif
endif
endif
####################################
```

Sau đó bạn phải dùng một chương trình soạn thảo văn bản để thay đổi mọi thiết lập sẵn có trong tệp thí dụ sao cho phù hợp với hệ thống máy bạn. Một số thiết bị như bộ xử lý card video cũng phải tương thích.

Ghi chú: Bạn cũng có thể khởi động DOSEMU từ một phân vùng (partition) trên ổ đĩa cứng thay vì từ đĩa mềm. Muốn truy cập ổ đĩa cứng, bạn chỉ cần lập cấu hình cho một ổ đĩa cứng hoặc một phân vùng trong tệp dosemu.conf.

5.7.3 Chạy DOSEMU

Muốn chạy DOSEMU bạn chỉ cần gõ lệnh tại dấu nhắc Linux. Muốn thoát ra, bạn gõ exitemu. Bảng 5.9 liệt kê các tuỳ chọn từ dòng lệnh cho DOSEMU. Bạn dùng tùy chọn -? để hiển thị toàn bộ các tham số.

Bảng 5.9 Các tham số DOSEMU tại dòng lệnh

Tham số	Mô tả
-A	Khởi động từ ổ đĩa mềm A
-C	Khởi động từ ổ đĩa cứng
-c	Tối ưu hoá hiệu năng video từ các terminal ảo
-D	Lập các tuỳ chọn gỡ lỗi

-e	Xác định bộ nhớ EMS
-F#	Xác định số lượng (#) đĩa mềm để sử dụng từ dosemu.conf
-f	Hoán chuyển định nghĩa của các ổ đĩa mềm A và B
-H#	Xác định số lượng (#) đĩa cứng để sử dụng từ dosemu.conf
-k	Sử dụng bàn phím được định nghĩa bằng các tham số rawkeyboard trong tệp dosemu.conf
-p	Chép các thông số gỡ lỗi vào một tệp
-t	Phát ngắt thời gian 9
-V	Kích hoạt mô phỏng màn hình VGA
-x	Xác định bộ nhớ XMS
-?	Hiển thị trợ giúp tóm tắt cho từng lệnh
-2	Mô phỏng máy 286
-3	Mô phỏng máy 386
-4	Mô phỏng máy 486

Từ dấu nhắc DOS của DOSEMU, bạn có thể chạy hầu hết các chương trình DOS ngoại trừ chương trình nào đòi hỏi DPMI (DOS Protected Mode Interface: Giao diện theo chế độ DOS có bảo vệ). Bạn chỉ cần gõ tên của chương trình, sau đó DOSEMU sẽ căn cứ theo đường dẫn mà bạn đã cung cấp để nạp và chạy, với điều kiện là DOSEMU tìm được chương trình ấy.

Chạy chương trình với DOSEMU có nhiều rắc rối, đa phần bởi vì máy đang mô phỏng DOS thay vì chạy DOS thật. Việc mô phỏng sẽ làm giảm tốc độ hệ thống và tốc độ sẽ đặc biệt chậm nếu máy phải chạy các chương trình Linux khác ở các terminal.

Nhiều chương trình của DOS sẽ chiếm bộ xử lý CPU, khiến các chương trình Linux khác khó mà chen vào. Để giải toả bớt vấn đề, người ta đã viết một chương trình mang tên garrot giúp cho Linux có điều kiện chiếm CPU. Bạn có thể tải garrot xuống từ website FTP mang tên sunsiste.unc.edu tại thư mục /pub/linux/alpha/dosemu.

5.8 Chạy các chương trình Windows với Linux

Vì DOSEMU không thể thực hiện các chương trình của Windows nên người ta đã làm ra phần mềm mô phỏng gọi là Wine. Wine viết tắt từ WINdows Emulator, hoặc từ Wine Is Not a Windows Emulator. Cả hai lối viết tắt này đều được giải thích trong mục FAQ của Windows (FAQ: những câu thường xuyên được hỏi).

Nếu muốn dùng thử Wine, bạn nên đọc qua Windows FAQ. Wine chưa được triển khai rộng như DOSEMU, chính vì thế mà nó còn bỏ sót nhiều lỗi và nhiều chương trình Windows cũng chưa được nó chấp nhận. Thật ra muốn dùng Wine bạn phải cài đặt Windows trên một phân vùng nào đó mà Linux truy cập được, bởi vì Wine còn phụ thuộc nhiều vào Windows. Wine cũng đòi hỏi hệ thống đồ họa X WINdow phải được cài đặt trước.

Muốn chạy thử Wine, bạn cần những thứ sau đây:

- Bản kernel của Linux, phiên bản 0.99.13 hoặc mới hơn (thí dụ 2.4.18-3).
- Mã nguồn của Wine, bởi vì Wine chỉ có ở dạng này.
- Ít nhất 16 MB bộ nhớ RAM, nếu chạy với 64 MB hoặc nhiều hơn thì càng tốt.
- X Window đã cài đặt và thiết lập cấu hình xong.
- Một thiết bị điều khiển con chạy (cursor) trên màn hình, chẳng hạn như chuột.
- Microsoft Windows cài đặt ở phân vùng nào mà Linux có thể truy cập được.

Vì Wine đang được tiếp tục phát triển cho nên có thể còn có phiên bản mới. Bạn nên coi thư mục /pub/Linux/ALPHA/wine/developement, ở địa chỉ sunsite.unc.edu để có những thông tin cập nhật. Tệp sẽ được đặt tên tuỳ theo ngày phát hành, chẳng hạn như Wine-20020509.tar.gz. Muốn tìm hiểu thêm về Wine, bạn có thể tải các tệp FAQ và HOWTO mới nhất xuống rồi đọc chúng. Những tệp này sẽ giúp bạn biên soạn, cài đặt, thiết lập cấu hình và sử dụng Wine.

Cài đặt Wine giống với cài đặt DOSEMU, chỉ khác là bạn có thể đặt tệp nén ở bất cứ thư mục nào. Bạn dùng lệnh **tar** để bung tệp ra như ở thí dụ sau:

```
[root@web wine] # gzip – d wine – 20020509.tar.gz
[root@web wine] # tar – xvf wine – 20020509.tar
```

Việc biên dịch mã nguồn của Wine yêu cầu tỉ mỉ hơn so với DOSEMU, tựa như biên dịch một kernel mới vậy. Bạn phải trả lời nhiều câu hỏi trong tiến trình xây dựng. HOWTO về Wine giải thích cặn kẽ việc này. Tiếp theo bạn sẽ cung cấp tham số về thời gian chạy máy. Những tham số này được lưu tại tệp /usr/local/etc/wine.conf (chú ý: tuỳ theo phiên bản của wine hay Linux mà vị trí này có thể thay đổi). Có thể chỉnh sửa tệp này bằng cách thủ công, song tốt hơn bạn nên dùng chương trình cấu hình kèm theo để thực hiện. Sau khi lập xong cấu hình cho các tệp biên dịch và tệp tham số thời gian chạy máy, bạn chỉ cần ra lệnh **make** để xây dựng Wine. Để bắt đầu sử dụng Wine, bạn gọi phần mô phỏng và cung cấp đường dẫn đến một tệp thi hành của Windows.

```
[lan_anh@web~] $wine /dosc/windows/winmine.exe
```

hay:

```
[lan_anh@web~] $wine C:\\dosc\\windows\\winmine.exe
```

Chương 6. Nâng cấp và cài đặt phần mềm với RPM

Các phần mềm dùng cho Linux có thể được nâng cấp hoặc tạo mới. Việc cài đặt phần mềm có thể là thủ công (tự biên dịch từ các tệp nguồn) hay bằng RPM. Trong chương này chúng ta sẽ cùng nghiên cứu về RPM với các nội dung chính như sau:

- *Các thuật ngữ liên quan*
- *Chính sách nâng cấp phần mềm*
- *Cài đặt phần mềm*
- *Sử dụng RPM*
- *Nâng cấp kernel*

Hệ thống Linux lúc đầu chỉ chứa một số tiện ích và các tệp dữ liệu cơ bản. Sau đó tuỳ theo nhu cầu, quản trị viên hệ thống sẽ thêm vào những lệnh khác, cài đặt các chương trình ứng dụng cho user và bổ sung các tệp dữ liệu cần thiết. Các ứng dụng được cập nhật thường xuyên và phần mềm hệ thống cũng thay đổi nếu có những tính năng mới hoặc sửa được các lỗi. Quản trị viên hệ thống chịu trách nhiệm cài đặt thêm phần mềm, rồi lập cấu hình, duy trì hoặc xoá bỏ những phần mềm ấy.

Cài đặt có nghĩa là chép các tệp chương trình hữu quan vào ổ cứng của hệ thống và sau đó là lập cấu hình (nghĩa là phân bố tài nguyên) cho ứng dụng để hệ thống chạy suôn sẻ. Lập cấu hình cho một chương trình là cho chương trình biết xem phải cài đặt các thành phần của ứng dụng ở đâu và ứng dụng đó phải hoạt động trong môi trường hệ thống như thế nào.

Các bản phát hành của Red Hat và Caldera giúp bạn cài đặt và nâng cấp phần mềm một cách dễ dàng hơn bằng lệnh **rpm**. Tuy nhiên bạn cũng sẽ phải cài đặt những phần mềm không thuộc dạng **rpm**. Thông thường nhiều gói phần mềm sẵn có trên Internet được công bố ở dạng nén **tar.**

Trên hệ thống lớn, quản trị viên phải cài đặt nhiều ứng dụng khác nhau vì hầu hết các user không có quyền truy cập băng từ hoặc đĩa mềm. Ngoài ra khi user muốn cài đặt các thành phần của ứng dụng vào thư mục hệ thống thì cũng phải được sự đồng ý của quản trị viên. Những thành phần ấy có thể là các thư viện, tiện ích và thiết bị khác được dùng chung mà khi muốn truy cập thì phải vào một số thư mục đặc biệt mà một số user bình thường không được phép.

6.1 Các thuật ngữ liên quan

Dưới đây là một số thuật ngữ và định nghĩa liên quan đến việc cài đặt các phần mềm.

Thuật ngữ	**Định nghĩa**
superuser	User có quyền ưu tiên cao nhất, cũng được gọi là root user.
Quản trị viên hệ thống	Người chịu trách nhiệm tối ưu hoá hệ điều hành và giúp hệ thống này chạy tốt. Quản trị viên hệ thống được các ưu tiên của superuser và có quyền cài đặt phần mềm mới vào hệ thống.

Cài đặt ứng dụng	Việc cài đặt ban đầu hoặc cập nhật chương trình cho hệ thống. Việc này thường đòi hỏi bạn phải là superuser để có quyền truy cập băng từ và đĩa mềm của hệ thống.
Lập cấu hình	Tiến trình tạo điều kiện cho một ứng dụng làm việc hài hoà với hệ thống. Lập cấu hình có thể bao gồm việc sắp xếp cho n hiều user dùng chung một ứng dụng, làm cho ứng dụng ấy chạy được từ nhiều thư mục khác nhau, hoặc chia sẻ ứng dụng với toàn mạng.

Bảng 6.1: Các thuật ngữ liên quan đến việc cài đặt ứng dụng

6.2. Chính sách nâng cấp phần mềm

Nên nâng cấp những phần mềm nào và bao lâu thì thực hiện một lần? Câu trả lời tuỳ thuộc vào mục đích sử dụng của hệ thống – cá nhân hay cơ quan – và tuỳ theo yêu cầu của các user. Phần mềm thường thay đổi phiên bản nhanh chóng, chưa kể việc nhiều thành phần khác nhau của Linux được cập nhật từ mọi nơi. Do đó bạn sẽ không kịp sử dụng hệ thống một cách thành thạo nếu cứ gắng sức chạy theo những bản nâng cấp liên tục ra đời.

Mỗi khi nâng cấp phần mềm cho hệ thống, bạn không nhất thiết phải cài đặt lại toàn bộ Linux. Thông thường chỉ một phần nhỏ của phần mềm hệ thống phải thay đổi khi cài đặt phiên bản nâng cấp. Có thể bạn sẽ phải nâng cấp phần kernel hoặc thư viện hệ thống. Tuy nhiên khi nâng cấp gói phần mềm ứng dụng thì bạn phải cài đặt một phiên bản hoàn toàn mới.

Nhìn chung, bạn nên nâng cấp hệ thống nếu phiên bản mới của hệ thống hoặc của ứng dụng có thể giải quyết được những vấn đề quan trọng, hoặc tăng thêm chức năng nào đó mà bạn cần có. Vấn đề như thế nào được gọi là quan trọng thì tuỳ bạn quyết định.

Ghi chú: Nên sao lưu hệ thống hiện hành trước khi nâng cấp phần mềm, đề phòng trường hợp hỏng hóc bạn vẫn có thể trở lại sử dụng hệ thống cũ.

6.3 Cài đặt phần mềm

6.3.1 Giới thiệu

Cài đặt các chương trình trọng yếu vào hệ thống Linux thường phức tạp hơn là cài vào một hệ điều hành đơn nhiệm như DOS. Bản chất multiuser của Linux cho phép mỗi ứng dụng trên hệ thống cùng lúc thoả mãn được yêu cầu truy cập từ nhiều phía khác nhau. Rắc rối hơn nữa, hầu hết các chương trình đều đòi hỏi phải được lập cấu hình cho hợp với hệ thống trước khi sử dụng. Do đó quản trị viên phải xác định từng mục cho hợp với cấu hình hệ thống trong tiến trình lập cấu hình.

Thí dụ user này dùng terminal loại cũ chạy ở chế độ văn bản, trong khi user khác đang sử dụng một thiết bị đời mới nhất với X Window. Lúc ấy superuser phải đảm bảo rằng ứng dụng đáp ứng được thiết bị cũ chỉ gửi đi các ký tự ASCII – nghĩa là chữ và số - trong khi thiết bị X Window phải nhận được đồ hoạ và màu sắc.

Cài đặt chương trình vào Linux phức tạp hơn, bởi vì quản trị viên phải tạo ra thư mục mới để chứa các tệp liên kết với chương trình mới. Một vài gói phần mềm mới yêu cầu lập lại cấu hình cho các thiết bị hệ thống.

Một user bình thường chỉ phải bỏ công tìm hiểu về các chức năng của ứng dụng mới và nhớ thêm vài câu lệnh mới, trong lúc đó quản trị viên có trách nhiệm đảm bảo rằng tài nguyên hệ thống phải được phân bổ, lập cấu hình và duy trì đúng đắn. Và đương nhiên ứng dụng mới không được xung đột với các chương trình sẵn có trên hệ thống.

Nhìn từ bên ngoài, việc cài đặt phần mềm bằng cách sử dụng menu và câu lệnh có vẻ đơn giản, song đối với hệ thống thì đó là công việc phức tạp. Những ứng dụng dành cho hệ điều hành một người sử dụng (chẳng hạn như DOS) thường chỉ chạy một mình mà không gặp cạnh tranh. Trên hệ thống Linux, ngay cả khi chỉ có một người đăng nhập, vẫn có nhiều tiến trình đang làm việc cùng lúc. Mức độ phức tạp sẽ tỷ lệ thuận với số người sử dụng, chưa kể đến việc nhiều người dùng một ứng dụng cùng lúc.

Sở trường của Linux là điều phối được nhiều tiến trình, nhiều chương trình, thiết bị và user cùng lúc. Muốn tồn tại trong môi trường chặt chẽ này, ứng dụng phải được nạp vào đúng cách, nếu không toàn bộ hệ thống sẽ treo, mọi chương trình sẽ ngưng và hàng loạt user sẽ bị thiệt thòi. Do đó khi nạp ứng dụng mới vào hệ thống, quản trị viên hoặc superuser phải thử nghiệm ứng dụng sau khi cài đặt và làm sao cho ứng dụng ấy thích hợp với cả hệ thống. Muốn hiểu tiến trình nạp phần mềm vào hệ thống Linux, bạn phải biết rõ trách nhiệm và quyền hạn của quản trị viên hệ thống là gì.

6.3.2 Công việc của quản trị viên hệ thống

Nếu sử dụng Linux trên hệ thống nhỏ, có khả năng bạn là quản trị viên hệ thống của chính bạn. Bạn chỉ cài đặt và chạy các chương trình của bạn. Trách nhiệm của bạn là sao lưu tệp, duy trì một khoảng trống thích hợp trên ổ cứng, quản lý bộ nhớ, cùng với một số việc khác đảm bảo cho hệ thống chạy hữu hiệu và có năng suất.

Quản trị viên của một hệ thống lớn phải có thêm những trách nhiệm như sau:

- Khởi động và đóng tắt hệ thống.

- Bảo đảm còn đủ khoảng trống trên ổ cứng và hệ thống tệp không bị lỗi.

- Bảo đảm số lượng tối đa các user truy cập được phần cứng và phần mềm hệ thống.

- Bảo vệ hệ thống chống lại các hành động xâm nhập bất hợp pháp và phá hoại.

- Thiết lập liên lạc với các hệ thống tin học khác.

- Tạo ra và xoá bỏ các trương khoản của hệ thống.

- Làm việc với các hãng cung cấp phần cứng và phần mềm, với các chuyên gia và những người có trách nhiệm hỗ trợ hệ thống.

- Cài đặt, lắp đặt và gỡ lỗi cho terminal, máy in, ổ đĩa, cùng với các cấu kiện khác.

- Cài đặt và duy trì phần mềm, kể cả các ứng dụng mới và bản cập nhật hệ điều hành.

Rất nhiều khi các user thích đăng nhập với tư cách là root và gõ đủ thứ lệnh, thậm chí họ có thể gây ra mọi loại vấn đề. Vì vậy quản trị viên cần dành quyền sử dụng trương

khoản root chỉ cho công việc quản trị. Còn để làm các công việc hàng ngày thì mỗi người chỉ nên dùng trương khoản cá nhân của mình mà thôi.

6.4 Sử dụng RPM

Hai bản phát hành Red Hat Linux và Caldera OpenLinux đều sử dụng gói phần mềm (Package) để quản lý việc cài đặt. Gói phần mềm là một chương trình đầy đủ đã được thử nghiệm và lập sẵn cấu hình cài đặt. Gói phần mềm được xây dựng từ các tệp mã nguồn mở, làm cho người sử dụng lẫn người triển khai đều biết được mình đang có cái gì trong tay. Để quản lý những phần mềm ấy, Red Hat Software đã phát triển công cụ RedHat Package Manager (RPM).

RPM làm việc theo sáu chế độ khác nhau, trong đó bạn có thể sử dụng năm chế độ từ dòng lệnh hoặc bằng gnorpm, một công cụ căn cứ trên X Window. Bạn cũng có thể sử dụng GnomeRPM (gnorpm) hay KDERPM. Các chế độ đó là cài đặt, tháo bỏ cài đặt, cập nhật hoá, tìm, kiểm sát và xây dựng. Bạn chỉ có thể xây dựng một gói phần mềm RPM từ dòng lệnh với cú pháp sau đây:

```
rpm [tuỳ chọn] tên_gói phần mềm
```

Với tuỳ chọn là một trong những flag (cờ hiệu) mà RPM dùng để thao tác các gói phần mềm và tên_gói phần mềm xác định tên của gói phần mềm được sử dụng. Tên gói phần mềm thường có dạng quota-1.55-4.i386.rpm. Tên gói phần mềm bao gồm những thành phần như sau:

tên	quota
phiên bản	1.55
ấn bản	4
cấu trúc máy tính	i386
phần mở rộng	.rpm

Tuy nhiên tệp của gói phần mềm mang tên gì cũng được bởi vì bản thân thông tin về gói phần mềm đã có sẵn bên trong tệp.

6.4.1 Vị trí của các gói phần mềm

Đa phần các gói phần mềm được cung cấp kèm theo bản phát hành này nằm ở thư mục /RedHat/RPMS trên CD-ROM. Muốn cài CD-ROM và liệt kê các gói phần mềm khác nhau, bạn dùng những lệnh sau:

```
cd /mnt
mount /mnt/cdrom
cd cdrom/RedHat /RPMS
ls | more
```

Hầu hết các gói phần mềm đã được mặc định cài đặt trong quá trình cài đặt Linux. Nếu lúc ấy bạn chọn không cài đặt và hiện giờ lại muốn cài đặt thì cứ tiến hành tuỳ ý.

RPM cũng giúp cài đặt các gói phần mềm nằm trên những máy tính khác bằng FTP mà bạn sẽ xem ở đoạn tiếp theo.

Xem "Cài đặt các thành tố của phần mềm"

6.4.2 Cài đặt gói phần mềm bằng RPM

Từ dòng lệnh, bạn dùng tuỳ chọn –i như sau:

```
rpm –i quota –1.55-4.i386.rpm
```

Lệnh này sẽ cài đặt gói phần mềm quota vào hệ thống máy bạn. Tuỳ chọn –i cho phép cài đặt gói phần mềm quota-1.55-4.i386.rpm vào hệ thống cục bộ. RPM thực hiện quá trình cài đặt qua các bước sau:

-Kiểm tra tính phụ thuộc. Mỗi gói phần mềm có khả năng phụ thuộc vào phần mềm khác đã cài sẵn trên máy.

-Kiểm tra tiềm năng xung đột. RPM xét xem một thành phần đã cài đặt sẵn hay chưa, hoặc thành phần ấy có cũ hơn thành phần đang trong quá trình cài đặt hay không.

-Xử lý các tệp cấu hình. RPM sẽ thử cung cấp một tệp cấu hình thích hợp và nếu phát hiện một tệp cấu hình có sẵn, RPM sẽ lưu tệp này lại để đối chiếu trong tương lai.

-Cài đặt tệp. RPM mở các gói phần mềm thành phần và đặt chúng vào thư mục thích hợp.

-Xử lý sau khi cài đặt. Sau khi cài đặt các thành phần xong rồi, RPM tiến hành các công việc cần thiết để lập cấu hình hệ thống cho đúng đắn.

-Cập nhật cơ sở dữ liệu. RPM ghi chép lại lộ trình của mình vào một cơ sở dữ liệu.

Trong khi cài đặt, câu lệnh nêu trên sẽ không thông báo phản hồi, song bạn có thể dùng tuỳ chọn –v (verbose) để lấy thông báo. Bảng 6.2 liệt kê các tuỳ chọn trong khi cài đặt.

Tuỳ chọn	Mô tả
-vv	Cung cấp thông tin đầy đủ.
-h	Thỉnh thoảng hiển thị dấu # trong khi cài đặt, nhờ vào đó biết rằng RPM đang thực sự làm việc chứ máy không bị treo.
--percent	Thay vì hiển thị dấu #, tuỳ chọn này sẽ hiển thị tỷ lệ phần trăm khối lượng công việc đã thực hiện trong quá trình cài đặt.
--test	Không cài đặt gói phần mềm, nhưng thử nghiệm cài đặt thử và báo cáo lỗi.
--replacefiles	Thay thế các tệp từ những gói phần mềm khác.
--force	Lệnh cho RPM không quan tâm đến các lỗi do xung đột và cứ tiếp tục cài đặt.

Bảng 6.2: Các tuỳ chọn khi cài đặt

Muốn cài đặt một gói phần mềm từ máy khác, bạn có thể dùng giao thức tải FTP với một địa chỉ URL để xác định gói phần mềm ấy.

```
rpm-i ftp://ftp.netwharf.com/pub/RPMS/quota-1.55-4.i386.rpm
```

Câu lệnh này giả sử rằng chiếc máy từ xa chấp nhận chế độ FTP vô danh.

Ghi chú: Bạn có thể cùng lúc nhập username và mật khẩu vào dòng lệnh như sau:

```
rpm –i ftp://mark@ftp.netwharf.com/pub/RPMS/quota-1.55-4.i386.rpm
Password for mark@ftp.netwharf.com: <enter your password here>
```

Tuy nhiên đây không phải là cách an toàn để nhập lệnh, bởi vì có thể ai đó sẽ nhìn trộm, hoặc truy lục lệnh này từ tệp ghi chép lịch trình các câu lệnh của bạn.

Xem "Sử dụng FTP bằng trình duyệt web"

6.4.3 Gỡ bỏ cài đặt gói phần mềm bằng RPM

Một trong những tiện lợi khi sử dụng RMP là việc cài đặt các chương trình mới rất dễ. Việc gỡ bỏ cài đặt cũng không có gì khó khăn. Bạn dùng tuỳ chọn –e như sau:

```
rpm –e quota –1.55.i386.rpm
```

Muốn gỡ bỏ một gói phần mềm ra khỏi hệ thống máy bạn, RPM phải qua các bước như sau:

-Kiểm tra tính phụ thuộc. RPM kiểm tra cơ sở dữ liệu xem có gói phần mềm nào khác phụ thuộc vào gói phần mềm ấy hay không. Nếu có, RPM sẽ không xoá, trừ khi bạn khẳng định là phải xoá.

-Chuẩn bị gỡ bỏ. RPM thi hành một script chuẩn bị cho việc gỡ bỏ cài đặt.

-Kiểm tra các tệp cấu hình. RPM lưu bản sao của mọi tệp cấu hình đã thay đổi.

-Xoá các tệp. RPM xoá tất cả các tệp kết hợp với gói phần mềm được xác định.

- Dọn dẹp. RPM thi hành một script dọn dẹp sau khi gỡ bỏ cài đặt.

-Cập nhật cơ sở dữ liệu. RPM gỡ bỏ tất cả mọi chỉ mục tham chiếu đến gói phần mềm đã tháo bỏ.

Cũng như đối với tuỳ chọn –i, bạn có thể sử dụng các tuỳ chọn –v và –vv để lấy thông báo đầy đủ từ lệnh erase. Bạn cũng có thể dùng tuỳ chọn –**test** để xem thử việc gì sẽ xảy ra khi gói phần mềm được gỡ bỏ cài đặt. Cuối cùng, tuỳ chọn –nodeps báo cho RPM cứ tiến hành gỡ bỏ cài đặt mà không cần quan tâm đến mọi phụ thuộc. Bạn nên cẩn thận với tuỳ chọn này, vì nếu bạn gỡ bỏ cài đặt một gói phần mềm đang có chương trình nào khác phụ thuộc vào nó thì sau này chương trình ấy sẽ không làm việc trơn tru được.

6.4.4 Cập nhật gói phần mềm bằng RPM

Với RPM, bạn được thuận lợi khi nâng cấp phần mềm bằng tuỳ chọn –U (viết hoa). Giả sử chương trình mang tên quota được một tác giả thêm vào nhiều chức năng và sau đó phổ biến lại với tên quota-1.55-4.i386.rpm. Muốn nâng cấp phiên bản cũ, bạn ra lệnh như sau:

```
rpm-U quota-1.55.4.i386.rpm
```

Trong khi nâng cấp, RPM cài đặt gói phần mềm đã xác định, sau đó xoá tất cả các phiên bản cũ. RPM cũng bỏ ra một khoảng thời gian khá lớn để xử lý các tệp kết hợp với gói phần mềm. Do đó trong khi RPM đang nâng cấp, có thể máy sẽ có thông báo như sau, cho bạn biết rằng một tệp cấu hình được lưu vào tệp mới:

```
Saving syslog.conf to syslog.conf.rpmsave
```

Điều này có nghĩa là RPM đã tạo ra một tệp cấu hình mới có khả năng tương thích với hệ thống máy bạn. Sau khi nâng cấp xong, bạn nên đối chiếu hai tệp cấu hình để chỉnh sửa tệp mới, nếu thấy cần thiết.

6.4.5 Tìm các gói phần mềm

Muốn biết những gói phần mềm nào đã cài đặt vào hệ thống, bạn dùng lệnh như sau:

```
rpm –qa
```

Lệnh sẽ hiển thị danh sách các gói phần mềm hiện có trên hệ thống. Muốn lấy thông tin từ một gói phần mềm nhất định, bạn cần gõ tuỳ chọn –q. Bảng sau liệt kê các tuỳ chọn mà bạn có thể sử dụng với nhóm lệnh **rpm** –q để tìm các gói phần mềm.

Tuỳ chọn	Mô tả
-q tên	Cung cấp tên, phiên bản và số phát hành của gói phần mềm.
-qa	Liệt kê tất cả các gói phần mềm đã cài đặt trên hệ.
-qf tệp	Tìm gói phần mềm liên kết với tệp.
-qp gói_phần_mềm	Tìm gói_phần_mềm.
-qi gói_phần_mềm	Cung cấp tên, mô tả, bản phát hành, kích cỡ, ngày tạo ra, ngày cài đặt và các thông tin khác về gói phần mềm.
-ql gói_phần_mềm	Liệt kê tất cả các tệp liên kết với gói phần mềm.

Bảng 6.3: Các tuỳ chọn tìm RPM

Cẩn thận: Các tuỳ chọn –q không chạy trơn tru khi bạn xác định một số kết nối tượng trưng (symbolic link). Để có kết quả tốt, bạn hãy chuyển đến thư mục thực sự của tệp trước khi sử dụng các tuỳ chọn –q.

Xem "Các mối liên kết"

Thí dụ bạn tìm thấy một gói phần mềm mới và muốn biết thêm thông tin, bạn gõ lệnh sau:

```
rpm –qip quota –1.55-4.i386.rpm
```

Lệnh sẽ hiển thị đại loại như:

```
Name: quota  Distribution: Manhattan
Version: 1.55 Vendor: Red Hat software
Release: 9 Build Date: Thu May 7 22:45:481998
Install date: (not installed) Build Host: porky.redhat.com
Group: Utilities/SystemSourceRPM: quota-1.55-9.src.rpm
Size: 82232 Packager: RedHat Software bugs@redhat.com
Summary: Quota administration package
Description:
Quotas allow the system administrator to limit disk usage by a user and /or
group per filesystem. This package contains the tools which are needed to
enable, modify, and update quotas.
```

6.4.6 Kiểm tra gói phần mềm

Chế độ làm việc cuối cùng của RPM là kiểm tra xem xét lại gói phần mềm. Đến một lúc nào đó bạn sẽ muốn thử nghiệm tính nhất quán của một tệp trên hệ thống. Giả sử bạn nghi ngờ là có một tệp bị hỏng do một chương trình hoặc một user. Bạn muốn so sánh với những tệp nguyên thuỷ mà bạn đã cài đặt. RPM giúp bạn thực hiện việc này bằng tuỳ chọn –V (V viết hoa). Lệnh kiểm tra sẽ so sánh kích cỡ, tổng kiểm tra (checksum) MD5, lập cấu hình nhóm, loại tệp, người sở hữu tệp và permission (quyền truy cập hay phạm vi tác động đến tệp). Muốn biết xem từ khi cài đặt đến nay, một tệp nào đó của gói phần mềm có bị chỉnh sửa gì chưa, bạn gõ lệnh **rpm** –V với tên gói phần mềm. Thí dụ muốn kiểm tra gói phần mềm có tên quota, bạn gõ lệnh như sau:

```
rpm – V quota
```

Nếu không có gì thay đổi, RPM sẽ không hiển thị gì hết. Ngược lại RPM sẽ trình bày một dãy 8 ký tự cho thấy điều gì đã thay đổi và tên tệp đã thay đổi. Lúc ấy bạn có thể xem lại một số tệp trong gói phần mềm để quyết định xem có phải cài đặt lại gói phần mềm ấy hay không. Bảng 6.4 liệt kê các mã báo lỗi hiển thị sau khi gõ lệnh.

Mã	Ý nghĩa
c	Tệp này là tệp cấu hình
5	Tệp này không qua được thử nghiệm tổng kiểm MD5
S	Kích cỡ tệp đã thay đổi sau thời điểm cài đặt
L	Có vấn đề với các mối liên kết tượng trưng
T	Thời gian chỉnh sửa tệp không trùng với nguyên thuỷ
D	Thuộc tính thiết bị
U	Các thiết lập cho user đã thay đổi
G	Các thiết lập cho nhóm đã thay đổi
M	Chế độ

Bảng 6.4: Mã báo lỗi khi kiểm tra

6.4.7 Cài đặt phần mềm không của Linux

Rất tiếc là đa phần các chương trình không của Linux thì không ở dạng gói phần mềm RPM, nhất là các chương trình được tải từ cơ sở dữ liệu xuống qua cổng FTP vô danh.

Quá trình cài đặt những phần mềm đó rất khác nhau, đi từ mức độ hết sức đơn giản cho đến vô cùng khó khăn, hầu như không thể cài đặt được. Việc này tuỳ thuộc vào cách các tác giả viết ra các script cài đặt và tuỳ thuộc vào tư liệu về cài đặt của họ.

6.4.7.1 Các định dạng của gói phần mềm

Các gói phần mềm tải qua cổng FTP vô danh được ép thành tệp nén. Những tệp này được tạo ra bằng nhiều cách khác nhau. Thông thường thì cây thư mục chứa tệp nguồn, tư liệu, tập thi hành và các tệp khác gộp chung thành tệp gộp (**tar file**) bằng chương trình **tar.** Sau đó tệp gộp được nén để chiếm ít chỗ.

Thường thì gói phần mềm có đuôi mở rộng giúp bạn biết tệp theo định dạng (format) nào. Nếu đó là .gz thì tệp được nén bằng chương trình **gzip** của GNU. Đây là định dạng tệp nén phổ biến nhất cho các gói phần mềm Linux. Nếu đuôi mở rộng là .Z, đó là tệp nén bằng chương trình **compress**. Thí dụ gói phần mềm foo.tar**.**gz là tệp gộp **tar,** được nén bằng **gzip**.

Ghi chú: Đôi khi một tệp **tar** được nén bằng **gzip** sẽ mang đuôi .tgz thay vì .tar**.**gz.

Xem "Sử dụng lệnh tar"

6.4.7.2 Cài đặt phần mềm

Sau khi xem qua định dạng của gói phần mềm, bạn quyết định đặt các tệp nguồn ở đâu để bắt đầu xây dựng gói phần mềm. Có những gói phần mềm rất lớn do đó bạn nên đặt chúng vào hệ thống tệp nào đó còn dư nhiều chỗ. Có người tạo ra hệ thống tệp riêng cho nguồn, sau đó **mount** (lắp đặt) vào một thư mục, chẳng hạn như /urs/local/src, hoặc /src. Tuỳ bạn muốn đặt vào đâu cũng được, song hãy nhớ chừa đủ chỗ để cho phần mềm sau đó được biên dịch thành công.

Bạn di chuyển gói phần mềm đến cây nguồn đã thiết lập, sau đó bung ra toàn bộ. Đối với những tệp được nén bằng lệnh **gzip**, bạn bung ra bằng lệnh **gzip**, thí dụ:

```
gzip –d foor.tar.gz
```

sẽ bung ra tệp nén foo.tar**.**gz và thay thế bằng tệp gộp mang tên foo.tar**.** Bạn xem bảng 6.5 về các tuỳ chọn của lệnh **gzip**.

Flag	Tên Flag	Mô tả
-a	ascii	Văn bản dạng ASCII; chuyển đổi các ký tự cuối dòng bằng cách sử dụng những quy ước cục bộ.
-c	stdout	Ghi ra stdout (đầu xuất chuẩn, tức màn hình; sẽ giải thích sau), giữ các tệp nguyên thuỷ không thay đổi.
-d	decompress	Giải nén (bung ra)
-f	force	Khẳng định ghi chồng lên tệp xuất và nén các mối liên kết.
-h	help	Liệt kê phần trợ giúp.
-l	list	Liệt kê nội dung tệp nén.
-L	license	Hiển thị bản quyền phần mềm
-n	no-name	Không lưu và không phục hồi tên và ngày giờ nguyên thuỷ.
-N	name	Lưu và phục hồi tên và ngày giờ nguyên thuỷ.
-q	quiet	Bỏ qua tất cả các nhắc nhở cảnh báo.
-s suf	suffix.suf	Sử dụng hậu tố suffix.suf vào các tệp nén.
-t	test	Thử nghiệm tính toàn vẹn của tệp nén.
-v	verbose	Chuyển sang chế độ có thông báo.
-V	version	Hiển thị số phiên bản

-1	fast	Nén nhanh hơn.
-9	best	Nén chặt hơn (nghĩa là tệp sẽ nhỏ hơn)
file		Xác định tệp nào cần thao tác; nếu bỏ trống, máy sẽ sử dụng stdin (đầu nhập chuẩn, tức bàn phím; sẽ giải thích sau)

*Bảng 6.5: Các tuỳ chọn của lệnh **gzip***

Đối với những tệp được nén bằng lệnh **compress**, bạn bung ra bằng lệnh **uncompress**, thí dụ:

```
uncompress foo.tar.z
```

sẽ bung ra tệp nén foo.tar**.**Z và thay thế bằng tệp gộp mang tên foo.tar**.**

Sau khi bung ra tệp nén xong, bạn chuyển tệp **tar** vào cây thư mục. Bạn sẽ đặt nguồn của từng gói phần mềm riêng rẽ vào thư mục của chúng trên cây thư mục.

6.4.7.3 Sử dụng lệnh tar

Chú ý rằng đối với lệnh **tar,** bạn có thể có 3 cách ghi tuỳ chọn:

- Gợi nhớ với 2 dấu trừ và tên tuỳ chọn. Thí dụ: **tar** --help

- Viết tắt với một dấu trừ và tuỳ chọn viết tắt. Thí dụ: **tar** -h

- Kiểu cũ: khi có nhiều tuỳ chọn và quen dùng bạn có thể viết gộp tất cả các tuỳ chọn cần sử dụng:. Thí dụ: **tar** zxvf ttt**.**

Tuy nhiên cách này dễ gây hiểu lầm và cho kết quả sai lệch. Thí dụ:

```
tar cvbf 20 /dev/rmt0
```

khác

```
tar –c –v –b 20 –f /dev/rmt0
```

bởi vì ở lệnh thứ hai thì 20 là trị số của tuỳ chọn –b và /dev/rmt0 là trị số của –f. Nếu ta viết lại lệnh như sau: **tar** cvbf /dev/rmt0 20, kết quả sẽ khác nữa.

Trước khi giải nén tệp **tar,** bạn phải xem lại tệp đó có được tạo ra với một thư mục hay không và thư mục ấy có được coi là mục ghi đầu tiên hay không. Bạn dùng lệnh:

```
tar tvf tên_tệp_tar|more
```

để xem thư mục ghi đầu tiên trong tệp **tar** có phải là thư mục hay không. Nếu phải, tệp **tar** sẽ tạo ra thư mục khi được bung ra. Nếu không có thư mục ở cấp cao nhất của tệp **tar,** tất cả các tệp ở cấp cao nhất sẽ được giải nén vào thư mục hiện hành. Trong trường hợp này, bạn phải tạo ra một thư mục và chuyển tệp **tar** vào đấy trước khi bung ra**.**

Ghi chú: Trước khi bung ra tệp **tar,** bạn nên kiểm tra xem có thư mục ở cấp cao nhất hay không. Trong trường hợp tệp **tar** bung ra vài trăm tệp vào thư mục hiện hành thì rắc rối lắm.

Một khi đã đặt tệp **tar** vào chỗ mà bạn muốn bung ra nó ra, bạn dùng lệnh như sau để bung ra cây nguồn vào tệp **tar**:

```
tar xvf tên_tệp_tar
```

Bước tiếp theo tuỳ thuộc vào cách viết chương trình của gói phần mềm mà bạn đang cài đặt. Thường thì bạn sẽ chuyển sang thư mục cấp cao nhất của nguồn phần mềm và tìm một tệp nào đó có dạng như README.1ST. Ở thư mục nguồn cấp cao thường có vài tệp tư liệu giải thích quá trình cài đặt.

Ghi chú: Với hầu hết các phiên bản Linux, bạn có thể giải nén một tệp **tar** luôn một thể khi khai thác nó. Bạn chỉ cần thêm flag z vào lệnh **tar,** chẳng hạn như:

```
tar zxvf foo.tar.gz
```

Quá trình cài đặt thường bao gồm việc hiệu chỉnh tệp Makefile để chỉnh sửa các thư mục đích. Phần mềm sẽ đặt các tệp nhị phân đã biên dịch vào những thư mục này. Thông thường bạn chạy lệnh **make** và tiếp theo đó là lệnh **make** install.

Quá trình thực hiện lệnh **make** có thể thay đổi (gọi các chương trình biên dịch hay cài đặt) theo từng gói phần mềm mà bạn cài đặt. Đối với vài gói phần mềm, một dạng shell script chuyên lập cấu hình sẽ yêu cầu bạn trả lời vài câu hỏi trước khi biên dịch phần mềm cho bạn. Bạn nên đọc trước các tệp tư liệu đi chung với gói phần mềm (nhất là tệp INSTALL).

6.4.8 Xem lại các quyền truy cập

Thông thường, việc thiết lập các quyền truy cập cho gói phần mềm sẽ diễn ra trong quá trình cài đặt. Mỗi ứng dụng sẽ có một script cài đặt đi kèm và script này sẽ cài đặt từng tệp với phần sở hữu và quyền truy cập tương ứng. Chỉ khi có trục trặc và khi có một user không thực hiện được những việc được quyền, thì lúc ấy bạn mới phải tìm ra thư mục nơi mà ứng dụng được chép vào và xem lại các quyền truy cập.vTrên nguyên tắc thì tệp mà bạn dùng để kích hoạt ứng dụng đã bao gồm những quyền truy cập cho phép tất cả các user sử dụng ứng dụng đó. Chỉ superuser mới có khả năng xoá bỏ hoặc ghi đè lên. Thường thì ứng dụng được cài đặt vào một thư mục cho phép đọc và thi hành, chứ không ghi đè lên.

Xem "Các quyền truy cập tệp"

6.4.9 Giải quyết vấn đề

Một ứng dụng được viết tốt và hỗ trợ tốt sẽ được cài đặt vào hệ thống máy bạn mà không đòi hỏi bạn cung cấp nhiều thông tin. Ứng dụng tốt sẽ thiết lập các quyền truy cập đúng cách, sao cho công việc của bạn được đơn giản hoá, nghĩa là bạn chỉ cần chạy thử và báo cho user khác biết rằng ứng dụng mới đã khả thi rồi. Tuy nhiên không phải mọi thứ đều trơn tru. Nếu vì lý do nào đó mà chương trình không thể hoàn thành việc nạp tư liệu hoặc sau khi cài đặt xong mà vẫn không chạy tốt, bạn phải có trách nhiệm xác định lý do rồi sau đó đưa ra giải pháp.

Nếu một chương trình không được cài đặt hoàn hảo, công việc gỡ lỗi của quản trị viên thường chỉ là đọc tư liệu trợ giúp và các tệp README đi kèm với ứng dụng để xem qua danh sách các vấn đề thường gặp và giải pháp của chúng. Tuy nhiên bạn khó biết hết tất cả các phần mềm viết cho Linux, do đó phải cầu viện ở ngoài.

Nếu không thể gỡ lỗi bằng thông tin đi kèm gói phần mềm, bạn thử liên lạc với các nhóm tin Usenet để tham khảo những gì mọi người bàn bạc về phần mềm đó. Nếu gửi câu hỏi đến nhóm tin đúng, nhiều vấn đề của bạn sẽ được giải quyết. Nếu mạng Internet cũng không giúp gì được, bạn vẫn có thể liên lạc qua e-mail với người phát

triển ứng dụng ấy. Xin nhắc bạn rằng Linux là phần mềm miễn phí, do đó hầu hết các gói phần mềm cho Linux cũng không mất tiền. Vì vậy có lẽ bạn cũng không nên hy vọng luôn luôn nhận được sự trợ giúp dễ dàng.

6.4.10 Gỡ bỏ các ứng dụng

Nếu có gói phần mềm nào đó không còn được sử dụng hoặc đã có phiên bản tốt hơn, bạn nên gỡ bỏ nó khỏi hệ thống.

Cũng như khi cài đặt, gỡ bỏ một chương trình trên hệ Linux rắc rối hơn so với thao tác trên những hệ điều hành chỉ dành cho một user. Nhiều khi việc gỡ bỏ không đơn giản chỉ là xoá hết tệp của ứng dụng và sau đó xoá luôn thư mục. Các driver và những liên kết phần mềm khác cũng phải tháo ra để sau này không gặp rắc rối. Mỗi khi máy hiển thị lời nhắc hay lời cảnh báo, bạn nên ghi chúng vào một tệp nhật ký (logfile) để sau này tổng kết xem những gì đã đổi thay sau khi cài đặt. Từ đó bạn suy ra nên gỡ bỏ và thay đổi những tệp nào để cho việc gỡ bỏ một gói phần mềm được hoàn tất tốt.

6.5 Nâng cấp Kernel

Cùng với những bản nâng cấp và gói phần mềm khác, những phiên bản kernel Linux mới cũng được phát hành đều đặn. Những phiên bản này đã được sửa lỗi và tăng thêm chức năng. Bạn nâng cấp kernel để lập lại cấu hình hoặc để bổ sung driver cho các thiết bị mới.

Trước khi nâng cấp kernel, bạn nên sao lưu phần mềm hệ thống và dành một đĩa mềm khởi động Linux, phòng khi lỡ tay làm hỏng. Mô tả chi tiết về các thao tác xây dựng lại kernel cho Linux nằm ở chương "Lập cấu hình kernel Linux".

Tiến trình nâng cấp kernel được mô tả chi tiết trong phần tư liệu HOWTO về Kernel. Tư liệu này được niêm yết đều đặn trên Internet, tại các website FTP Linux hoặc các nhóm thông tin Linux. Bạn nên sao chép một bản HOWTO để đọc kỹ trước khi nâng cấp kernel.

Bước đầu của tiến trình cơ bản nâng cấp kernel là truy cập các nguồn kernel mới qua FTP vô danh tại các website về Linux. Khi đã có nguồn mới, bạn cần lưu lại nguồn cũ. Bạn chuyển thư mục /usr/src/linux sang tên khác, chẳng hạn như /usr/src/linux.old. Bạn bung ra gói phần mềm nguồn kernel vào thư mục /usr/src và thư mục Linux sẽ được tự động tạo ra. Đến đây, bạn chuyển sang thư mục Linux để đọc các tư liệu và các tệp README.

Từ đây trở đi, tiến trình có thay đổi chút ít. Bạn gõ lệnh **make** config để chạy phần script về cấu hình và cung cấp thông tin hệ thống. Nếu phần này hoàn tất, bạn gõ lệnh đại loại như **make** dep (dependencies) để kiểm tra tình trạng phụ thuộc các tệp, nhằm đảm bảo kernel mới sẽ tìm ra tất cả các tệp cần thiết để biên dịch.

Sau công đoạn kiểm tra tính phụ thuộc, bạn gõ lệnh **make** clean để xoá mọi tệp đối tượng cũ rải rác ở thư mục kernel nguồn. Nếu đến thời điểm này mọi việc vẫn tốt, bạn yên tâm gõ **make** để biên dịch kernel mới. Biên dịch xong, bạn dùng chương trình quản lý mồi LILO (LILO Boot Manager) để cài đặt kernel mới.

Xin nhắc bạn lần nữa là hãy đọc kỹ phần HOWTO của kernel trước khi bắt tay vào việc, như thế bạn sẽ tránh mọi phiền phức bực bội. Đồng thời cũng giúp bạn tránh tình trạng bỏ rác vào hệ Linux hiện hành.

6.6 Cài đặt trong môi trường X bằng RPM

Nếu như bạn không muốn nhớ các câu lệnh phức tạp ở trên mà muốn thực hiện các công việc một cách dễ dàng như trong môi trường Windows, bạn có thể sử dụng trình RPM của GNOME.

Tất cả các chức năng của RPM dạng dòng lệnh đều được đưa vào trong GNOME-RPM (gnorpm) dưới dạng các cửa sổ, menu, thanh công cụ hay hộp thoại.

6.6.1 Khởi động GNOME-RPM

Có thể khởi động Gnome-RPM theo một trong các cách sau:

- Trong môi trường GNOME, chọn lần lượt các mục sau Main Menu Button => Programs => System => GnoRPM

- Trong môi trường KDE, chọn lần lượt các mục sau Main Menu Button => Programs => System => GnoRPM

- Tại một terminal/console bạn gõ:

```
# gnor &
```

Minh hoạ 6.1: Trình quản lý gói phần mềm GnoRPM

Ta sẽ làm việc với cửa sổ như sau:

Các gói phần mềm sẽ được trình bày thành nhóm/cây bên bảng bên trái và các gói phần mềm cụ thể trong từng nhóm được liệt kê trên bảng bên phải.

Có thể chọn các tác động trong menu pop-up Operation hay trên thanh công cụ.

Minh hoạ 6.2: Lựa chọn gói phần mềm

6.6.2 Chọn gói phần mềm

Khi chọn một nhóm phần mềm, danh sách các gói phần mềm trong nhóm hiện ra cho phép bạn chọn một hay nhiều gói phần mềm để tác động. Bạn giữ phím Ctrl (Control) và nhấp chuột trái lên biểu tượng (biểu tượng) của phần mềm cần chọn. Bạn cũng có thể giữ phím Shift và nhấp lên biểu tượng của phần mềm thứ nhất và biểu tượng của phần mềm cuối cùng nếu bạn chọn một nhóm các phần mềm nằm liên tiếp nhau.

Dòng thông báo trên thanh trạng thái ở bên dưới cửa sổ cho biết số lượng phần mềm mà bạn định chọn.

6.6.3 Cài đặt phần mềm mới

Bạn chọn Install trên thanh công cụ để cài đặt phần mềm mới.

Những gì bạn thấy trong cửa sổ Install sẽ phụ thuộc vào sự lựa chọn khi dùng Filter.

Theo mặc định, Red Hat sẽ tìm trong đường dẫn /mnt/cdrom/RedHat/RPMS để tách ra các gói phần mềm có thể được cài đặt. Bạn có thể thay đổi đường dẫn mặc định này trong Operations => Preferences.

Một thông tin ngắn về phần mềm bạn chọn sẽ hiện lên trong bảng Package Info.

Minh hoạ 6.3: Cài đặt thêm gói phần mềm RPM

Chọn Add nếu bạn muốn cài đặt phần mềm bạn đang chỉ định (highlight).

Chọn Query để biết thông tin về phần mềm đã chọn một cách chi tiết hơn. Bạn cũng có thể chọn Upgrade khi phần mềm được chỉ định đã được cài đặt trước (với phiên bản cũ hơn).

6.6.4 Lập cấu hình mặc định cho trình cài đặt

Khi nhập vào Operations => Prerences, bạn được phép xác định lại cách cài đặt mặc định cho trình GNOME-RPM.

Minh hoạ 6.4: Thông tin về gói phần mềm

Bạn không nên bật tuỳ chọn “no dependency checks”. Trong Linux, cũng như trong Windows, hiện nay, các phần mềm thường được xây dựng dựa trên các hàm thư viện có sẵn của Hệ điều hành hay của một số phần mềm khác.

Minh hoạ 6.5: Định cấu hình mặc định cho trình cài đặt

Bạn cũng có thể định các cấu hình về Network với việc dùng Webfind, Rpm**find** hay Distributions để cài đặt các gói phần mềm thông qua mạng hay Internet.

Minh hoạ 6.6: Định cấu hình mạng để cài đặt RPM

6.6.5 Gỡ bỏ phần mềm

Khi chọn Uninstall, bạn quyết định gỡ bỏ phần mềm đã chọn.

Minh hoạ 6.7: Gỡ bỏ phần mềm

Màn hình Remove Packages sẽ nhắc bạn kiểm tra lại các phần mềm sẽ bị tháo bỏ.

Nếu bấm YES, tiến trình gỡ bỏ sẽ bắt đầu.

Khi chấm dứt, tên của các gói phần mềm sẽ không còn hiện lên ở bất kỳ cửa sổ nào.

Chương 7. Quản trị hệ thống Linux

Chương này sẽ bàn về một số công việc và vấn đề chủ chốt của một quản trị viên hệ thống trong môi trường multiuser của Linux. Nếu đọc giáo trình này để tìm hiểu và cài đặt Linux, thì tức khắc gần như bạn đã trở thành quản trị viên hệ thống. Nhiều mục trong chương này sẽ thiên về quản trị hệ thống cho các mạng cơ quan. Tuy nhiên, cho dù chỉ là user duy nhất dùng Linux trên máy gia đình, bạn cũng nên làm quen với việc quản trị mạng lớn để mở rộng nhận thức về những vấn đề tổng quát hơn.

Những chủ đề chính sẽ được đề cập trong chương này bao gồm:

- *Tầm quan trọng của quản trị hệ thống*
- *Khái niệm multiuser*
- *Các hệ thống xử lý tập trung*
- *Các hệ thống xử lý phân tán*
- *Mô hình khách/chủ*
- *Quản trị trong môi trường mạng*
- *Xác định vai trò quản trị viên mạng.*

Một hệ thống Linux phải có ít nhất một quản trị viên hệ thống để quản lý và kiểm tra hiệu năng chung của hệ thống ấy. Quản trị viên hệ thống phải biết làm sao cho hệ thống chạy đúng và trong trường hợp hỏng hóc phải biết tự mình hoặc nhờ ai đó sửa chữa. Ngoài ra quản trị phải biết cách cung cấp tài nguyên phần cứng lẫn phần mềm cho các user. Quản trị viên phải lập cấu hình nguyên thuỷ cho hệ thống, sau đó liên tục theo dõi điều chỉnh sao cho hệ thống chạy hiệu quả và tin cậy. Quản trị viên phải đáp ứng mọi yêu cầu hệ thống, vì vậy phải đảm trách nhiều công việc khác nhau.

Trong nhiều trường hợp, Linux được nối mạng với những máy không chạy Linux. Những máy đó có thể chạy các hệ điều hành kiểu UNIX hoặc một hệ hoàn toàn khác. Bởi vì Linux là một hệ điều hành kiểu UNIX cho nên phần lớn thông tin sau đây áp dụng cho cả Linux và UNIX, tức là ở nhiều nơi trong giáo trình này, ta sử dụng chung hai từ Linux và UNIX với cùng một khái niệm như nhau.

7.1 Tầm quan trọng của quản trị hệ thống

Ở khía cạnh này hay khía cạnh khác, tất cả các hệ UNIX đều có điểm khác nhau, do đó phải được quản lý khác nhau. Linux không là ngoại lệ. Trách nhiệm quản lý của bạn thay đổi trên cơ sở các tham biến chẳng hạn như số user phải quản lý, loại thiết bị ngoại vi kết nối vào hệ, những mối liên kết mạng và mức độ an ninh cần có.

Một quản trị viên hệ thống, dù làm việc một mình hoặc với tổ kỹ thuật, cũng phải bảo đảm một môi trường có đủ khả năng đáp ứng những người sử dụng hệ thống. Quản trị viên có quyền hạn và trách nhiệm cung cấp và duy trì một hệ thống hiệu quả và tin cậy.

Tuy nhiên trong chế độ multiuser luôn xuất hiện nhiều mục tiêu và ưu tiên cạnh tranh khác nhau. Quản trị viên phải sử dụng quyền hạn và trách nhiệm của mình để dung hoà những mâu thuẫn ấy.

Việc phân quyền quản trị cũng thay đổi tuỳ theo từng hệ thống. Trên những hệ thống lớn, công tác quản trị có thể chia cho nhiều người. Ngược lại, các hệ thống nhỏ không nhất thiết phải có một người lo quản trị trong toàn bộ thời gian mà chỉ cần một user kiêm nhiệm chức quản trị viên. Khi làm việc trong môi trường mạng, hệ thống thường được quản lý bởi một người quản trị riêng.

Mỗi hệ Linux có một user với quyền hạn truy cập rất nhiều thứ trên máy. Use này được gọi là superuser và có tên đăng nhập là root (gốc). Khi đăng nhập hệ thống, root user dùng ký tự / như là thư mục "nhà" của mình (home directory hay thư mục gốc của hệ thống tệp), hoặc /home/root như một thư mục "nhà" đặc biệt. Với Red Hat, /root là thư mục "nhà"của trương khoản root.

Quản trị viên hệ thống đăng nhập với tư cách là root để thực thi một số công việc đòi hỏi ưu tiên cao. Khi vào làm việc bình thường, superuser cũng đăng nhập như là một user bình thường. Tên đăng nhập của superuser (root) cần được sử dụng có giới hạn. Số user được phép đăng nhập với tư cách root nên hạn chế, tối đa là hai hoặc ba người. Người nào đăng nhập với tư cách root coi như có quyền hạn tuyệt đối trên hệ thống. Người ấy có thể thay đổi thuộc tính bất kỳ tệp nào, có quyền đình chỉ hoặc khởi động hệ thống, sao lưu dữ liệu hệ thống, cùng với nhiều thao tác khác.

Quản trị viên phải nắm vững nhiều khía cạnh kỹ thuật của (mạng) máy tính và cần hiểu rõ nhu cầu của các user và mục tiêu chính của hệ thống. Bất kỳ hệ thống nào cũng là một tài nguyên có giới hạn, do đó các chính sách về quyền sử dụng tài nguyên ấy phải được thiết lập hợp lý và thực thi nghiêm túc. Quản trị viên vừa chịu trách nhiệm kỹ thuật vừa đóng vai trò người thực thi chính sách.

Vai trò này, cộng với quyền được thao tác vô hạn trên hệ thống, đòi hỏi quản trị viên phải là người có chuyên môn, có trách nhiệm và có khả năng xã giao. Bản liệt kê chính xác các công việc của một quản trị viên hệ thống thường thay đổi tuỳ theo từng nơi. Tuy nhiên tất cả các quản trị viên đều phải làm những việc sau đây:

- Quản lý user. Thêm user vào, bớt user ra và thay đổi giới hạn thao tác, cùng với những ưu tiên của user.

- Lập cấu hình thiết bị. Bảo đảm các thiết bị như máy in, terminal, modem, ổ băng từ đều chạy tốt và nhiều người được dùng chung.

- Thực hiện sao lưu. Lập lịch sao lưu, tiến hành và lưu trữ các dữ liệu quan trọng để có thể phục hồi sau khi các tệp gốc bị hỏng hóc hay mất mát.

- Đóng tắt hệ thống. Đóng tắt hệ thống đúng quy trình để tránh tình trạng thiếu nhất quán của hệ thống tệp.

- Huấn luyện user. Gián tiếp hoặc trực tiếp đào tạo user sao cho họ sử dụng hệ thống có hiệu quả và tăng năng suất.

- Bảo vệ hệ thống. Tránh tình trạng các user can thiệp vào công việc của nhau một cách tình cờ hay cố ý.

- Ghi chép các thay đổi trên hệ thống. Duy trì sổ nhật ký ghi chép mọi hoạt động có liên quan đến hệ thống.

- Tư vấn cho các user. Đóng vai trò "chuyên gia tại chỗ" để trợ giúp cho các user bình thường.

7.2 Khái niệm multiuser

Một hệ điều hành kiểu Linux/UNIX bao gồm hai tính chất cơ bản: multitasking (đa nhiệm) và multiuser (đa người dùng). Tính đa nhiệm là khả năng thực thi nhiều công việc cùng một lúc. Thí dụ bạn có thể đọc thư điện tử trong khi đang tiến hành biên dịch một chương trình.

Mỗi công việc như thế - cho dù chỉ là một lệnh đơn giản gõ từ dòng nhập lệnh hoặc một ứng dụng phức tạp - đều khởi động một hay nhiều tiến trình khác nhau. Bất cứ chương trình nào chạy trên Linux đều liên kết với một tiến trình khác. Và bởi vì có thể chạy nhiều tiến trình cùng lúc nên Linux là một hệ điều hành đa nhiệm.

Bạn có thể kết nối với một máy tính chạy Linux/UNIX (coi như một máy chủ) bằng nhiều cách, trên cơ sở dùng một terminal hay một máy PC cũng được. Bạn có thể ở gần bên máy chủ và chỉ cần kết nối bằng dây cáp. Nếu bạn ở xa máy chủ thì có thể kết nối bằng đường điện thoại thông thường hoặc đường truyền cao tốc.

Chính việc dùng loại máy nào và được kết nối như thế nào sẽ xác định xem tài nguyên của máy tính là dạng tập trung hay phân tán.

Các hệ điều hành đơn nhiệm, chẳng hạn như DOS, được thiết kế cho một người dùng. Tất cả công việc được tiến hành trên một máy PC và hệ điều hành đơn nhiệm chỉ có quyền truy cập một tài nguyên (như máy in, thiết bị lưu trữ và bộ xử lý) vào mỗi thời điểm xác định.

Hệ thống multiuser sử dụng các mô hình xử lý tập trung và xử lý phân tán để tạo thuận lợi cho nhiều user làm việc cùng một lúc.

- Trong môi trường xử lý tập trung, nhiều user truy cập tài nguyên của một máy tính (những máy tính lớn có thể phục vụ hàng trăm, hàng nghìn user). Việc lưu trữ, in ấn, ghi vào bộ nhớ cũng như các công tác xử lý đều được một chiếc máy đó thực hành.

- Trong môi trường xử lý phân tán, việc xử lý có thể diễn ra tại chính trạm làm việc của user và (các) máy chủ chỉ phân phối phần mềm ứng dụng và dữ liệu. Các thiết bị in ấn và lưu trữ có thể kết nối với trạm làm việc của user hoặc với máy chủ.

7.3 Các hệ thống xử lý tập trung

Trên đà phát triển của công nghệ thông tin, trong những thập niên 1950 và 1960, các hệ điều hành multiuser đã ra đời, cho phép nhiều user chia sẻ tài nguyên từ các terminal riêng lẻ. Sử dụng **batch**-processing sequence (trình tự xử lý theo lô), hai user có thể thực thi hai chương trình khác nhau trong khi vẫn dùng chung một bộ xử lý trung tâm, một thiết bị lưu trữ và một thiết bị kết xuất. Với sự phổ biến của mạng điện thoại chuyển mạch (PSTN), máy tính bắt đầu sử dụng được nhiều tài nguyên tin học ở xa mà không phụ thuộc khoảng cách địa lý. Trong mô hình này, từng bộ xử lý sẽ sử dụng tài nguyên xử lý liên lạc để kết nối với các terminal ở xa. Từ đó đã phát triển ra

mô hình front-end processing (xử lý mặt trước) về mặt liên lạc và mô hình xử lý tập trung về mặt dữ liệu.

Trước khi các máy tính được giảm giá mạnh mẽ và trở nên phổ biến khắp nơi thì hầu hết các hệ UNIX đều dùng mô hình xử lý tập trung. Với cách xử lý này, một máy tính lớn (mainframe) có thể phụ trách mọi xử lý. Các user chỉ việc kết nối tới đó và dùng chung tài nguyên của máy lớn. Hiện nay mô hình này càng ngày càng ít được dùng, mặc dù nó vẫn thích hợp cho những trường hợp các user ở cách nhau quá xa. Thí dụ cơ quan của bạn có một trung tâm xử lý chính và tất cả các chi nhánh từ xa đều có thể truy cập trung tâm này. Trên bàn của mỗi user là một terminal, bao gồm bàn phím và màn hình được kết nối với máy lớn và dùng chung những tài nguyên như máy in, bộ lưu trữ, vv...

Mô hình xử lý tập trung thường gồm nhiều thành phần như server, bộ xử lý mặt trước (front processor), terminal, modem và những bộ ghép nối nhiều cổng. Khi một user cần truy vấn từ xa, yêu cầu này được gửi về trung tâm và máy tính tại đây sẽ xử lý, sau đó trung tâm sẽ gửi kết quả về nơi yêu cầu. Mọi dữ liệu đều được xử lý và lưu trữ bởi máy lớn.

7.4 Các thành phần của mô hình xử lý tập trung

Muốn làm việc theo mô hình xử lý tập trung, bạn phải có một số thành phần như server, bộ xử lý mặt trước, terminal, modem và những bộ ghép nối nhiều cổng.

Có thể định nghĩa server như là một máy tính được thiết lập cấu hình nhằm chia sẻ tài nguyên với những máy khác. Thí dụ bạn có thể dùng một máy tính tương thích họ IBM PC làm server với điều kiện máy phải có đủ chỗ trên ổ cứng và đủ RAM.

Bộ xử lý mặt trước kết nối các kênh liên lạc với server và giữ vai trò thao tác các chi tiết liên lạc để server rảnh rang mà xử lý dữ liệu.

Terminal gồm có hai loại phổ biến, đó là terminal thường (dumb) và terminal thông minh (smart). Trước nay UNIX được sử dụng với terminal thường, vốn chỉ có bàn phím và màn hình mà thôi. Điểm nổi bật đối với terminal thường là chúng không có khả năng xử lý. Cổng liên lạc ở terminal thường được nối trực tiếp hoặc gián tiếp với server. Khi gõ bàn phím ở terminal thường, mỗi ký tự gõ vào đều được chuyển về server xử lý. Trong khi đó terminal thông minh có thể xử lý tại chỗ vài công việc đơn giản. Chẳng hạn như máy thu tiền hoặc máy rút tiền tự động là những terminal thông minh. Terminal thông minh lưu trữ yêu cầu giao dịch, sau đó mới chuyển toàn bộ yêu cầu này, thay vì chuyển từng ký tự mỗi lần gõ phím như terminal thường.

Muốn kết nối terminal với server qua mạng điện thoại, bạn cần hai modem. Modem thứ nhất ở đầu này chuyển đổi tín hiệu số của terminal (hoặc của máy tính) thành tín hiệu tương tự (analog) phù hợp cho đường điện thoại, modem thứ hai kết nối đường điện thoại ở đầu kia với server. Muốn làm việc từ xa, qua terminal bạn quay số điện thoại ở đầu kia và liên lạc được với server khi modem ở đầu kia nhận lời.

Muốn tăng số lượng các cổng cho user kết nối vào, bạn cần cài đặt một bộ ghép nối đa cổng. Thông thường một máy PC chỉ có hai cổng nối tiếp COM1 và COM2. Nếu muốn PC của mình trở thành server cho hơn hai user, bạn phải dùng bộ ghép nối đa cổng. Đó là một bìa để lắp đặt vào bus trong PC, đi cùng một cái hộp có tám ổ nối

hoặc nhiều hơn và với một bó cáp nối bìa với hộp. Bộ ghép nối đa cổng còn đi kèm một phần mềm để giúp PC điều khiển các cổng nối tiếp đó.

7.5 Các hệ thống xử lý phân tán

Ở mô hình xử lý phân tán, terminal được thay thế bằng trạm làm việc (workstation), vốn là một máy vi tính chạy DOS hoặc Linux/UNIX, và bạn có thể chạy chương trình hoặc lưu thông tin trên server hay ở trạm làm việc cũng được. Sau khi xử lý tệp ở trạm làm việc xong, bạn lưu trữ thông tin trên server để những user khác có thể truy cập. Bạn có thể tuỳ ý in từ máy in tại chỗ hoặc máy in kết nối với server.

Bởi vì hiện nay các trạm làm việc rất mạnh và rẻ tiền, có khả năng là cơ quan của bạn sử dụng hệ thống xử lý phân tán thay vì hệ thống xử lý tập trung.

7.6 Các thành phần của mô hình xử lý phân tán

Mô hình xử lý phân tán sử dụng các server tệp, trạm làm việc, bìa giao diện mạng, cùng với một số thiết bị khác như bộ tập trung (hub), bộ khuếch đại (repeater), bộ cầu (bridge), bộ định tuyến (router) và bộ cổng ngõ (gateway). Chức năng của server tệp là phân phối các tệp và các đoạn chương trình đến các trạm làm việc. Hơn 90% công việc xử lý được tiến hành tại các trạm làm việc, chỉ có từ 5 đến 10% công việc dành cho server, chủ yếu là các nhiệm vụ quản trị.

Ngoài chức năng server tệp, bạn có thể dùng PC như là một trạm làm việc Linux. Linux được thiết kế để chạy với một cấu hình phần cứng rất khiêm tốn: trước kia vào giữa thập niên 1990 bạn đã có thể chạy các bản Linux đầu tiên với CPU 386 SX và 4 MB RAM! Vì các máy tính hiện nay mạnh hơn hẳn yêu cầu tối thiểu của Linux, bạn có thể yên tâm, tất nhiên còn phải tuỳ theo phiên bản và nhà sản xuất Linux: như với RedHat, bản 5.x có thể chạy với 4 MB RAM còn bản 7.x thì chỉ chạy tốt khi bộ nhớ RAM lớn hơn 64 MB. Còn phải chừa bao nhiêu khoảng trống trên ổ cứng thì tuỳ vào việc bạn muốn cài đặt bao nhiêu phần mềm. Nếu chạy Linux hoàn toàn từ CD-ROM (thí dụ như bản SuSE Linux 7.3 cho CPU Intel) thì bạn chỉ cần 5 MB đĩa cứng. Nếu cài đặt từ đĩa cứng thì cần tối thiểu từ 80 đến 300 MB, còn cài trọn bộ thì cần khoảng 3.5 GB, tuỳ theo phiên bản và nhà sản xuất.

Nói chung, các tài nguyên cần dành ưu tiên cho trạm làm việc, nơi mà phần lớn công tác xử lý được thực thi. Tuỳ vào loại công việc dự định thực hiện, bạn sẽ dần dà đưa thêm tài nguyên vào. Thí dụ những phần mềm xử lý văn bản chiếm chẳng bao nhiêu tài nguyên (ổ cứng, RAM, chất lượng màn hình) khi so với những công việc xử lý hình ảnh, chẳng hạn như các chương trình CAD (thiết kế bằng máy tính). Với những ứng dụng kiểu CAD, bạn cần những ổ cứng lớn (hơn 1 GB) và đầy đủ RAM (hơn 16 MB, có thể lên đến 512 MB), cùng với màn hình và bìa video độ phân giải cao (1280 X 1024 điểm hoặc nhiều hơn). Sau đó, bạn có thể cần đến một ổ băng từ để sao lưu và một ổ CD-ROM để đọc các chương trình lớn.

-NIC (bìa giao diện mạng) được gắn vào một khe trên bo mẹ (mother board) và thực hiện qua ổ nối (BNC hoặc RJ-45) mối liên kết vật lý giữa máy tính với dây cáp mạng. Bạn có thể nối mạng bằng cáp đồng trục hoặc cáp xoắn đôi.

-Hub (bộ tập trung) là nơi kết nối các cáp mạng, chẳng hạn loại hub khuếch đại thụ động (passive repeater hub) thường có bốn ổ nối RJ-45. Hub chuyển mạch (switching hub) thường có ít nhất tám cổng RJ-45. Loại Hub này vừa khuếch đại vừa chủ động chuyển mạch các tín hiệu.

-Repeater (bộ khuếch đại) có chức năng khuếch đại hoặc tái sinh tín hiệu trên mạng để tăng khoảng cách kết nối của cáp mạng.

-Brigde (bộ cầu) được dùng để nối hai mạng có giao thức và giao diện giống nhau.

-Router (bộ định tuyến) được dùng ở các mạng lớn và phức tạp, cho phép tạo lập và lựa chọn đường đi cho các gói tin đạt mục đích về cùng một địa chỉ trên mạng. Router sẽ xác định đâu là lộ trình tối ưu tại mỗi thời điểm và gửi gói tin theo lộ trình ấy.

-Gateway (cổng ngõ) sẽ chuyển đổi các giao thức cần thiết để cho hai mạng dùng các giao thức khác nhau liên lạc được với nhau. Thí dụ muốn liên lạc giữa các mạng TCP/IP, Netware hoặc AppleTalk thì đều phải qua trung gian gateway thích hợp.

7.7 Topo mạng

Topo mạng hay phác hình mạng là phác thảo hình vẽ đường nối của các thiết bị tin học trong một mô hình mạng. Tên của các loại topo thường lấy từ phác hình nối dây cáp giữa các thiết bị mạng, trạm làm việc và máy chủ. Ba topo cơ bản nhất có hình sao, hình tuyến và hình vòng. Mạng nào có hơn một topo thì gọi là mạng lai.

7.7.1 Topo hình sao (star)

Như minh hoạ cho thấy, ở topo hình sao tất cả các trạm làm việc đều kết nối vào hub trung tâm hoặc server tệp trung tâm. Mô hình này có thể sử dụng hub chủ động hay thụ động. Hub thụ động chỉ đơn giản là điểm tập trung kết nối cho các trạm làm việc, trong khi hub chủ động có khả năng chuyển mạch và khuếch đại các tín hiệu. Các mạng Ethernet 10/100/1000 Base-T là những thí dụ về sử dụng topo hình sao.

7.7.2 Topo hình tuyến (bus)

Tất cả các trạm làm việc và server tệp trong topo hình tuyến đều kết nối trực tiếp và chia sẻ một đường đi chung. Topo hình tuyến thường được dùng cho các mạng Ethernet 10 Base-2 và Ethernet 10 Base-5 nối bằng cáp đồng trục.

7.7.3 Topo hình vòng (ring)

Trong topo hình vòng, các thiết bị mạng, server và trạm làm việc được bố trí giống như ở topo hình tuyến, chỉ khác là hai đầu tuyến được nối lại để tạo thành vòng tròn kín. Topo hình vòng sử dụng một loại thiết bị khuếch đại riêng, gọi là MAU (Multistation Access Unit: đơn vị truy cập đa trạm). Mạng Token Ring của IBM là một thí dụ về topo hình vòng.

7.7.4 Topo lai (hybrid)

Ở những thập niên 1970 và 1980, các mạng máy tính đã bắt đầu triển khai ở cấp phòng ban với nhiều topo khác nhau. Trong các cơ quan lớn, thường có tình trạng không

đồng nhất, thí dụ: phòng kế toán sử dụng Ethernet hình tuyến, xưởng sản xuất cài đặt t Token Ring, các phòng còn lại có mạng Ethernet hình sao, trong khi bộ phận quản trị lại dựa vào công nghệ xử lý tập trung với máy lớn mainframe. Chính việc cần liên kết những loại mạng khác nhau như thế đã cho ra đời topo lai và mạng diện rộng (WAN). Loại mạng này kết hợp các topo cơ bản hình vòng, sao và tuyến.

7.8 Mô hình client/server

Kết quả của việc xử lý phân tán là mô hình client/server (khách/chủ). Các hệ Linux có thể được dùng cho mô hình này với tư cách là khách, hoặc là chủ, hoặc cả hai.

Để tìm hiểu việc thiết lập quan hệ client/server, bạn hãy giả định nhiều trạm làm việc Linux (client) được kết nối với một server theo topo hình tuyến. Server này sẽ dành thư mục riêng cho từng client. Mỗi thư mục chứa nhiều tệp và đêm đêm lại được sao lưu đều đặn. Server cũng có những thư mục chung để các client có thể chia sẻ một số tệp với nhau. Thường thì một máy in laser được nối với server để mọi người dùng chung và một ổ băng từ cũng nối như thế để sao lưu các ổ cứng dung lượng lớn. Ngoài ra, một số client còn có máy in riêng của mình được kết nối tại chỗ.

7.9 Quản trị trong môi trường mạng

Hình thức thường gặp của một mạng Linux/UNIX bao gồm nhiều máy tính lớn nhỏ khác nhau được kết nối bằng cáp mạng hoặc đường điện thoại. Quản trị mạng là công việc của một hoặc nhiều người tại một (số) trong những máy của mạng.

Hầu như ai cũng có khả năng tìm hiểu về Linux và quản trị một mạng máy vi tính. Với tính kiên trì cộng với nhiều thực hành, ngay cả những người với hiểu biết hạn chế về tin học cũng có thể học cách quản trị một mạng Linux.

7.10 Xác định vai trò quản trị viên mạng

Gần như nơi nào có một số máy chạy UNIX/Linux được nối mạng thì nơi ấy sẽ có một người chuyên trách mạng. Quản trị viên mạng cần có kiến thức nhất định để quyết định các kết nối vào hệ thống (qua mạng cục bộ hay modem), mức độ an ninh cần thiết và việc chia sẻ các thiết bị ngoại vi. Hàng ngày, quản trị viên kiểm tra danh sách tên hệ thống, địa chỉ mạng và việc vào ra của các user.

Những tổ chức lớn với mạng lưới hàng trăm máy vi tính thường có nhiều quản trị viên ở các lĩnh vực liên quan. Việc này là cần thiết nếu các user có nhu cầu chuyên môn, thí dụ về in ấn hoặc tạo mẫu đa dạng. Việc quản lý máy in và công tác in đòi hỏi kiến thức sâu rộng về những loại máy in nhất định và làm cách nào để cho chúng tương thích với các máy tính và phần mềm tác nghiệp.

7.11 Lựa chọn phần cứng và phần mềm

Là một quản trị viên hệ thống, bạn phải quan tâm đến nhiều yếu tố trước khi quyết định chọn phần cứng và phần mềm cho những hệ thống nằm trong phạm vi trách nhiệm của mình.

Nếu những hệ thống ấy ở cự ly gần và tập trung trong cùng một toà nhà, thì mạng cục bộ (LAN) là giải pháp vừa ít tốn kém vừa có tốc độ cao. Lúc này bạn chỉ cần gắn cho mỗi chiếc máy Linux một bìa giao diện mạng Ethernet có chạy giao thức TCP/IP, bởi vì TCP/IP là giao thức chuẩn cho các bản phát hành Linux.

Để kết nối tốc độ thấp ở cự ly lớn hơn, bạn nên dùng modem thoại chạy giao thức PPP (Point–to–Point Protocol), hoặc giao thức SLIP (Serial Line Internet Protocol, giao thức Internet đường nối tiếp), cho các kết nối TCP/IP không đồng bộ. Bạn cũng có thể dùng giao thức UUCP (UNIX-toUNIX Copy Protocol) cho thư điện tử, diễn đàn News và truyền tệp FTP (mặc dù UUCP có giới hạn nhất định).

Đối với cự ly dài hơn và tốc độ cao hơn, bạn chọn công nghệ ISDN hoặc xDSL hoặc thuê đường truyền riêng (Leased Line) của một công ty viễn thông phù hợp.

Bạn không nên mua bất kỳ thiết bị mạng nào đã qua sử dụng. Một số thiết bị mạng được bán cùng với phần mềm điều khiển (driver) giúp chúng chạy với DOS, nhưng điều đó không có nghĩa là chúng sẽ chạy trơn tru với Linux. Hệ thống Linux có thể có nhiều driver chuẩn cho mạng được gắn kèm (built-in). Bảng sau liệt kê vài bìa Ethernet được Linux hỗ trợ. Bạn nên kiểm tra lại tệp HOWTO về Ethernet để theo dõi những cập nhật mới nhất.

Bảng 7.1: Vài loại bìa Ethernet được Linux hỗ trợ

Nhà sản xuất	**Bìa**
3Com	3c503, 3c503/16, 3c509, 3c579
SMC (Western Digital)	WD8003, WD8013, SMC Elite, SMC Elite Plus, SMC Elite 16 ULTRA
Novell Ethernet	NE1000, NE2000, NE1500, NE2100
D-Link	DE-600, DE-650, DE-100, DE-200, DE-220-T
Hewlett-Packard	27245A, 27247B, 27252A, 27247A, J2405A
Digital	DE200, DE210, DE202, DE100, DEPCA (rev.E)
Allied Telesis	AT1500, AT1700
PureData	PDUC8028, PI8023

Các phần mềm ứng dụng không được đóng gói chung với những sản phẩm về mạng vẫn có thể sử dụng ở môi trường mạng. Thí dụ bạn cài đặt một ứng dụng trên Linux, sau đó nhiều user trên những máy khác vẫn có thể dùng ứng dụng này bằng cách chạy các lệnh từ xa được viết cho UNIX. Một thí dụ khác: bạn có thể chia sẻ một ứng dụng bằng cách lắp ghép từ xa hệ thống tệp chứa ứng dụng ấy, sau đó lại chạy ứng dụng ngay từ hệ thống tại chỗ.

7.12 Những công việc chung trong quản trị mạng

Quản trị mạng có nhiều công việc khác nhau. Mạng máy tính không chỉ được thiết lập một lần rồi thôi mà nó luôn luôn phát triển. Khi mua phần cứng và phần mềm, quản trị viên cần hiểu rõ các user trông chờ điều gì ở mình và họ sẽ được những tiện ích nào.

7.12.1 Thiết lập hệ thống

Nếu là người dùng mạng, dù chỉ trong một phân đoạn (segment) Ethernet, bạn nên thực hiện các thử nghiệm về tính liên tác (interoperability). khi kết nối xa qua mạng điện thoại, bạn cũng nên thử nghiệm đường truyền.

Khi là quản trị viên, bạn càng cần thử nghiệm toàn bộ việc nối các cáp, thiết bị mạng và thiết bị tin học. Phần mềm mạng phải được cài đặt và sẵn sàng kết nối. Ngay cả khi hệ thống có các tiện ích Plug and Play (cắm là chạy) bạn cũng cần kiểm tra lại chúng.

Điều có lợi khi mua một máy chưa cài đặt sẵn hệ điều hành là bạn có cơ hội thiết lập hệ thống tệp sao cho phù hợp với yêu cầu riêng biệt của mình. Bạn biết hệ thống mạng sẽ dùng phần mềm nào, cũng như tổng số user và cường độ sử dụng máy của họ.

Lưu ý: để khỏi tốn nhiều thời gian và công sức vào việc thiết lập lại mạng, bạn cần sao lưu tức khắc các tệp cấu hình ngay sau khi thiết lập một hệ thống.

Khi thiết lập Linux xong và thấy hệ thống chạy tốt, bạn sẽ cài các phần mềm ứng dụng. Như đã biết, phần mềm chạy với Linux thường phức tạp hơn các hệ đơn dụng, do đó bạn nên thử những phần mềm ấy nhiều lần sau khi cài đặt để đảm bảo rằng chúng chạy trơn tru, trước hết là các tiện ích.

Đến đây, mặc dù chưa nối mạng song bạn vẫn cần chuẩn bị đưa danh sách user vào hệ thống. Bạn phải cấp các định danh đăng nhập (login ID) cho một vài user chính, cùng với mật khẩu chung ban đầu. Việc này vừa đảm bảo an ninh ban đầu, vừa tạo cơ hội cho những user quan trọng sử dụng hệ thống ngay sau khi cài đặt xong.

Sau khi cài đặt, bạn nên nối mạng ngay để chắc chắn rằng mình có thể liên lạc với mọi nơi. Tiếp theo bạn thử chuyển một số tệp lớn nhỏ khác nhau từ máy này sang máy khác. Thư điện tử phải được thông suốt. Tất cả các máy phải nhận diện được mỗi máy mới nối vào hệ thống, có nghĩa là bạn phải thêm tên của máy mới vào cơ sở dữ liệu host name. Nếu đang sử dụng dịch vụ tên miền tại chỗ, bạn phải thêm host name vào cơ sở dữ liệu DNS. Nếu không dùng DNS thì bạn thêm tên vào các tệp /etc/hosts trên các hệ thống khác.

7.12.2 Thao tác các thiết bị ngoại vi

In ấn là một thao tác phổ biến mà quản trị viên cần quan tâm chuẩn bị sẵn sàng. Theo dõi, bảo trì máy in là một công việc khá quan trọng và chiếm nhiều thời gian của quản trị viên. Người quản trị phải biết quy trình in ấn và thiết lập các cấu hình thích hợp với đặc điểm của từng loại thiết bị.

Còn modem là phương tiện ít tốn tiền nhất để nối mạng ở cự ly lớn. Modem làm việc với giao thức PPP hoặc UUCP có thể giúp một nhóm nhỏ quản trị viên trông coi được nhiều máy tính. Tuy nhiên cũng như máy in, modem cần được chăm sóc từ đầu để luôn luôn chạy tốt. Bạn nên chọn thử một vài modem có thương hiệu nổi tiếng và tìm hiểu kỹ các tính năng của chúng.

7.12.3 Giám sát hệ thống

Khi cài đặt xong UNIX/Linux, bạn có thể thiết lập các công cụ để giám sát hệ thống mới. Việc giám sát một hệ thống đang hoạt động là công việc thường trực của quản trị

viên, song gánh nặng này sẽ ổn định sau một thời gian (dĩ nhiên nếu bạn không liên tục bổ sung vào mạng nhiều máy mới hoặc phần mềm mới). Thỉnh thoảng cũng có thiết bị nào đó ngưng hoạt động hoặc cần điều chỉnh. Một quản trị viên giỏi là người biết rõ vấn đề phát sinh do phần cứng hay phần mềm và cách giải quyết.

7.12.4 Nâng cấp phần mềm

Có vài gói phần mềm trong Linux cần được cập nhật thường xuyên. Quản trị viên phải lo việc này, bởi vì Linux được cấp miễn phí trên Internet và luôn được chỉnh sửa. Việc chỉnh sửa nhằm để khắc phục những lỗi đã phát hiện, nhưng bạn lại mất thời gian tải nạp bản cập nhật và cài đặt nó trên các máy.

Bạn không nên cài đặt hết mọi phiên bản mới vào toàn bộ các máy, mà trước hết hãy thử nghiệm bản nâng cấp trên một máy. Khi nào chắc chắn rằng bản nâng cấp chạy tốt, lúc ấy hãy cập nhật nó vào những máy khác.

Quản trị viên giỏi là người biết cài đặt những phần chỉnh sửa hoặc những phiên bản mới mà không cần phải thân chinh đi đến từng máy trong mạng. Mới nghe qua điều này có vẻ khó tin, song UNIX có những công cụ giúp bạn thực hiện công tác này.

7.13 Huấn luyện quản trị viên

Thông thường các cơ quan đều có người biết sử dụng máy vi tính, nhưng lại hiếm có quản trị viên hệ thống chuyên nghiệp.

Công tác quản trị UNIX/Linux đòi hỏi có kiến thức vững chắc về một số chủ đề chung sau đây:

-**Thiết kế và sử dụng các hệ Linux/UNIX**. Quản trị viên phải có hiểu biết xuyên suốt về một số kỹ thuật như đổi hướng (redirection), ống dẫn (pipeline), xử lý hậu trường (background processing), v.v.

-**Trình soạn thảo văn bản vi**. Hầu như bất kỳ chiếc máy tính nào được cài đặt UNIX/Linux thì đều có sẵn trình soạn thảo **vi**. Nhiều người không thích **vi** và chuyển sang dùng các trình soạn thảo khác thân thiện hơn, tuy nhiên quản trị viên nên có kiến thức và kỹ năng sử dụng **vi** bởi vì đó là mẫu số chung của các hệ UNIX/Linux.

-**Lập trình shell script**. Phần lớn chương trình dùng để quản trị UNIX/Linux được viết theo ngôn ngữ kịch bản shell script nhưng bạn nên sửa đổi chúng chút ít cho phù hợp nhu cầu của mình. Nhiều công cụ được nhắc đến ở đây đòi hỏi bạn phải biết phối hợp và sử dụng các chương trình shell. Mỗi user thường thích sử dụng loại shell quen thuộc nhất của mình. Thí dụ bash là một bản nâng cấp rất mạnh từ bản Shell Bourne mà Linux dùng làm shell mặc định. Ngoài ra hai bản khác là Z shell và T shell cũng hay được kèm theo các bản phát hành Linux. Tuy nhiên ở bước đầu bạn hãy dùng Shell Bourne cho thật thuần thục. Nên biết rằng các chương trình kịch bản của những đồng tác giả Linux đều được viết trong Shell Bourne. Bạn cũng nên tìm hiểu ngôn ngữ quản trị hệ thống Perl. Ngôn ngữ này giúp bạn một bộ công cụ rất tốt để quản trị hệ thống trong môi trường lập trình.

-Liên lạc. Muốn quản trị mạng máy tính cho thật hiệu quả, điều chủ yếu là bạn phải có kiến thức về TCP/IP và các giao thức có liên quan. Tương tự, bạn sẽ phải hiểu biết về

PPP nếu muốn thiết lập một kết nối Internet không đồng bộ. Tất cả các giao thức ấy có thể học ngay tại môi trường thực hành với nhiều tuỳ chọn sẵn có. Đương nhiên bạn vẫn có thể ghi danh học tại các lớp lý thuyết hoặc tự mua sách về học, song nếu như thế bạn phải mất thêm thời gian thử nghiệm.

-Các kiến thức cơ bản về UNIX. Trong nhà trường, các kiến thức cơ bản về UNIX thường không được dạy, thậm chí còn không được đề cập đến và bạn sẽ phải nắm bắt chúng trong quá trình thực tế. Thí dụ bạn sẽ biết rằng các tệp thi hành nhị phân thường được lưu trữ tại các thư mục bin, chẳng hạn như /usr/bin, hoặc /bin, hay là /urs/local/bin. Tương tự, những thư mục lib, chẳng hạn như /usr/lib được dùng để chứa tệp thư viện, do đó bạn sẽ đưa các thư viện riêng của mình vào một thư mục đại loại như /usr/local/lib. Nắm được và làm theo những kiến thức cơ bản về UNIX/Linux, bạn sẽ tiết kiệm được thì giờ khi dò tìm và giải quyết nhiều vấn đề phức tạp hơn.

Bạn nên theo những chương trình đào tạo thực hành, tức là học đến đâu làm đến đấy. Học xong một bài nào đó ở lớp, bạn cần áp dụng ngay điều mình vừa học vào thực tiễn.

Chương 8. Trình soạn thảo văn bản vi

Có rất nhiều trình soạn thảo văn bản trong UNIX cũng như Linux. Nhưng **vi** là trình soạn thảo được nhiều chuyên gia UNIX và Linux sử dụng nhất. Có lẽ một phần là do **vi** chiếm ít tài nguyên, nhưng cũng có thể do nó tồn tại trong UNIX/Linux ngay từ đầu và không có một hệ UNIX hay Linux nào mà không chứa nó như một trình soạn thảo mặc định. Nói vậy thôi chứ việc dùng trình nào là quyền của bạn.

Các chủ đề chính sẽ được đề cập đến trong chương này bao gồm:

- Giới thiệu **vi**
- Sử dụng **vi**
- Tóm tắt các lệnh **vi**
- Thiết lập môi trường **vi**

8.1 Giới thiệu vi

Qua những chương trước, bạn đã thấy sự tiện lợi khi các chuỗi lệnh hoặc kịch bản shell cùng được chứa trong một tệp. Người dùng máy tính tạo ra dữ liệu, thư điện tử, danh sách, bản ghi nhớ, ghi chú, báo cáo, v.v.. và những công việc vừa kể đều nhờ đến một trình soạn thảo văn bản (text editor) nào đó để xử lý hay hiệu chỉnh. Linux có sẵn một số chương trình xử lý văn bản như thế.

Tối thiểu, bạn cũng cần một trình soạn thảo có khả năng lưu trữ công việc của mình vào một tệp dưới dạng mã ASCII. Linux đi kèm trình soạn thảo mang tên **vi**, mà bạn có thể dùng để hiệu chỉnh hầu như tất cả mọi thứ văn bản. **vi** rất hữu ích cho quản trị viên hệ thống vì chương trình này có sẵn trên mọi nền UNIX, do đó khi đã quen tay với **vi** rồi, bạn tha hồ thao tác mọi hệ thống chạy UNIX. Ngoài ra **vi** cũng có lợi vì chiếm rất ít tài nguyên khi thi hành lệnh. Trong khi một số chương trình khác không thể chạy được vì xung đột phần cứng hoặc vì hệ thống thiếu bộ nhớ, thì **vi** vẫn hoạt động bình thường.

Hai trình **vi** và **ex** đi kèm với bản phát hành Red Hat thực ra là hai tên khác nhau của trình soạn thảo **vim** (VI iMproved). Hai tên **vi** và **ex** gắn liền với **vim**, khi chạy **vi** gần như bạn đang chạy **vim**. Mời bạn xem tệp /urs/share/vim/vims6/doc/vi_diff.txt hay /usr/share/doc/vim-common-6.1/doc/vi_diff.txt để biết tóm tắt những điểm dị biệt giữa **vi** và **vim**.

Hiện nay phiên bản phổ biến nhất của **vi** là 6.1 nhờ có thêm nhiều tính năng tiện lợi như có thể chia màn hình ra thành nhiều cửa sổ theo chiều đứng (vertical split) trong khi bản 5.8 thì chỉ có chia theo chiều ngang (horizontal split). Hay là có thể hiển thị chữ có màu cho các từ khoá - điều này rất thuận tiện cho việc biên soạn một tệp cấu hình.

*Minh hoạ 8.1: Soạn thảo với **vi***

Linux có nhiều trình soạn thảo văn bản khác: một chương trình có giao diện đồ hoạ để dùng với XFree86 và hai chương trình phi đồ hoạ mang tên **ed** và **ex**. Cả hai chương trình phi đồ hoạ đều soạn theo dòng (line-oriented), nghĩa là bạn chỉ có thể làm việc từng dòng một.

Một chương trình khác mang tên **emacs** cũng có mặt trong các bản phát hành Linux. **vi** và **emacs** là những trình soạn thảo kiểu toàn trang (màn hình) tức là đoạn văn xuất hiện theo từng màn hình một để bạn có thể chỉnh sửa và thêm thắt tùy theo ngữ cảnh.

Chương này không bàn nhiều về **ed** hay **ex**, bởi vì bạn sẽ thấy **vi** rất dễ dùng và có sẵn trên mọi máy chạy UNIX, bao gồm cả Linux.

Minh hoạ 8.2: Chia màn hình soạn thảo

Cho dù nhiều hệ điều hành ngày nay có những trình soạn thảo mạnh hơn và dễ sử dụng, bạn vẫn nên tìm hiểu về **vi** bởi bất kỳ bản UNIX nào cũng có nó. Đôi lúc **vi** là trình soạn thảo duy nhất sẵn có vào lúc cần thiết, vì vậy bạn cần biết vài thao tác căn bản. Hơn nữa giờ đây với những phiên bản mới (v.6.x) **vi** đã trở nên thân thiện hơn và thậm chí có cả phiên bản chạy trên Windows.

Cuối thập niên 1960, tin học được triển khai trong hoàn cảnh terminal của người sử dụng là máy điện tín (teletype), một dạng bàn phím với máy in kết quả ra giấy, chạy chậm rì rì (thời đó màn hình rất ít được biết). Do đó trình soạn thảo văn bản dùng cho môi trường như thế chỉ cho phép soạn theo dòng, mà người dùng cũng chỉ thấy và chỉ làm việc được trên một dòng duy nhất mà thôi. Từ di sản đó, hiện nay các hệ UNIX vẫn còn dùng hai trình soạn thảo văn bản theo dòng là **ed** và **ex.**

Đến đầu thập niên 1970, nhiều sinh viên được dùng UNIX không mất tiền và họ đã cùng góp sức triển khai hệ điều hành này. Đại học California và Berkeley có nhiều đóng góp, trong đó có trình soạn thảo toàn trang, thay thế việc xử lý từng dòng một. Trình soạn thảo toàn trang này được gọi là **vi**, viết tắt từ chữ "Visual", do nó cho phép nhìn thấy cả đoạn văn thay vì từng dòng một. Người sử dụng làm việc với các terminal nhìn trực tiếp vào màn hình, thay vì làm việc với các thiết bị điện tín in ra giấy.

Lưu ý: Không nhất thiết phải là chuyên gia mới sử dụng được **vi**. Khi cần trợ giúp, bạn chỉ việc gõ lệnh: **man vi**. Ngoài ra bạn có thể bấm <Esc> và gõ: **help** XXXX với XXXX là từ khoá mà bạn thắc mắc (thí dụ: **help** split); nếu bạn muốn xem tổng quan thì chỉ gõ: **help**).

Ghi chú: Chương này không bàn đầy đủ mọi khía cạnh của **vi**. Ở đây bạn chỉ tìm hiểu sơ qua về một số lệnh để thực thi các thao tác soạn thảo cần thiết nhất. Muốn biết thêm những đặc trưng khác của **vi**, mời bạn xem các trang **man** đi kèm với Linux.

8.1.1 vi là gì?

Cũng như hàng triệu người khác đang sử dụng **vi**, bạn sẽ thấy ứng dụng này khởi động nhanh và có thể dùng cho việc đơn giản cũng như phức tạp. Bạn dùng **vi** để nhập văn bản, chỉnh sửa hoặc xoá bỏ văn bản. Ngoài ra bạn còn tìm kiếm và thay thế, cắt, chép và dán từng khối văn bản. **vi** cũng giúp bạn cắt xén tỉa gọt văn bản cho đúng ý mình. Bạn tuỳ nghi di chuyển con chạy đến bất kỳ chỗ nào trên màn hình cũng như trên phạm vi văn bản đang được xử lý. Trình soạn thảo **vi** chưa phải là một phần mềm xử

lý văn bản thực thụ hoặc phần mềm chế bản văn phòng, nó không có **menu** và cũng không có các tiện ích trợ giúp.

Ghi chú: Phiên bản nguyên thuỷ của **vi** không có tiện ích trợ giúp. Tuy nhiên các phiên bản sau này, chẳng hạn như **vim** của Red Hat đã có chút ít trợ giúp trực tuyến.

Các ứng dụng xử lý văn bản thường có hai định dạng, một loại trên màn hình và một dùng khi in ra, chẳng hạn như trình bày văn bản dạng in đậm, in nghiêng, hoặc gạch dưới. **vi** lại không như thế.

Các lệnh khác của Linux có thể thực thi vài điều như vừa kể, chẳng hạn như lệnh **lp** thực hành in và lệnh **nrof** có khả năng định dạng văn bản. Một vài chương trình xử lý văn bản như TeX và LaTeX, có thể xử lý các câu lệnh nhúng trong văn bản, chẳng hạn như in đậm hoặc gạch dưới.

Trình soạn thảo **vi** chạy ở hai chế độ khác nhau:

- Ở chế độ câu lệnh, những gì bạn gõ vào sẽ được hiểu như ra lệnh cho **vi**. Lệnh sẽ bảo **vi** lưu tệp, thoát khỏi **vi**, chuyển con chạy đến các vị trí khác nhau trong tệp, chỉnh sửa, sắp xếp, xoá bỏ, thay thế và tìm kiếm đoạn văn bản.

- Ở chế độ nhập liệu hoặc còn gọi là nhập văn bản (chế độ INSERT), những gì bạn gõ vào được máy hiểu là nội dung của tệp mà bạn đang chỉnh sửa. Theo chế độ này, **vi** hành động như một máy đánh chữ đơn thuần.

Trong khi chỉnh sửa văn bản, bạn tha hồ chuyển từ chế độ này sang chế độ khác, chỉ cần nhớ rằng mình đang ở chế độ nào và biết cách chuyển đổi. Chương này sẽ giới thiệu tuỳ chọn **showmode** giúp bạn biết xem **vi** đang thuộc chế độ nào. Chỉ cần tập luyện ít lâu là bạn sẽ thấy **vi** rất tiện lợi để xử lý các tệp ASCII Linux, đặc biệt là các tệp cấu hình và shell script.

8.1.2 Tiến trình soạn thảo

Soạn thảo là tạo ra văn bản mới hoặc chỉnh sửa văn bản đã có sẵn. Tạo ra văn bản mới tức là viết chữ vào một tệp được đặt tên theo dạng Linux. Khi chỉnh sửa, bạn dùng tên tệp có sẵn để gọi bản sao của tệp ấy hiển thị ra màn hình. Trong cả hai trường hợp, khi sử dụng trình soạn thảo thì bạn đã lưu trữ văn bản tại bộ nhớ của máy ở một vùng được gọi là vùng đệm.

Vùng đệm sẽ không làm thay đổi gì ở nội dung của tệp cho đến khi bạn ra lệnh lưu giữ (save) nội dung vùng đệm. Chỉ khi nào bạn ra lệnh này xong, máy sẽ ghi những điều chỉnh sửa vào đĩa cứng và những gì bạn vừa thay đổi mới có hiệu lực thật sự. Do đó nếu sau khi thao tác một lúc, bạn quyết định là không nên thay đổi gì ở nội dung cũ thì cứ để nguyên như thế. Trong một phiên làm việc, bạn muốn lưu những thay đổi ấy bao nhiêu lần cũng được và bạn nên ra lệnh lưu một cách đều đặn, phòng khi mất điện đột xuất hoặc treo máy. Khi ra lệnh lưu, bạn không phải thoát khỏi trình soạn thảo, mặc dù có nhiều cách thoát khỏi trình soạn thảo.

Trình soạn thảo **vi** có tính tương tác bởi vì chương trình sẽ "đối thoại" với bạn trong phiên làm việc. **vi** liên lạc với bạn bằng cách hiển thị tình trạng phiên làm việc, hoặc hiển thị những thông báo lỗi, hoặc đôi lúc trên màn hình không xuất hiện chữ gì cả

(đúng theo kiểu Linux). Dòng cuối cùng trên màn hình, gọi là dòng trạng thái, bao gồm những thông báo của Linux.

Bạn sử dụng trình soạn thảo để chỉnh sửa, sắp xếp, xoá bỏ, tìm kiếm và thay thế đoạn văn bản. Các thao tác chỉnh sửa như thế được tiến hành ở chế độ soạn thảo. Trong nhiều trường hợp, lệnh chỉ là ký tự đầu tiên của từ tiếng Anh chỉ hành động của lệnh. Thí dụ chữ <i> tương ứng với hành động **i**nsert (chèn vào) và <r> được dùng để **r**eplace (thay thế) một ký tự.

Hầu hết các câu lệnh đều diễn ra trên một dòng hoặc vài dòng ký tự. Dòng được đánh số từ 1 (dòng đầu tiên) cho đến dòng cuối cùng của vùng đệm.

Khi bạn thêm hoặc bớt dòng, số hiệu của dòng sẽ được tự động chỉnh lý. Số hiệu của dòng tức là địa chỉ của dòng đó trong vùng đệm.

Vùng địa chỉ là hai địa chỉ hoặc hai số hiệu của dòng được cách nhau bởi dấu phẩy. Nếu muốn xác định vùng địa chỉ từ dòng thứ ba đến dòng thứ tám của vùng đệm, bạn viết 3, 8.

Vị trí con chạy (cursor) cũng cho bạn biết vị trí hiện hành của vùng đệm soạn thảo. Một vài lệnh mà bạn gõ ở chế độ ra lệnh sẽ tác động đến ký tự ngay chỗ con chạy đang bấm nháy. Nếu bạn không chuyển con chạy, những thay đổi sẽ diễn ra tại vị trí ấy. **vi** có nhiều lệnh để di chuyển con chạy tới lui trong vùng đệm soạn thảo, bởi vì **vi** là trình soạn thảo kiểu toàn trang. Bạn ra lệnh cho **vi** di chuyển con chạy đến nhiều nơi và bạn trực tiếp nhìn thấy những thay đổi diễn ra khi bạn thao tác. Vì thế **vi** phải có khả năng di chuyển và chỉnh sửa văn bản trên terminal của chính bạn cũng như ở các loại terminal khác. **vi** biết bạn đang dùng terminal nào và khả năng màn hình của terminal ấy ra sao bằng cách xem xét một biến shell mang tên TERM. Linux dùng biến TERM để xác định xem terminal làm được gì, chẳng hạn như gạch dưới, đảo màn hình, xoá màn hình, gán phím chức năng và khả năng màu sắc của terminal.

Hỏi: Trình soạn thảo **vi** trên máy tôi hình như chưa ăn khớp với màn hình hoặc terminal, bởi vì tôi thấy xuất hiện một số ký tự lạ.

Đáp: Có thể là biến TERM chưa được lập đúng cách. Một triệu chứng khác của việc thiết lập terminal không đúng cách là từng khối, từng mảng ký tự ghi đè lên những đoạn văn bản. $TERM sẽ cung cấp giá trị của thiết lập hiện hành cho terminal.

Muốn kiểm tra trị của TERM, bạn gõ: **echo** $TERM.

Trong một số Linux khác RedHat, nếu bạn đang làm việc trên một terminal loại vt100 hoặc mô phỏng vt1000, lệnh vừa kể sẽ hiển thị kết quả như sau (xin nhớ là phải gõ lệnh trên terminal chứ không phải trong khi đang ở trong trình soạn thảo **vi**):

```
vt100
```

Nếu máy không phản hồi đúng loại terminal và nếu bạn đang dùng shell bash bạn phải lập trị của TERM như sau:

```
TERM = vt100
export TERM
```

Còn nếu đang dùng shell C, bạn gõ (và lưu ý các khoảng trống quanh dấu =):

```
setenv TERM = vt100
```

```
export TERM
```

Terminal mà bạn đang dùng có thể khác với vt100 và nên lập lại TERM cho phù hợp.

Xem thêm "Thiết lập môi trường shell".

Với Red Hat, giá trị của TERM là linux.

Trong **vi**, bạn cũng có thể định lại giá trị biến TERM bằng lệnh sau:

```
: set term = xxxx
```

Ngoài ra trên một số hệ điều hành khác, ta sẽ có giá trị TERM khác nhau, thí dụ:

```
GUI: "builtin_gui"
Amiga: "amiga"
BeOS: "beos-ansi"
Mac: "mac-ansi"
MiNT: "vt52"
DOS: "pcterm"
OS/s: "os2ansi"
Unix: "ansi"
VMS: "ansi"
Win 32: "win32"
```

Hỏi: Tôi khởi động **vi** nhưng không được phản hồi như mong đợi.

Đáp: Kiểm tra lại xem terminal của bạn được lập đúng cách chưa. Loại terminal và tên terminal của bạn không khớp nhau. Loại terminal của bạn phải khớp với một trong những loại terminal chứa trong thư mục /usr/share/terminfo.

8.2 Sử dụng vi

Bạn chỉ cần gõ **vi** tại dấu nhắc shell tại dòng lệnh để kích hoạt **vi**. Muốn tạo ra hoặc chỉnh sửa một tệp nào đó, bạn gõ lệnh **vi** chung với tên tệp. Chẳng hạn muốn tạo tệp abc với **vi**, bạn gõ **vi** abc.

Khi **vi** được kích hoạt, màn hình terminal cũ sẽ tự xoá đi, đồng thời một dấu sóng (~) sẽ xuất hiện ở phía bên trái mỗi đầu dòng, từ dòng đầu tiên hay những dòng đã soạn thảo. Dấu sóng báo hiệu dòng trống của vùng đệm. Con chạy nằm phía ngoài cùng bên trái của dòng đầu tiên. Bạn sẽ trông thấy từ 20 đến 22 dấu sóng ở phía trái màn hình. Nếu màn hình không đúng như thế, bạn nên kiểm tra lại giá trị của TERM (như đã mô tả ở phần Hỏi-Đáp phía trên) hoặc báo cho quản trị viên biết.

Còn nếu mọi sự diễn ra như mô tả, bạn đã khởi động được **vi** và **vi** đang ở chế độ chờ lệnh.

8.2.1 Hai chế độ của vi

Như đã nói ở trên, trình soạn thảo **vi** làm việc theo hai chế độ: chế độ câu lệnh và chế độ nhập liệu. Ở chế độ lệnh, những gì bạn gõ vào sẽ được **vi** hiểu là mệnh lệnh. Bạn sẽ dùng để lưu trữ, di chuyển, xoá, thay đổi v.v. Bạn cũng có thể chuyển lệnh cho shell. Khi gõ vào một ký tự để ra lệnh nhưng bản thân ký tự ấy lại không phải là ký tự lệnh,

thì **vi** sẽ kêu bip bip để bạn xem lại. Dòng cuối màn hình sẽ hiện ra vị trí của con chạy khi bạn dịch chuyển nó. Với các phiên bản cũ, bạn phải dịch chuyển bằng các phím ***j, k, l, ;*** nhưng kể từ bản 6.1, bạn đã có thể dịch chuyển bằng các phím mũi tên.

Ở chế độ nhập liệu (văn bản), bạn chèn ký tự vào phía trước hoặc thêm vào phía sau con chạy. Muốn chuyển đổi từ chế độ lệnh sang chế độ nhập liệu, bạn bấm:

- Phím <a> để thêm ký tự vào phía sau con chạy.

- Phím <i> để chèn ký tự vào phía trước con chạy.

Đa số các trình soạn thảo khác đều khởi động ở chế độ nhập liệu để bạn gõ văn bản vào ngay và khi muốn ra lệnh bạn phải dùng các phím chức năng. Trong khi đó với **vi,** bạn phải gõ <a> hoặc <i> để vào chế độ nhập liệu và sau đó nếu muốn chuyển sang chế độ lệnh, bạn bấm <Esc>.

Khi chuyển sang chế độ nhập liệu, phía cuối màn hình sẽ hiện dòng chữ "--INSERT--"

8.2.2 Tạo ra tệp vi đầu tiên

Mục này sẽ hướng dẫn bạn từng bước để dùng **vi** tạo ra một tệp. Bạn đừng quan tâm đến tính chính xác vì mục đích của thí dụ là để bạn làm quen với các thao tác và khái niệm sử dụng **vi**, việc chuyển qua lại giữa hai chế độ khác nhau và lưu kết quả soạn thảo. Khi gặp khó khăn, bạn có thể thoát khỏi ngoài và làm lại bằng cách bấm <Esc>, sau đó gõ: **q!**

1. Đầu tiên bạn gõ **vi** để kích hoạt chương trình **vi**. Bạn sẽ thấy một màn hình đầy những dấu sóng về phía trái.

2. Bạn vào chế độ nhập liệu để viết ký tự ở dòng đầu tiên. Bấm phím <a> và đừng bấm <Enter>. Bạn không nhìn thấy ký tự a, nhưng bạn cứ tiếp tục nhập liệu.

3. Hãy nhập vài dòng văn bản vào vùng đệm bằng cách gõ vào thí dụ sau đây:

```
Test vi.
a. Learn vi commands.
b. Print test file.
```

Nếu gõ sai, bạn sử dụng phím lùi (backspace) để sửa chữa trên dòng hiện hành. Những đoạn khác của chương này sẽ chỉ bạn vài cách khác để chỉnh sửa.

4. Bấm <Esc> để chuyển sang chế độ lệnh. Còn nếu đang ở chế độ lệnh rồi mà còn bấm <Esc>, máy sẽ bíp lên một tiếng để bạn biết.

5. Lưu những gì trong vùng đệm vào thông tin mang tên vi**_test**.1 (tên tệp này chỉ để làm thí dụ). Muốn thế bạn gõ

```
:w vi_test.1
```

Những ký tự **:w vi_test.1** không được xuất hiện trong nội dung văn bản mà phải hiển diện ở dòng cuối màn hình (dòng trạng thái). Lệnh **w** sẽ ghi nội dung vùng đệm vào tệp vi**_test.**1.

6. Nhìn dòng trạng thái ở cuối màn hình xem hành động của bạn đã được xác nhận chưa, nếu có bạn sẽ thấy:

```
vi_test.1 [New file] 3 lines, 48 characters
```

Câu khai báo này xác nhận rằng tệp mang tên vi_**test**.1 đã được tạo xong, đó là một tệp mới tạo, chứa ba dòng gồm 48 ký tự.

7. Gõ **:q** để thoát khỏi **vi.**

Khi gõ **:q** bạn vẫn đang ở chế độ lệnh và sẽ trông thấy **:q** tại dòng trạng thái. Tuy nhiên khi bấm <Enter> **vi** sẽ chấm dứt hoạt động và bạn trở về dấu nhắc đăng nhập của shell.

Bạn theo các bước vừa mô tả bên trên để thực hiện công việc soạn thảo và chỉnh sửa văn bản. Trước khi tiếp tục, bạn nên thử làm lại cho nhuần nhuyễn.

Những điều cần nhớ về **vi:**

-Trình **vi** bắt đầu khởi động ở chế độ lệnh.

-Từ chế độ lệnh bạn chuyển sang chế độ nhập liệu bằng cách bấm <a> (để thêm ký tự vào sau con chạy) hoặc <i> (để chèn ký tự vào trước con chạy).

-Chế độ nhập liệu cho phép bạn viết văn bản.

-Chế độ lệnh cho phép bạn ra lệnh.

-Bạn ra lệnh cho **vi** để lưu tệp và chỉ có thể thoát khỏi **vi** một khi đang ở chế độ lệnh.

-Từ chế độ nhập liệu, muốn chuyển sang chế độ lệnh, bạn bấm <Esc>.

8.2.3 Sử dụng một tệp có sẵn để kích hoạt vi

Muốn chỉnh sửa hoặc xem nội dung một tệp sẵn có trong thư mục hiện hành, bạn chỉ việc gõ **vi** và tên tệp. Bạn thao tác thử với tệp vừa được tạo ra ở đoạn trên bằng cách gõ vào lệnh sau:

```
vi vi_test.1
```

Bạn sẽ thấy màn hình hiển thị như sau (số lượng dòng trống với dấu sóng ở đây được xóa bớt đi, ít hơn số dòng thật bạn sẽ thấy trên màn hình):

```
Test vi.
a. Learn vi commands.
b. Print test file.
~
~
vi_test.1 3 lines, 48 characters
```

Như đã nói bên trên, ký tự sóng sẽ xuất hiện ở lề trái mỗi dòng trống tại vùng đệm. Bạn hãy nhìn vào dòng trạng thái: dòng này cho biết tên tệp bạn đang chỉnh sửa và số dòng cùng với số ký tự của tệp.

> ***Hỏi:*** *Tôi gõ vào tên một tệp mà tôi biết là đã có sẵn, nhưng* **vi** *lại làm như tôi đang tạo ra một tệp mới.*

Đáp: Có thể là bạn đã gõ vào tên một tệp không có thực trong thư mục hiện hành.

Hỏi: *Tôi định chỉnh sửa một tệp, song* ***vi*** *hiển thị lời nhắn là tôi không được phép, sau đó tôi lại thấy dấu nhắc hiện ra.*

Đáp: Bởi vì bạn không được phép sửa tệp ấy! Ngoài ra, bạn cũng không thể chỉnh sửa một thư mục. Có nghĩa là nếu bạn gõ **vi** tên-thư-mục, với tên-thư-mục là tên của một thư mục nào đó, thì **vi** sẽ báo bạn biết bạn vừa mở một thư mục ra nhưng không để cho bạn chỉnh sửa thư mục ấy. Nếu bạn định chạy thử **vi** với một tệp thi hành (dạng nhị phân, không phải là tệp ASCII), bạn sẽ thấy màn hình hiện ra toàn là ký tự lạ lùng (đây là những ký tự điều khiển). Bạn sẽ không đọc và không chỉnh sửa được gì cả. **vi** ngầm hiểu rằng tệp được lưu trữ dưới dạng dòng.

Hỏi: *Tôi mở một tệp bằng* ***vi****, song máy báo là dòng quá dài.*

Đáp: Như thế có nghĩa là bạn định mở một tệp dữ liệu có dạng một chuỗi duy nhất rất dài những byte nối tiếp nhau. Bạn vẫn có thể chỉnh sửa tệp này, song với nguy cơ rất lớn là làm hỏng nó.

Hỏi: *Tôi mở một tệp bằng* ***vi****, song màn hình hiển thị vài ký tự gì đó rất kỳ lạ.*

Đáp: Có thể bạn dùng **vi** để mở một tệp được tạo ra trước đó bởi một chương trình xử lý văn bản.

8.2.4 Thoát khỏi vi

Trong tất cả các trường hợp vừa kể trên, bạn hãy thoát khỏi **vi** và trở về dấu nhắc shelll bằng cách bấm <Esc> để trở lại chế độ lệnh, sau đó gõ vào **q!** để thoát khỏi **vi** mà không thay đổi nội dung tệp.

Bảng 8.1: Những cách thoát khỏi ***vi***

Lệnh	Hành động
:q	Thoát khỏi và không thay đổi gì cả ở vùng đệm, hoặc thoát khỏi sau khi vùng đệm đã được thay đổi và lưu vào tệp.
:q!	Thoát khỏi và không lưu lại bất kỳ thay đổi nào trong vùng đệm, so với lần vùng đệm được lưu vào tệp gần đây nhất.
:wq, :x, hoặc zz	Ghi nội dung vùng đệm vào tệp và thoát khỏi.

Ghi chú: Xin nhớ rằng muốn thoát khỏi **vi**, bạn phải ở chế độ lệnh. Muốn chuyển sang chế độ lệnh, hãy bấm <Esc>. Còn nếu đang ở chế độ lệnh và còn bấm <Esc> thì máy sẽ kêu bip bip.

Hãy sử dụng **vi** để thực hành chỉnh sửa tệp vi_**test**.1 vừa được tạo ra lúc nãy. Bạn gõ vào **vi** vi_**test**.1 là màn hình hiển thị tương tự như sau:

```
Test vi.
a. Learn vi commands.
b. Print test file.
~
vi_test.1 3 lines, 48 characters
```

Màn hình báo như thế là vì thoạt tiên **vi** ghi nội dung vùng đệm vào tệp vi_**test**.1 và thoát khỏi.

Bạn hãy khởi động **vi** lần nữa với cùng tệp (gõ **vi** vi_**test**.1) và bạn sẽ thấy như sau:

```
Test vi.
a. Learn vi commands.
b. Print test file.
~
vi_test.1 3 lines, 48 characters
```

Và mặc dù **vi** khởi động ở chế độ lệnh, nhưng để cho chắc ăn, bạn hãy bấm <Esc>. Rồi bây giờ bạn hãy bấm phím trống (space bar) của bàn phím sao cho con chạy di chuyển đến phía dưới dấu chấm câu sau chữ vi ở dòng đầu tiên. Hãy thay dấu chấm câu bằng dấu chấm than bằng cách bấm <**r**> và gõ **!**. Bây giờ dòng đầu sẽ trông giống như sau:

```
Test vi!
```

Và bởi vì bạn đã thay đổi nội dung vùng đệm, **vi** sẽ không cho phép bạn thoát khỏi trừ khi bạn ra lệnh lưu lại những thay đổi ấy, hoặc ra lệnh thoát khỏi và không thay đổi gì cả. Nếu bạn thoát **vi** bằng cách gõ **:q** thì **vi** sẽ hiển thị lời nhắn như sau để bạn nhớ rằng bạn vẫn chưa ghi tệp vào đĩa kể từ khi nội dung đã bị thay đổi:

```
No write since last change (:quit! overrides)
```

Còn nếu bạn muốn vứt bỏ tất cả các thay đổi với tệp (nghĩa là không thay đổi gì cả) hãy gõ **:q!** để thoát khỏi. Muốn lưu lại những thay đổi trong nội dung, bạn gõ **:wq** hoặc lệnh tương đương (**zz** hoặc **:x**).

Ghi chú: **vi** không sao lưu bản cũ (backup) của các tệp. Sau khi gõ **:wq**, tệp nguyên thuỷ bị thay đổi và không thể phục hồi như cũ. Bạn phải tự sao lưu các tệp **vi**.

Xem "Thực hiện sao lưu và phục hồi tệp".

Xin nhắc bạn là đừng phóng tay sử dụng lệnh **:q!** bởi vì mọi thay đổi nội dung tệp sẽ mất hết, không còn lưu lại. Do đó thay vì ra lệnh **:q!** tốt hơn nên lưu tệp thành một tệp mang tên khác. Bạn sẽ gặp chủ đề này ở mục "lưu một tệp mới".

8.2.5 Hoàn nguyên (undo)

Với trình **vi**, bạn có thể hoàn nguyên (phục hồi nguyên thuỷ) những gì có trước khi thao tác vừa rồi được thực hiện, hoặc hoàn nguyên những thay đổi ở vùng đệm khi bạn chưa ghi những thay đổi ấy vào tệp. Bạn thực hiện việc hoàn nguyên ở chế độ lệnh. Thí dụ bạn vô tình xoá đi một dòng văn bản, thay đổi một điều gì đó, hoặc viết sai. Muốn hoàn nguyên, bạn bấm <Esc> để chuyển sang chế độ lệnh rồi sau đó bấm <u>, mọi thứ sẽ được giữ nguyên như lúc bạn chưa thay đổi gì cả.

Sau đây là một thí dụ sử dụng lệnh hoàn nguyên. Bạn lại mở tệp vi_**test**.1:

```
Test vi.
a. Learn vi commands.
b. Print test file.
~
```

```
vi_test.1 3 lines, 48 characters
```

Bạn thử di chuyển con chạy đến dòng thứ nhất, thêm cụm từ "**for** 60 minutes" vào giữa chữ **vi** và dấu chấm rồi bấm <Enter>.

Vào thời điểm con chạy xuất hiện phía dưới ký tự đầu tiên của dòng thứ nhất, bạn bấm phím trống (space bar) nhiều lần để con chạy chạy đến dấu chấm ngay sau chữ **vi**. Bạn bấm <**i**> để chuyển sang lệnh nhập rồi gõ cụm từ "**for** 60 minutes" vào. Bấm <Esc> để trở về chế độ lệnh. Màn hình hiển thị như sau:

```
Test vi for 60 minutes.
a. Learn vi commands.
b. Print test file.
~
vi_test.1 3 lines, 63 characters
```

Giả sử bạn chợt thấy thử **vi** suốt "60 phút" là quá nhiều nên không thích để cụm từ mới "**for** 60 minutes" nữa và muốn hoàn nguyên như trước khi viết thêm nó, hãy bấm <Esc> để chuyển sang chế độ lệnh rồi bấm <**u**>. Giờ đây dòng này trở lại như cũ:

```
Test vi.
```

Sau đó, biết đâu bạn lại đổi ý và muốn giữ lại cụm từ kia? Bạn lại bấm <**u**> lần nữa và cụm từ "**for** 60 minutes" lại xuất hiện. Bạn sử dụng lệnh hoàn nguyên bao nhiêu lần cũng được, để hoàn nguyên sự thay đổi hoặc hoàn nguyên chính hành động hoàn nguyên. Ngay cả khi bạn thoát khỏi và không thay đổi gì ở vùng đệm, **vi** vẫn cho rằng vùng này đã có gì đó thay đổi, do đó bạn phải thoát khỏi bằng **:q!** hoặc bằng **:wq**.

Nếu muốn lưu tệp với những thay đổi vừa qua thành một tệp khác, bạn gõ **:**

w vi**_test**.2.

Lưu ý: Bạn có thể dùng phím lùi <Backspace> để sửa chữa các ký tự khi đang đánh máy một dòng. Tuy nhiên khi thụt lùi thì bạn phải xoá đi những ký tự ngay trước đó. Phím mũi tên chỉ trái <[redacted]không xoá ký tự. Sau này ta sẽ bàn về các phím mũi tên.

8.2.6 Ghi vào tệp và lưu vùng đệm

Bạn đã biết cách ghi nội dung vùng đệm vào tệp rồi thoát khỏi **vi**. Nhưng đôi khi bạn cũng muốn lưu vùng đệm vào tệp mà không thoát khỏi **vi**. Như đã nói ở phía trên, bạn nên thường xuyên lưu tệp trong khi soạn thảo, phòng khi mất điện đột xuất hay sụp mạng. Để lưu vùng đệm, bạn dùng lệnh **:w** với các biến thể như trong bảng sau.

Bảng 8.2: Những lệnh để lưu hoặc ghi một tệp

Lệnh	Hành động
:w	Ghi nội dung vùng đệm vào tệp mà **vi** đang hiệu chỉnh.
:w tên_tệp	Ghi nội dung vùng đệm vào tệp đã có sẵn tên.
:w! tên_tệp	Ghi đè lên một tệp sẵn có hay tệp có thuộc tính chỉ-đọc (read-only mode).

Bạn lưu một tệp bằng nhiều cách và dùng lệnh ghi tuỳ theo từng trường hợp. Các mục sau đây mô tả bốn trường hợp khác nhau.

8.2.6.1 Lưu một tệp mới

Khi khởi động **vi**, nếu muốn lưu một tệp vào đĩa thì bạn đặt tên cho tệp ấy. Do đó bạn phải ra lệnh ghi như sau:

```
:w tên_tệp
```

Lệnh này bảo **vi** ghi nội dung vùng đệm vào tệp tên_tệp. Nếu hành động này hoàn tất, bạn sẽ thấy tên tệp cùng với số dòng và số ký tự trong tệp. Nếu trong câu lệnh bạn lại gõ tên một tệp sẵn có trong máy, dòng trạng thái sẽ hiển thị:

```
File exists - use w! filename to overwrite.
```

Xem mục "Ghi đè lên một tệp sẵn có"

8.2.6.2 Lưu vào tệp hiện hành

Thí dụ bạn khởi động **vi** với một tệp sẵn có, thay đổi gì đó trong tệp, sau đó muốn lưu lại. Bạn chỉ việc gõ :w. Lệnh này sẽ ghi nội dung vùng đệm vào tệp hiện hành. Dòng trạng thái sẽ báo bạn biết tên tệp, số dòng và số ký tự được ghi vào tệp.

8.2.6.3 Lưu như một tệp mới

Thí dụ bạn bắt đầu làm việc với vi_**test**.1, ghi chú gì vào đó rồi muốn lưu như là một tệp mới tạo, trong khi vẫn để nguyên vi_**test**.1. Bạn thực hiện động tác "Lưu như" (Save as) bằng cách gõ thêm tên tệp mới:

```
:w tên_tệp_2
```

Lệnh ghi này giống như lệnh ghi mô tả ở mục trên. Nội dung vùng đệm được ghi vào tệp mang tên tên_tệp_2. Khi hành động này hoàn tất, bạn sẽ thấy tên tệp cùng với số dòng và số ký tự trong tệp. Nếu trong câu lệnh bạn gõ tên một tệp đã sẵn có trong máy, dòng trạng thái sẽ hiển thị:

```
File exists – use ! to overwrite.
```

Mời bạn xem giải thích ở mục tiếp theo.

8.2.6.4 Ghi đè lên một tệp sẵn có

Khi bạn ra lệnh lưu nội dung vùng đệm vào tệp sẵn có trong máy và khác với tệp hiện hành, bạn phải báo cho **vi** biết là bạn muốn ghi đè, hoặc thay thế tệp sẵn có ấy. Nếu khi lưu nội dung vùng đệm mà bạn lại gõ tên một tệp đã sẵn có trong máy, dòng trạng thái sẽ hiển thị:

```
File exists - use ! to overwrite.
```

Lúc này nếu thực sự muốn ghi đè lên tệp sẵn có, bạn gõ:

```
:w! tệp_sẵn_có
```

Với cú pháp này, tệp_sẵn_có là tên tệp mà bạn muốn thay thế. Hãy cẩn thận, bởi vì một khi đã ghi đè lên thì bạn sẽ không thể phục hồi nguyên thuỷ một tệp được.

8.2.7 Định vị con chạy

Phải di chuyển con chạy đến một vị trí nào đó để thực hiện các thao tác soạn thảo. Những câu lệnh mà bạn gõ vào, khi đang ở chế độ lệnh để chọn vùng chỉnh sửa, được gọi là những câu lệnh dùng để định vị con chạy.

8.2.7.1 Các phím mũi tên

Trên nhiều hệ điều hành, bạn có thể dùng các phím mũi tên để định vị con chạy.

Những phím này cùng với các phím trang trước <Page Up> và trang tiếp <Page Down> đều được Linux chấp nhận, với điều kiện là terminal mà bạn đang dùng phải phù hợp với các biến môi trường TERMCAP đã khai báo.

Mời bạn thực hành thí dụ sau đây về các lệnh định vị con chạy: hãy tạo một tệp mang tên vi**_test**.3 chứa danh sách những tệp và thư mục thứ cấp dưới thư mục /usr. Bạn gõ:

```
ls /usr >vi_test.3
```

Sau khi tệp được tạo ra xong, bạn kích hoạt **vi** với tệp vi**_test**.3, tiếp theo bạn sử dụng các phím mũi tên cùng với <Page Up> và <Page Down> để di chuyển quanh vùng chỉnh sửa. Đôi khi xảy ra trường hợp các phím mũi tên tạo ra nhiều ký tự lạ trong tệp. Để kiểm tra xem tình trạng này có diễn ra không, bạn bấm <Esc> để vào chế độ lệnh rồi gõ **:q** nếu trình **vi** để cho bạn thoát khỏi mà không báo rằng tệp đã thay đổi thì có nghĩa là mọi thứ đều ổn.

Lưu ý: **vi** cho phép bạn xoá các ký tự lạ khỏi màn hình bằng cách bấm <Ctrl-l>**.**

8.2.7.2 Các phím di chuyển gần

Bạn vẫn có thể định vị con chạy trong **vi** mà không cần dùng đến các phím mũi tên. Mục này giới thiệu bạn vài cách định vị con chạy còn hiệu quả hơn là dùng các phím mũi tên.

Trước đây khi **vi** mới được phát triển, nhiều terminal còn chưa có các phím mũi tên, các phím h, j, k và l đã được chọn dùng để định vị con chạy vì chúng khá thuận tay cho những người sử dụng bàn phím. Thoạt đầu có lẽ bạn phải mất một ít thời gian với những phím ấy, song những người sử dụng **vi** có kinh nghiệm đều thích dùng chúng hơn phím mũi tên. Sau đây là một số phím như thế:

-Phím <l> hoặc phím trống di chuyển con chạy sang phải một ký tự.

-Phím cộng <+> hoặc <Enter> chuyển con chạy về đầu dòng tiếp theo (nếu bấm <j>, bạn sẽ xuống dòng dưới nhưng vẫn ở cột cũ).

-Phím trừ <-> chuyển con chạy về đầu dòng trên nó (nếu bấm <k>, bạn sẽ lên dòng trên nhưng vẫn ở cột cũ).

-Phím <h> chuyển con chạy sang trái một ký tự.

-Phím số <0> chuyển con chạy đến đầu dòng.

-Phím đô-la <$> chuyển con chạy đến cuối dòng.

Một số lệnh khác của **vi** cho phép bạn định vị con chạy tương ứng với những từ (word) trên một dòng. Một từ được hiểu như là một chuỗi các ký tự được phân cách với các từ

khác bằng dấu trống (dấu trống) hoặc các dấu ?, - . (hỏi, phảy, trừ, chấm). Những lệnh ấy sử dụng các phím sau:

- Phím <w> dịch con chạy về phía trước một từ (tức là khởi điểm của từ tiếp theo).

- Phím <b> dịch con chạy về khởi điểm (ký tự đầu) của từ hiện hành.

- Phím <e> dịch con chạy về ký tự cuối của từ hiện hành.

Bạn hãy thực hành bằng cách gõ lệnh sau:

```
vi vi_test.1
```

để mở tệp vi_**test**.1, sau đó di chuyển con chạy (được quy ước bằng dấu gạch dưới) đến chữ "t" trong từ "**test**" của dòng thứ ba. Màn hình hiển thị đại loại như:

```
b. Print test file.
```

Muốn di chuyển đến điểm đầu của từ tiếp theo, bạn bấm <w>, con chạy sẽ nằm dưới ký tự "f" của từ "file". Sau đó bạn thử bấm <e> để dịch về điểm cuối của từ ấy, nghĩa là ký tự "e". Bạn lại chuyển về điểm đầu của từ đó bằng cách bấm <b>.

Bạn có thể tiến về phía trước nhiều từ, cho đến khởi điểm của một từ nào đó bằng cách bấm phím số tương ứng số từ, trước khi bấm <w>. Thí dụ con chạy hiện nay đang ở dưới ký tự "P" của từ "Print", nếu bạn muốn tiến lên thêm hai từ nữa (dưới ký tự "f" của từ "file") thì hãy bấm <2><w>.

Tương tự như thế, bạn lùi bốn từ bằng cách bấm <4><b>. Còn bấm <2><e> sẽ đưa con chạy tiến tới điểm cuối của từ thứ nhì.

Bạn có thể áp dụng phím số như thế với cả các phím h, j, k, l, - và +. Thí dụ bấm <5><j> sẽ đưa con chạy xuống 5 dòng. Nếu vùng đệm không có đủ 5 dòng thì máy sẽ kêu bip bip, đồng thời con chạy đứng yên tại cột hiện hành.

8.2.7.3 Các phím di chuyển xa

Bạn có thể định vị con chạy đến đầu, giữa, hoặc cuối màn hình. Mỗi lần như thế con chạy sẽ chỉ xuất hiện ở đầu dòng. Những phím lệnh sau đây giúp bạn định vị con chạy trên màn hình:

- Phím <Shift-h> đưa con chạy về dòng đầu màn hình (**h**igh).

- Phím <Shift-m> đưa con chạy đến dòng ở giữa (**m**edium) các dòng đang hiển thị.

- Phím <Shift-l> đưa con chạy đến dòng cuối màn hình (**l**ow).

Ngoài ra những phím lệnh sau đây còn đưa con chạy di chuyển xa hơn nữa:

- Phím <Ctrl-b> đưa con chạy lui về (**b**ack) một màn hình trước đó.

- Phím <Ctrl-f> đưa con chạy tới màn hình tiếp theo (**f**ollow).

- Phím <Shift-g> di chuyển đến dòng cuối cùng của tệp (trong vùng đệm).

Muốn di chuyển đến một dòng xác định trong vùng đệm, bạn gõ số hiệu của dòng đó trước khi bấm <Shift-g>. Thí dụ muốn chuyển đến dòng 83 của tệp, bạn bấm <8><3><Shift-g>. Còn bấm <1><Shift-g> sẽ đưa bạn đến dòng đầu tiên của tệp.

8.2.8 Thêm ký tự vào văn bản

Muốn viết thêm ký tự vào vùng đệm soạn thảo, thoạt tiên bạn phải chuyển từ chế độ lệnh sang chế độ nhập. Đang khi ở chế độ nhập, mỗi lần bạn bấm <Enter> **vi** sẽ "mở", nghĩa là thêm một dòng vào vùng đệm. Trước khi gõ thêm ký tự vào, hãy nhớ di chuyển con chạy đến ngay chỗ mà bạn muốn viết thêm. Bạn bấm <a> để sang chế độ nhập và bắt đầu viết thêm ký tự vào phía sau vị trí con chạy. Bấm <i> sẽ cho phép chèn ký tự vào phía trước con chạy. Sau khi gõ xong, bạn bấm <Esc> để trở lại chế độ lệnh.

Sau đây là hai thí dụ để thực hành gõ ở chế độ nhập (vị trí con chạy được tượng trưng bằng dấu gạch dưới).

a) Sử dụng <i> để chèn ký tự:

Câu nguyên thuỷ:

```
This report is important.
```

Bạn bấm <i> để chèn ký tự vào phía trước từ "important", gõ từ "very", bấm phím trống <Spacebar> và bấm <Esc>.

Lúc này câu cũ được chỉnh lại như sau:

```
This report is very_important.
```

Bạn để ý là con chạy hiện đang ở vị trí sau cùng mà bạn thao tác và trong trường hợp này đó là khoảng trống.

b) Sử dụng <a> để thêm ký tự:

Câu nguyên thuỷ:

```
This report is important.
```

Bấm <a> để thêm ký tự phía sau từ "is", rồi bấm phím trống, gõ từ "very", sau đó bấm <Esc>. Câu được chỉnh sửa sẽ là:

```
This report is very important.
```

Con chạy ở vị trí sau cùng mà bạn vừa thao tác và trong trường hợp này là nằm dưới ký tự "y" của từ "very". Khi muốn thêm ký tự vào cuối dòng, bạn chuyển con chạy đến cuối dòng và bấm <a>. Ngoài ra nếu muốn thêm ký tự vào cuối dòng khi con chạy đang ở bất kỳ đâu giữa dòng, bạn chỉ cần ra lệnh <Shift-a> để con chạy chạy về cuối dòng và chuyển sang chế độ nhập liệu. Tương tự như thế, chỉ với một lệnh <Shift-i>, bạn sẽ ở đầu dòng hiện hành và có thể chèn ký tự trong chế độ nhập liệu.

Muốn chèn một dòng ký tự vào phía trên hoặc phía dưới dòng hiện hành, bạn bấm <Shift-o> hoặc <o>. Mỗi lần gõ, các phím đó sẽ "**o**pen" (mở) một dòng trong vùng đệm và cho phép chèn ký tự. Hai thí dụ sau minh hoạ cách chèn dòng ký tự:

a) Dùng <o> để chèn dòng dưới dòng hiện hành:

Câu trước khi sửa:

```
All jobs complete
please call
if you have any questions.
```

Giả sử con chạy đang ở dòng thứ hai. Bạn bấm <o> để thêm ký tự vào dưới dòng ấy. Bạn hãy gõ những ký tự sau:

```
Phan Lan Anh, tel. 955-1837,
```

Rồi bấm <Esc>. Câu sau khi sửa sẽ là:

```
All jobs complete
please call
Phan Lan Anh, tel. 955-1837,
if you have any questions.
```

b) Dùng <Shift-o> để chèn dòng trên dòng hiện hành:

Câu trước khi sửa:

```
All jobs complete
please call
if you have any questions.
```

Giả sử con chạy đang ở dòng thứ ba. Bạn bấm <Shift-o> để thêm một hoặc nhiều dòng phía trên dòng này. Đến đây bạn gõ:

```
Phan Lan Anh, tel. 955-1837,
```

Bấm <Esc>. Câu sau khi sửa sẽ là:

```
All jobs complete
please call
Phan Lan Anh, tel. 955-1837,
if you have any questions.
```

Trong cả hai trường hợp, khi bạn bấm <Esc> con chạy sẽ nằm dưới ký tự sau cùng vừa gõ (số 7>. Ở thí dụ này bạn chỉ thêm vào hai dòng thôi. Muốn thêm nhiều dòng bạn cứ bấm <Enter> ở cuối mỗi dòng. Nhớ bấm <Esc> để chuyển sang chế độ lệnh trước khi ra lệnh.

Bảng 8.3: Các lệnh chèn ký tự vào văn bản

Phím	Hành động
<a>	Thêm ký tự vào phía sau vị trí con chạy
<Shift-a>	Chuyển sang chế độ nhập liệu và thêm ký tự vào cuối dòng hiện hành
<i>	Chèn ký tự vào phía trước vị trí con chạy
<Shift-i>	Chuyển sang chế độ nhập liệu và chèn ký tự vào cuối dòng hiện hành
<o>	Chèn dòng mới vào phía dưới dòng hiện hành
<Shift-o>	Chèn dòng mới vào phía trên dòng hiện hành

8.2.9 Xoá ký tự văn bản

Muốn xoá ký tự bạn phải ở chế độ lệnh. Còn đang ở trong chế độ nhập liệu mà bạn quên và ra lệnh xoá, thì những phím lệnh bạn gõ vào sẽ được máy ghi vào vùng đệm

như là một chuỗi ký tự văn bản. Nếu việc này xảy ra, bạn hãy bấm <Esc> để chuyển sang chế độ lệnh và bấm <u> để hoàn nguyên việc đó.

Với **vi** bạn có thể xoá một ký tự, một từ, một số từ đi liền nhau, xoá tất cả văn bản cho đến cuối dòng, hoặc xoá nguyên một dòng. Vì **vi** là trình soạn thảo hiển thị cho nên tất cả các ký tự, từ, dòng đều biến mất khỏi màn hình khi bạn xoá chúng.

Bảng 8.4: Các lệnh xoá ký tự khỏi văn bản

Phím	Hành động
<x>	Xoá ký tự tại vị trí con chạy
<d><w>	Xoá ký tự tại vị trí con chạy của từ hiện hành cho đến khởi điểm của từ tiếp theo
<d><$>	Xoá từ vị trí con chạy cho đến cuối dòng
<Shift-d>	Cũng như trường hợp <d><$>: xoá phần còn lại của dòng hiện hành
<d><d>	Xoá toàn bộ dòng hiện hành, cho dù con chạy đang ở vị trí nào.

Tất cả các lệnh kể trên đều có hiệu lực từ vị trí con chạy. Bạn chuyển con chạy đến ký tự, từ, hoặc dòng muốn tác động, sau đó ra lệnh cần thiết. Bạn hãy thực hành những lệnh ấy cho quen.

Lệnh cũng có thể áp dụng với việc gõ thêm con số, như đã mô tả ở mục trước. Sau đây là vài thí dụ:

- Bấm <4><x> để xoá bốn ký tự.

- Bấm <3><d><w> để xoá ba từ.

- Bấm <8><d><d> để xoá tám dòng.

Lưu ý: Muốn **vi** hiển thị số hiệu của dòng, bấm <Esc> để vào chế độ lệnh, tiếp theo gõ **:se number**. Muốn làm ẩn số hiệu các dòng, gõ **:se nonumber.**

Bạn có thể xác định số lượng dòng cần phải xoá. Bạn bấm <Shift-;>, gõ số dòng từ bao nhiêu đến bao nhiêu (có dấu phẩy phân cách), sau đó bấm <d>, rồi bấm <Enter>. Thí dụ muốn bỏ các dòng từ 12 đến 36 (kể cả các dòng kê khai), bạn gõ :12,36d và bấm <Enter>.

Khi bạn xoá hai dòng trở lên, dòng trạng thái sẽ thông báo có bao nhiêu dòng bị xoá. Nên nhớ là bạn có thể bấm <u> để hoàn nguyên việc xoá.

8.2.10 Tìm kiếm

Cũng như nhiều trình soạn thảo và xử lý văn bản khác, **vi** có câu lệnh giúp bạn tìm kiếm một chuỗi ký tự. Từ vị trí con chạy, bạn có thể tìm theo hướng tới trước, ngược về sau, tìm từ đầu vùng đệm cho đến cuối vùng hoặc ngược lại. Bảng 8.5 tóm tắt những câu lệnh tìm kiếm. Khi thực hành, **vi** sẽ dò ra chuỗi ký tự cần tìm, sau đó đặt con chạy vào khởi điểm của chuỗi ký tự được tìm thấy.

Bảng 8.5: Các lệnh truy tìm chuỗi ký tự

Phím	Hành động

/chuỗi	Tìm hướng tới trước trong vùng đệm đến khi gặp chuỗi ký tự
?chuỗi	Tìm ngược về sau trong vùng đệm đến khi gặp chuỗi ký tự
<n>	Tìm lại lần nữa theo hướng hiện hành
<Shift-n>	Tìm lại lần nữa theo hướng ngược lại

Khi bạn gõ lệnh tìm kiếm chuỗi ký tự, lệnh sẽ xuất hiện tại dòng trạng thái.

Thí dụ muốn tìm chuỗi ký tự sales>100k trong một tệp, bạn phải đang ở chế độ lệnh và gõ:

/sales>100k

Câu lệnh này xuất hiện ở vùng trạng thái. Nếu chuỗi ký tự kể trên có mặt trong vùng đệm, **vi** sẽ đặt con chạy ngay dưới ký tự s đầu tiên của từ "sales". Bằng không, **vi** sẽ hiển thị "Pattern not found" ở dòng trạng thái.

Muốn tìm lần hiện diện tiếp theo của chuỗi ký tự trong vùng đệm, bạn bấm <n>. **vi** sẽ đặt con chạy dưới lần xuất hiện tiếp theo ấy, hoặc nếu không có lần xuất hiện tiếp theo, con chạy sẽ không di chuyển.

> ***Hỏi:*** *Tôi gõ vào một chuỗi mà tôi biết là có hiện diện trong tệp, nhưng* ***vi*** *lại không tìm ra.*

Đáp: Lỗi phổ biến nhất trong trường hợp này là bạn gõ sai chuỗi ký tự. Bạn hãy kiểm tra lại trước khi bấm <Enter>.

> ***Hỏi:*** *Tôi tìm kiếm một câu, trong câu đó có dấu phân cách, nhưng* ***vi*** *lại cho tôi kết quả không như ý.*

Đáp: Nếu trong chuỗi muốn tìm có một ký tự "đặc biệt" đối với **vi** thì kết quả nhiều khi không như ý. Thí dụ bạn muốn tìm một từ mà bạn biết nằm ở cuối câu (chẳng hạn như "end."), bạn không nên gõ dấu chấm câu vào chuỗi ký tự tìm kiếm. Đối với **vi,** dấu chấm có giá trị là "bất cứ ký tự nào", chứ không phải như chúng ta hiểu đó là "hết câu". Nếu bạn gõ /end. và bấm <Enter> **vi** sẽ đi tìm những hình thức như "ending", hoặc từ "end" có khoảng trống theo sau, hoặc từ "end" có dấu chấm theo sau. Trường hợp bạn vẫn muốn tìm chuỗi ký tự "end" có dấu chấm theo sau, bạn phải gõ /end\.

Công tác tìm kiếm với **vi** cũng mang tính "case-sensitive", tức là phải phân biệt chữ hoa và chữ thường. Nếu muốn tìm từ "Tiger" trong vùng đệm, bạn đừng gõ /tiger, mà phải gõ /Tiger.

8.2.11 Thay đổi và thay thế ký tự văn bản

Những lệnh thay đổi và thay thế của **vi** giúp bạn thay một ký tự, một từ hoặc phần còn lại của một dòng.

Bảng sau đây tóm tắt các lệnh ấy. Sau khi ra lệnh, bạn chỉ việc gõ thêm những ký tự muốn thay vào đó.

Bảng 8.6: Các lệnh thay đổi và thay thế

Phím	Hành động

<r>	Thay thế một ký tự
<Shift-r>	Thay thế một chuỗi ký tự
<c><w>	Thay từ hiện hành từ vị trí con chạy đến cuối từ ấy
<c><e>	Thay từ hiện hành từ vị trí con chạy đến cuối từ ấy (giống như <c><w>
<c><b>	Thay từ hiện hành từ ký tự đầu tiên của nó đến ký tự đứng trước vị trí con chạy
<c><$>	Thay một dòng từ vị trí con chạy đến cuối dòng
<Shift-c>	Thay một dòng từ vị trí con chạy đến cuối dòng (giống như <c><$>
<c><c>	Thay hết cả dòng hiện hành

Những thay đổi sẽ diễn ra dựa vào vị trí con chạy. Hãy nhớ chuyển sang chế độ lệnh trước khi gõ lệnh vào. Đặt con chạy ở vị trí muốn thay đổi rồi bấm <Esc> trước khi gõ lệnh.

Mỗi câu lệnh trên sẽ đưa bạn vào chế độ nhập liệu. Trừ trường hợp duy nhất khi bạn sử dụng <r> để thay thế một ký tự, còn lại thì bạn phải bấm <Esc> để kết thúc việc thay đổi, sau đó trở về chế độ lệnh.

Lưu ý: để nhanh chóng thay nhiều từ cùng lúc, bạn đặt một số nguyên (tương ứng với số từ muốn thay) trước hai phím <c><w>.

Sau đây là hai thí dụ để bạn thực hành các lệnh thay thế, cũng sử dụng cùng một kịch bản **trước** và **sau** khi thay đổi một câu.

a) Sử dụng <c><e> :

Nguyên thuỷ:

```
The report demonstraits thw,strengths of are approach.
```

Con chạy đang nằm dưới chữ đầu tiên viết sai chính tả, nơi mà bạn sẽ bắt đầu sửa. Bây giờ bạn bấm <c><e>, gõ tes và bấm <Esc>.

Trở thành:

```
The report demonstrates_thw, strengths of are approach.
```

b) Sử dụng <Shift-r> :

Nguyên thuỷ:

```
The report demonstrates thw, strengths of are approach.
```

Con chạy đang ở vị trí chữ viết sai thứ hai, nơi mà bạn muốn thay ký tự. Để đổi "thw" thành "the" và thêm một khoảng trống, bạn bấm <Schift-r>, gõ e, bấm phím trống và bấm <Esc>.

Trở thành:

```
The report demonstrates the_strengths of are approach.
```

c) Sử dụng <c><w> :

Nguyên thuỷ:

```
The report demonstrates the strengths of are approach.
```

Con chạy đang ở dưới ký tự của từ mà bạn sắp thay đổi. Bạn bấm <2><c><w>, gõ our approach và bấm <Esc>.

Trở thành:

```
The report demonstrates the strengths of our approach.
```

Hãy nhớ bấm <Esc> sau mỗi lần thay đổi để trở về chế độ lệnh.

8.2.12 Chép, cắt và dán

Mỗi khi bạn cắt xoá các từ, ký tự, hoặc dòng, thì đối tượng bị cắt xoá ấy sẽ lưu trong vùng được gọi là "vùng đệm tổng quát". Bạn có thể dán nội dung vùng này vào bất cứ chỗ nào của văn bản bạn đang chỉnh sửa bằng lệnh <p> hoặc <Shift-p>. Lệnh <p> dán ngay bên phải, tức sau vị trí con chạy. Lệnh <Shift-p> dán ngay bên trái, tức phía trước con chạy. Sau đây là hai thí dụ:

a) Sử dụng <p> để dán phía sau con chạy:

Nguyên thuỷ:

```
Carefully carry these out instructions.
```

Bạn cắt từ "out" và một khoảng trống bằng cách bấm <d><w>. Tiếp theo bạn chuyển con chạy đến khoảng trống phía sau từ "carry" và bấm <p> để dán "out".

Trở thành:

```
Carefully carry out_these instructions.
```

b) Sử dụng <Shift-p> để dán phía trước con chạy:

Nguyên thuỷ:

```
Carefully carry these out instructions.
```

Hãy cắt các ký tự "these" và một khoảng trống bằng cách bấm <d><w>. Tiếp theo bạn chuyển con chạy đến ký tự i đầu tiên trong "instructions" và bấm <Shift-p>.

Trở thành:

```
Carefully carry out these instructions.
```

Lưu ý: Muốn thay đổi vị trí hai ký tự liền kề, bạn đặt con chạy dưới ký tự thứ nhất và bấm <x><p>. Bạn thử đổi "toa" thành "tao" thử xem.

Những thí dụ trên giới thiệu cách dán ký tự sau khi đã cắt xong. Tuy nhiên bạn không nhất thiết phải cắt trước khi dán. Bạn có thể thực hành động tác gọi là "**y**ank" (kéo), tương đương động tác copy (chép) ở các trình xử lý văn bản khác. Hình thức ra lệnh kéo tựa như ra lệnh cắt. Sau đây là vài lệnh kéo:

Lệnh	Hành động
<y><w>	Kéo từ vị trí con chạy trong từ hiện hành đến khởi điểm của từ tiếp theo.
<y><$>	Kéo từ vị trí con chạy đến cuối dòng.

<Shift-y>	Cũng như <y><$>: kéo phần còn lại của dòng hiện hành.
<y><y>	Kéo hết dòng hiện hành.

Tất cả những lệnh trên có thể áp dụng cho một số ký tự, dòng hoặc từ, bằng cách gõ vào một số nguyên trước câu lệnh.

Muốn chép một đoạn văn bản gồm bốn dòng sang chỗ khác, bạn làm như sau:

1. Đặt con chạy tại khởi điểm của đoạn văn bản ấy.

2. Bấm <4><y><y> để **vi** thực hiện 4 lần động tác kéo từ con chạy cho đến cuối dòng.

3. Đặt con chạy ở một nơi khác trong văn bản.

4. Bấm <p> để dán bốn dòng ấy vào phía dưới dòng có con chạy.

Bạn cũng có thể tìm kiếm và thay thế từ suốt tệp hoặc trong một phạm vi nào đó. Dạng thức câu lệnh như sau:

```
:[phạm_vi] s/chuỗi_cũ/ chuỗi_mới/g
```

phạm_vi: Chỉ định phạm vi hoạt động; thí dụ bạn có thể dùng ký hiệu phần trăm (%) để hoạt động trọn hết một tệp, hoặc xác định một số dòng nào đó (chẳng hạn như 1,4).

s: Thông báo đây là động tác tìm kiếm và thay thế.

chuỗi_cũ: Chuỗi cần tìm trong tệp và thay thế bằng chuỗi_mới.

chuỗi_mới: Chuỗi mà máy phải chèn vào; chuỗi_mới thay thế chuỗi_cũ.

Thí dụ để thay thế từ viết sai chính tả là "receved" bằng chữ viết đúng chính tả và áp dụng xuyên suốt toàn nhóm tệp, bạn có thể ra lệnh như sau:

```
:%s/receved/ received/g
```

8.2.13 Lặp lại câu lệnh

vi không chỉ lưu lại những từ vừa được xoá hoặc vừa được chép để dùng lại sau này, vì còn lưu lệnh cuối cùng bạn vừa nhập để sử dụng lần nữa. Bạn bấm <.> để thực hiện động tác lập lại câu lệnh vừa ra.

Giả sử bạn vừa soạn xong bản báo cáo và sực nhớ ra rằng phải thêm hai dòng như sau vào những vị trí chính yếu của văn bản:

```
********** Please comment ********
********** On this section ********
```

Bạn làm như sau:

1. Đặt con chạy vào vị trí mà bạn muốn chèn hai dòng đó lần thứ nhất.

2. Chèn hai dòng ấy vào bằng cách bấm <o> để mở một dòng, sau đó gõ đầy đủ các ký tự sao và chữ.

3. Bấm <Esc> trở về chế độ lệnh.

4. Muốn chèn vào nơi khác của văn bản, bạn đặt con chạy vào chỗ đã chọn và bấm <.> để chèn hai dòng ấy vào lần nữa.

8.3 Tóm tắt các lệnh vi

Sau khi đã tìm hiểu một số lệnh cơ bản của trình soạn thảo văn bản **vi**, hãy xem bảng sau để biết tất cả các lệnh mà bạn có thể dùng với nó.

Lệnh	Hành động
<i>	Chèn ký tự trước con chạy
<shift-i>	Nhập ký tự tại đầu dòng
<a>	Thêm ký tự sau con chạy
<shift-a>	Nhập ký tự tại cuối dòng
<o>	Mở một dòng dưới con chạy
<shift-o>	Mở một dòng trên con chạy
<d><w>	Xoá cả từ
<d><d>	Xoá cả dòng
<shift-d>	Xoá đến cuối dòng
<x>	Xoá ký tự phía dưới con chạy
<c><w>	Thay cả từ
<c><c>	Thay cả dòng
<shift-c>	Thay cho đến cuối dòng
<shift-r>	Thay từ phía dưới con chạy
<shift-j>	Hợp các dòng lại
<e>	Chuyển con chạy đến cuối từ
<w>	Chuyển con chạy đến từ tiếp theo
<$>	Chuyển con chạy đến cuối dòng
<l>	Chuyển con chạy về phía phải một khoảng trống
<k>	Chuyển con chạy lên trên một dòng
<j>	Chuyển con chạy xuống dưới một dòng
<h>	Chuyển con chạy về phía trái một khoảng trống
<f><x>	Chuyển con chạy đến chỗ ký tự x xuất hiện lần đầu tiên
<shift-f><x>	Chuyển con chạy đến chỗ ký tự x xuất hiện lần cuối cùng
<;>	Lặp lại lệnh <f> hoặc <shift-f>
<số><\>	Chuyển con chạy đến số cột chỉ định
<shift-h>	Chuyển con chạy đến dòng đầu tiên của màn hình
<shift-l>	Chuyển con chạy đến dòng cuối cùng của màn hình

<shift-m>	Chuyển con chạy đến dòng giữa của màn hình
<shift-g>	Chuyển con chạy đến dòng cuối của tệp (vùng đệm)
<số><shift-g>	Chuyển con chạy đến dòng <số> (giống <ESC>: <số>)
<^>	Chuyển con chạy đến dầu dòng
<m><x>	Đánh dấu vị trí hiện hành bằng ký tự x
<Ctrl-d>	Cuộn tới nửa màn hình
<Ctrl-u>	Cuộn lui nửa màn hình
<Ctrl-f>	Cuộn tới một màn hình
<Ctrl-b>	Cuộn lui một màn hình
<Ctrl-l>	Vẽ lại màn hình
<Ctrl-shift-g>	Hiển thị tên tệp, dòng lệnh và số cột
<z><z>	Vẽ lại màn hình với dòng hiện hành ở giữa màn hình
<y><y>	Kéo nguyên dòng vào vùng đệm
<p>	Dán nội dung vùng đệm vào phía dưới con chạy
<shift-p>	Dán nội dung vùng đệm vào phía trên con chạy
x"[số]"<y><y>	Kéo một số dòng vào vùng đệm mang tên x
x<p>	Dán nội dung vùng đệm x phía sau con chạy
:w[tệp]	Ghi nội dung vào đĩa như là tệp
:q	Thoát khỏi **vi**
:wq	Lưu các thay đổi và thoát khỏi **vi**
:r tệp	Chép tệp đã chỉ định vào trình soạn thảo
:e tệp	Chỉnh sửa tệp
:! lệnh	Thi hành lệnh shell được chỉ định
: <số>	Chuyển con chạy đến số dòng được chỉ định
:f	In dòng hiện hành và tên tệp (giống như <Ctrl-shift-g>
/chuỗi	Tìm về phía trước đến khi gặp chuỗi ký tự
?chuỗi	Tìm về phía sau đến khi gặp chuỗi ký tự
:x,ys/chuỗi_cũ/chuỗi_mới	Thay chuỗi cũ bằng chuỗi mới từ dòng x đến dòng y
<ESC><u>	Hoàn nguyên lệnh sau cùng
<n>	Tìm lần hiện diện tiếp theo của chuỗi
.	Lặp lại lệnh sau cùng
~	Đổi ký tự từ chữ in sang chữ thường và ngược lại

<ESC>	Chuyển sang chế độ lệnh

*Bảng 8.7: Tóm tắt các lệnh **vi***

8.4 Thiết lập môi trường vi

Trình soạn thảo **vi** có nhiều tuỳ chọn mà bạn có thể dùng hay không dùng. Một vài tuỳ chọn như thế được quản trị viên thiết lập cho toàn thể mọi user dùng chung. Vài tuỳ chọn khác dành để bạn sắp xếp môi trường riêng cho mình. Chúng sẽ có tác dụng mỗi khi bạn kích hoạt **vi.**

Bảng 8.8 tóm tắt tất cả các tuỳ chọn môi trường để bạn dùng với **vi**. Khi thiết lập chúng (như mô tả ở mục tiếp theo), bạn có thể sử dụng những chữ viết tắt ở cột đầu tiên của bảng 8.8 hoặc nguyên tên tiếng Anh ở cột thứ hai.

*Bảng 8.8: Các tuỳ chọn môi trường của **vi***

Viết tắt	Nguyên tên tiếng Anh và chức năng của tuỳ chọn
ai	Autoindent, căn lề mỗi dòng cho dóng thẳng cột cùng với dòng phía trên (rất tiện khi soạn thảo chương trình). Mặc định là off
ap	Autoprint, hiển thị dòng hiện hành khi dòng thay đổi. Mặc định: on
eb	Errorbells, làm cho máy kêu bip bip khi bạn ra lệnh sai. Mặc định: off
nu	Number, hiển thị số hiệu của dòng khi bạn chỉnh sửa tệp. Mặc định: off
redraw	Redraw, luôn cập nhật màn hình mỗi khi có thay đổi. Mặc định là on
report	Report, theo dõi sự thay đổi trong tiến trình hiệu chỉnh và đưa kết quả ra dòng trạng thái khi vượt phạm vi cho trước. Thí dụ report=3 sẽ thông báo mỗi khi bạn ra lệnh xoá ba dòng, song sẽ im lặng nếu bạn xoá ít hơn ba dòng. Mặc định là report=5
sm	Showmatch, hiển thị một dấu mở ngoặc đơn mỗi khi bạn gõ vào một dấu đóng ngoặc đơn, rất tiện khi soạn chương trình. Mặc định là off
smd	Showmode, hiển thị INPUT, REPLACE, hoặc CHANGE ở phía bên phải của dòng trạng thái khi bạn gõ vào lệnh tương ứng. Mặc định là off
warn	Warn, hiển thị cảnh báo khi bạn muốn thoát khỏi **vi** khi vùng đệm bị thay đổi mà không được lưu vào tệp. Mặc định là on
wm=n	Wrapmargin, xác định lề phải, với n là một số nguyên. Nếu n lớn hơn 0, lệnh sẽ tự động nhảy dòng, sao cho không từ nào cách lề phải bằng n ký tự hoặc ít hơn. Thí dụ wm=5 sẽ bọc tròn chữ (và xuống hàng) khi một ký tự hiện diện trong phạm vi năm ký tự của cuối dòng. Muốn tắt tuỳ chọn này, bạn gõ wm=0 và đây là mặc định
ws	word search (một số hệ thống khác còn gọi là wrapscan), bọc tròn kể từ ký tự <eof> (ký tự cuối tệp) cho đến ký tự <bof> (ký tự đầu tệp) trong tiến trình tìm kiếm. Mặc định là on

8.4.1 Sử dụng lệnh set để xem và chọn

Muốn xem lại các tuỳ chọn đã lập cho hệ thống máy bạn, hãy gõ **:set** khi ở chế độ lệnh. Các tuỳ chọn đã lập cho phiên làm việc hiện hành sẽ hiện trên dòng trạng thái. Chúng sẽ thay đổi theo thiết lập của bạn. Sau đây là một thí dụ:

```
set:
autoprint errorbells redraw report=1 showmatch
showmode
term=vt100 wrap margin=5
```

Ghi chú: Nếu bạn ra lệnh set và không kèm đối số, máy chỉ hiển thị các tuỳ chọn do user thiết lập. Bạn có thể viết tắt lệnh set thành se. Muốn đặt một số tuỳ chọn trên cùng một dòng, bạn dùng lệnh se và phân cách các tuỳ chọn bằng khoảng trống, như ở thí dụ sau:

```
:se ap eb redraw report=1 sm smd warn wn=5 ws
```

Lưu ý ký tự đầu tiên là dấu hai chấm, báo cho **vi** biết là sắp có lệnh được gõ vào. Muốn xem danh sách các tuỳ chọn và cách thiết lập, bạn gõ **:set all** (bảng 8.7).

8.4.2 Thiết lập tuỳ chọn showmode

Một trong những tuỳ chọn thường được dùng nhất là **showmode**. Bạn thực tập sử dụng **showmode** bằng cách khởi động **vi** lần nữa với tệp vi**_test.**1.

Trước đây, có lẽ bạn đã để ý rằng không thể nào biết mình đang ở chế độ nào khi bạn gõ văn bản vào tệp vi**_test**.1. Giờ đây bạn có thể dùng tuỳ chọn **showmode** để hiển thị chế độ hiện hành ở dòng trạng thái. Khi lập tuỳ chọn **showmode, vi** sẽ hiển thị chế độ nhập liệu hiện hành ở dạng INPUT MODE bình thường, hoặc APPEND MORE, hoặc REPLACE 1 CHAR... v.v... Muốn lập tuỳ chọn ở **vi**, bạn bấm <Esc> để trở về chế độ lệnh và gõ **:set showmode**. Đến đây bạn gõ <i> để sang chế độ nhập và thấy INPUT MODE ở dòng trạng thái. Bạn bấm <Esc> để trở về chế độ lệnh.

8.4.3 Lập tuỳ chọn bật/tắt

Tuỳ chọn nào không cần đối số ở dạng số thì nó giống như một công-tắc chỉ có 2 trạng thái bật/tắt. Thí dụ ở mục trước bạn biết cách lập **showmode** bằng lệnh **:se showmode.** Muốn tắt tuỳ chọn này, bạn chỉ cần viết thêm **no** vào phía trước **:se noshowmode.**

8.4.4 Thay đổi tuỳ chọn theo từng phiên

Những tuỳ chọn bạn lập cho phiên làm việc **vi** chỉ hữu hiệu cho phiên làm việc ấy mà thôi. Muốn những tuỳ chọn hữu hiệu thường trực cho tất cả các phiên làm việc **vi**, bạn phải ghi các lệnh set vào tệp mang tên .exrc trong home directory của chính bạn. Bạn gõ lệnh sau đây xem thử tệp ấy có hiện diện hay không:

```
cd /
vi .exrc
```

Lệnh thứ nhất đưa bạn về home directory. Lệnh thứ hai khởi động **vi** bằng cách sử dụng tệp .exrc. Nếu đã có sẵn, tệp sẽ hiện trên màn hình **vi**. Nếu chưa có, **vi** sẽ báo bạn biết đó là tệp mới.

Các lệnh set trong tệp .exrc khởi động với từ set nhưng không có dấu hai chấm. Thí dụ sau đây lập tuỳ chọn **number** và **showmode**:

```
set number showmode
```

Ghi chú: Tệp .exrc chỉ bắt đầu có hiệu lực khi bạn khởi động **vi**. Nếu bạn tạo ra tệp này trong khi đang chạy **vi** thì bạn phải khởi động lại **vi** để các tuỳ chọn mới sẽ có hiệu lực.

Chương 9. Khởi động và đóng tắt

Hai thao tác bắt buộc và thông thường nhất đối với một quản trị viên là khởi động và đóng tắt hệ thống. Tuy nhiên muốn khởi động Linux thì những thao tác đó cần được tiến hành đúng hướng dẫn.

Các chủ đề chính sẽ được đề cập đến trong chương này bao gồm:

- *Trình quản lý mồi LILO*
- *Trình quản lý mồi GRUB*
- *Tiến trình khởi động*
- *Mồi Linux bằng đĩa mềm*
- *Khởi động bằng trình mồi*
- *Đóng tắt Linux.*

9.1 Giới thiệu chung

Muốn sử dụng máy tính, đầu tiên bạn phải mồi (boot) cho hệ điều hành của nó khởi động. Máy tính của bạn có thể được cài sẵn vài hệ điều hành, vì vậy bạn phải biết cách khởi động hệ nào khi bật công tắc điện. Có hai cách cơ bản là khởi động hệ điều hành bằng đĩa mềm hoặc khởi động từ ổ đĩa cứng nhờ một chương trình quản lý mồi.

Khi bật điện một máy PC, CPU sẽ tự đến một địa chỉ trong vùng nhớ hệ thống của BIOS và thực hiện các chỉ thị nằm ở đó.

BIOS là một bộ nhớ ROM nằm trên bìa mẹ của PC, ngoài chức năng cung cấp các chỉ thị vào ra cơ bản cho các thiết bị ngoại vi nó còn quản lý các bước đầu tiên của tiến trình khởi động.

BIOS giúp CPU kiểm tra bìa mẹ và các thiết bị ngoại vi nối vào đó xong rồi tìm kiếm ổ đĩa nào có chứa chương trình mồi. Thông thường, CPU sẽ tìm kiếm theo thứ tự từ ổ đĩa mềm đến ổ CD rồi đến ổ đĩa cứng. Thứ tự này có thể thay đổi bằng cách chọn một cấu hình khác có sẵn trong BIOS.

Khi kiểm tra ổ cứng, CPU sẽ tìm đến MBR (tức rãnh đầu tiên của ổ cứng đầu tiên) và nạp từ đó vào bộ nhớ một chương trình quản lý mồi.

MBR chứa các chỉ dẫn cho biết cách nạp một trình quản lý mồi (trình GRUB hoặc LILO của Linux, hay NTLDR của Windows NT/2000). BIOS sau khi nạp trình quản lý mồi sẽ chuyển quyền điều khiển cho trình này.

Trình quản lý mồi sẽ hiển thị trên màn hình một danh sách các tuỳ chọn để người dùng xem nên khởi động hệ điều hành nào.

Các chỉ dẫn cho việc nạp hệ điều hành thích hợp được ghi rõ trong các tệp cấu hình tương ứng với các trình (quản lý) mồi như sau:

- LILO lưu cấu hình trong tệp /etc/**lilo**.conf

- GRUB lưu cấu hình trong tệp /boot/grub/grub.conf

- NTLDR lưu cấu hình trong tệp c:\boot.ini

Trong giáo trình này, chúng ta chỉ tìm hiểu các trình mồi LILO và GRUB, bởi vì NTLDR thuộc về Windows NT và Windows 2000.

9.2 Trình quản lý mồi LILO

LILO là một trình mồi nằm trong các bản phát hành Red Hat Linux và là trình mồi mặc định của các phiên bản Red Hat trước 7.1, từ phiên bản 7.2, ta có thêm trình mồi GRUB. Nếu muốn sử dụng GRUB thay vì LILO, bạn có thể bỏ qua mục này và chuyển sang mục sau (trình quản lý mồi GRUB).

Bạn tuỳ ý cài đặt LILO vào đĩa cứng hoặc đĩa mềm. Cách dễ nhất để cài đặt LILO là dùng chương trình cài đặt của Red Hat hoặc Caldera, vì hai chương trình này giúp bạn tự động thực hiện một phần lớn công việc theo **menu** (thực đơn).

Lưu ý: Bạn nên cài đặt LILO bằng chương trình cài đặt của Red Hat hoặc Caldera. Cài đặt trình mồi là một tiến trình nhiều rủi ro, dễ gây ra hỏng hóc dữ liệu trên ổ cứng nếu làm không đúng cách (xem "Cài đặt LILO").

Khi cài đặt LILO xong, bạn có thể chọn ngay hệ điều hành mong muốn ở thời điểm mồi. Nếu không, LILO sẽ đếm giá trị timeout (thời gian chờ đợi), sau đó sẽ tự động mồi cho hệ điều hành mặc định khởi động.

9.2.1 Thiết lập cấu hình LILO

LILO đọc thông tin chứa ở tệp cấu hình /etc/lilo.conf để biết xem trong máy bạn có những hệ điều hành nào và các thông tin khởi động đang nằm ở đâu. LILO được lập cấu hình để mồi các đoạn thông tin của tệp trên cho từng hệ điều hành.

Sau đây là thí dụ về ba đoạn của một tệp /etc/lilo.conf. Đoạn 1:

```
boot = /dev/hda
map = /boot/map
install = /boot/boot.b
prompt
timeout = 50
message = /boot/message
lba32
default = linux
Đoạn 2:
image = /boot/vmlinuz-2.4.0-0.43.6
label=linux
initrd = /boot/inditrd-2.4.0-0.43.6.img
read-only
root = /dev/hda5
Đoạn 3:
```

```
other = /dev/hda1
label = dos
```

Đoạn thứ nhất cho LILO biết:

- Trình mồi nằm ở ổ đĩa cứng /dev/hda/

- Kiểm tra tệp map trong thư mục /boot/map

- Nạp các thông báo trong tiến trình mồi từ tệp /boot/message

- Có thể cài đặt một tệp đặc biệt (/boot/boot.b) như là một rãnh mồi mới

- Ổ đĩa cứng đang hỗ trợ LBA32 (dòng này thường có giá trị là linear, không nên đổi lại dòng này nếu bạn chưa biết rõ đĩa cứng của mình; bạn có thể tìm hiểu đĩa cứng có hỗ trợ LBA32 hay không bằng cách kiểm tra thông tin BIOS)

- Hệ điều hành mặc định là Linux

- Thời gian chờ trước khi nạp hệ mặc định là 5 giây (đơn vị tham số được tính bằng 1/10 của giây)

Đoạn thứ hai cho LILO biết:

- lõi Linux đang ở tệp /boot/vmlinuz-2.4.0-0.43.6

- Linux là tên hệ điều hành sẽ xuất hiện tại **menu** khởi động của LILO

- Vị trí root của hệ thống tệp Linux đang ở ổ đĩa cứng /dev/hda5

Đoạn thứ ba cho LILO biết:

- Có phân vùng của một hệ điều hành nữa đang ở ổ đĩa cứng /dev/hda1

- DOS là tên hệ điều hành thứ hai (còn dùng để chạy Windows)

9.2.2 Sử dụng LILO

Khi cài đặt LILO, bạn sẽ đặt giá trị timeout và hệ điều hành mặc định. Giá trị này cho phép bạn suy nghĩ trong một thời gian ngắn trước khi hệ mặc định tự khởi động, xem có nên đổi sang một hệ điều hành khác hay không. Sau thời gian timeout nếu bạn không can thiệp gì cả, LILO sẽ khởi động hệ điều hành mặc định.

Từ RedHat 7.x trở đi, ta có một **menu** với giao diện đồ hoạ để tiện cho việc chọn hệ điều hành.

Khi bật nút điện một máy tính có LILO ở đĩa cứng hay ở đĩa mềm, màn hình LILO: sẽ xuất hiện. Đến đây bạn có nhiều tuỳ chọn. Hoặc bạn chờ LILO khởi động hệ điều hành mặc định, hoặc bạn bấm <Ctrl>, <Alt> hay <Shift> cho LILO khởi động tức khắc. Bạn có thể nhấn <Enter> khi thấy tên hệ điều hành mà bạn muốn LILO khởi động ngay đang được chọn.

Sau đó việc khởi động bắt đầu được chuyển từ LILO về cho hệ điều hành quản lý. Nếu máy bạn không cài đặt GRUB, bạn có thể xem tiếp mục 9.4 để hiểu hơn về tiến trình khởi động ở mức hệ điều hành.

9.3 Trình quản lý mồi GRUB

Khi máy bạn sử dụng CPU họ x86 của Intel hoặc AMD, chỉ để chạy Linux và chỉ có một phiên bản duy nhất của Linux kernel, bạn sẽ không cần chú ý gì lắm về GRUB. Tuy nhiên, nếu bạn muốn có thể tuỳ chọn một trong vài hệ điều hành đã được cài đặt trên máy thì bạn nên xem qua mục này để hiểu thêm về trình quản lý mồi GRUB của Linux.

Minh hoạ 9.1: Màn hình khởi động GRUB

9.3.1 Định nghĩa

GRUB (Grand Unified Bootloader) cũng chỉ là một trình quản lý mồi tương tự LILO hay các trình mồi khác như Partition Magic, DriverStart... được nạp vào rãnh MBR để khởi động hệ điều hành mà bạn chọn. Nó cho phép đặt một chỉ dẫn đặc biệt trong MBR để máy bạn sẽ theo đó mà hiện ra chính xác các tuỳ chọn về hệ điều hành khi máy bắt đầu khởi động.

9.3.2 Tiến trình khởi động ở máy x86

Như phần trên đã trình bày (mục 9.1), BIOS sẽ giúp CPU kiểm tra các thiết bị nối vào máy tính, bìa Video, RAM, nhận diện các ổ đĩa cứng, đĩa mềm và các bìa mạch khác v.v., trước khi chuyển quyền điều khiển sang cho trình mồi có trên đĩa mềm hay trên rãnh MBR của đĩa cứng đầu tiên. Tiến trình nạp trình mồi (GRUB, LILO,...) và nạp hệ điều hành được chia thành những giai đoạn sau:

- Giai đoạn 1- Nạp trình mồi sơ cấp (primary boot loader): trình này chiếm một khoảng không gian rất nhỏ (cỡ dưới 512 bytes) trên MBR. Nhiệm vụ của nó là nạp một trình mồi thứ cấp nằm trên vùng khác ở đĩa cứng hay đĩa mềm và có kích thước lớn hơn nhiều.

- Giai đoạn 2- Nạp trình mồi thứ cấp (secondary boot loader): Đây mới thật sự là trình quản lý việc mồi hệ điều hành mà bạn chọn. Đối với GRUB, đó là đoạn mã chương trình cho phép hiện lên một **menu** hay một dấu nhắc để bạn lựa chọn.

- Giai đoạn 3- Nạp hệ điều hành vừa chọn: Sau khi GRUB nhận được chỉ dẫn chính xác để khởi động một hệ điều hành từ một phân vùng xác định, nó sẽ chuyển quyền điều khiển ngay cho hệ điều hành đó, dù cho GRUB được nạp từ dòng lệnh hay từ tệp cấu hình.

Lưu ý: Có một vài hệ điều hành lại cần đến thêm một giai đoạn nữa. Đó là khi trình mồi sơ cấp cần chuyển quyền cho trình mồi thứ cấp đang nằm trên một phân vùng mà trình sơ cấp không thể truy cập trực tiếp.

Ghi chú: Cách nạp hệ điều hành như trên gọi là mồi trực tiếp. Còn cách của DOS và Windows 9x là mồi theo móc xích: MBR chỉ chứa thông tin về các phân vùng đang có trên đĩa và một vài chỉ thị ra lệnh chuyển về địa chỉ đầu của phân vùng đang chứa hệ điều hành. GRUB hỗ trợ cả 2 cách mồi đó.

9.3.3 Tệp cấu hình GRUB

Tệp cấu hình GRUB (/boot/grub/grub.conf) cũng khá giống như tệp cấu hình LILO ở trên.

Dưới đây là một thí dụ về tệp cấu hình GRUB cho Linux và cho Windows 2000:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You do not have a /boot partition. This means that
# all kernel and initrd paths are relative to /.eg.
# root (hd0.0)
# kernel /boot/vmlinuz-version ro root=/dev/hdc1
# initrd /boot/initrd-version.img
# boot=/dev/hdc
default=0
timeout=10
splashimage=(hd0.0)/boot/grub/splash.xmp.gz
# section to load linux
title Red Hat Linux (2.4.18-3)
root (hd0.0)
kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hdc5
initrd /boot/initrd-2.4.18-3.img
boot
# section to load Windows 2000
title windows
rootnoverify (hd1.0)
chainloader +1
```

Như thí dụ về tệp cấu hình LILO trước đây, ta thấy tệp cấu hình GRUB cũng có 3 đoạn (section) cơ bản.

Đoạn thứ nhất mô tả các chỉ thị tổng quát như:

- Hệ điều hành mặc định là linux

- Thời gian chờ đợi người dùng gõ phím trước khi thực hiện lệnh mặc định là 1 giây

Đoạn thứ 2 cho biết các thông số để khởi động hệ Linux:

- Tiêu đề trên **menu** là Red Hat Linux

- Hệ điều hành này sẽ khởi động từ phân vùng đầu tiên của ổ đĩa thứ nhất (hd0.0) và cần phải **mount** phân vùng này trước

- Tệp vmlinuz đang được chứa trong thư mục root và gốc của hệ thống tệp đang nằm ở /dev/hdc5

- Dòng lệnh boot nhắc phải nạp ngay tệp hệ điều hành đã được khai báo ở trên

Đoạn thứ 3 cho biết các thông số về hệ điều hành thứ hai đang được cài đặt trong hệ thống:

- Tiêu đề là Windows.

- Hệ điều hành đang chiếm phân vùng thứ nhất của ổ đĩa thứ hai (hd1.0). Có điều với lệnh rootnoverify, GRUB không cần chú ý kiểm tra xem phân vùng này có được **mount** hay không.

- Câu lệnh chainloader đã sử dụng +1 làm tên tệp cần khởi động như một móc xích trong tiến trình: +1 có nghĩa là sector thứ nhất của phân vùng đang xem xét.

Bạn cần chú ý rằng mặc dù /boot nằm ở ổ đĩa hda1, nhưng thư mục gốc (/) lại nằm ở ổ đĩa hdc5.

Ta còn có thể sử dụng các câu lệnh khác có trong bảng sau:

Lệnh	Ý nghĩa
color <màu_1> <màu_2>	Màu của **menu** sẽ có dạng như màu_1, còn khi được chọn sẽ có dạng như màu_2
password <password>	Ngăn các user không có mật khẩu sẽ không sửa được **menu**. Có thể thêm một **menu** thứ hai sau lệnh này. Khi đó, nếu nhập mật khẩu thành công, GRUB sẽ nạp lại giai đoạn 2 để hiện **menu** thứ hai
hiddenmenu	Giấu **menu** không hiển thị ra để chọn cho đến khi đủ thời gian mặc định hay user bấm phím <ESC>
splashimage	Chỉ vị trí của tệp ảnh sẽ sử dụng trong tiến trình khởi động
#	Mở đầu một ghi chú cho người (máy sẽ bỏ qua không đọc)
displaymem	Hiển thị số lượng bộ nhớ đang sử dụng (current use of memory)

9.4 Tiến trình khởi động

Red Hat và phần lớn các bản phát hành sau này của Linux sử dụng tiến trình khởi động mang tên SysV init thay vì kiểu BSD init cũ. SysV init là chương trình đầu tiên mà kernel thực thi ngay từ lúc khởi động máy, do đó init được mang số định danh tiến trình (PID) số 1, trở thành tiến trình "mẹ" của tất cả các tiến trình khác chạy dưới Linux.PID của một tiến trình là mã số mà hệ điều hành dùng để nhận diện tiến trình ấy. Nhiều lệnh của Linux dùng PID ấy làm tham số.

Khi khởi động, Linux đi theo những bước sau:

- Kernel chạy SysV init, chương trình này nằm trong thư mục /sbin.

- SysV init chạy shell script /etc/rc.d/rc.sysinit.

- rc.sysinit lập các biến hệ thống khác nhau và thực hiện các thao tác ban đầu.

- SysV init chạy shell script /etc/rc.d/rc.serial.

- rc.serial sử dụng một số lệnh setserial để thiết lập cấu hình của các cổng serial cho hệ thống. Bạn có thể xem trang **man** của lệnh setserial để biết cách sử dụng.

- SysV init chạy tất cả các script được quy ước là mặc định.

- SysV init chạy script /etc/rc.d/rc.local.

Chương trình này khởi động nhiều tiến trình khác, đồng thời chuyển thông báo cho thiết bị và cho tệp đăng nhập hệ thống /var/log/messages biết về trạng thái của từng tiến trình đã khởi động.

Lưu ý: Tệp đăng nhập hệ thống /var/log/messages là để giúp bạn giải quyết các vấn đề có thể gặp khi khởi động. Kernel lưu tất cả những thông báo lỗi trong tệp này, do đó bạn khỏi mất công ghi chép lại khi chúng xuất hiện (nhưng nhanh chóng bị xoá mất).

Sau đây là một trình tự khởi động điển hình:

```
April 1 23:23:42 ns syslogd 1.3-3: restart.

April 1 23:23:43 ns kernel: klogd 1.3-3, log source = /proc/kmsg started.

April 1 23:23:45 ns kernel: Loaded 4189 sysmbols from /boot/System.map.

April 1 23:23:45 ns kernel: Symbold match kernel version 2.0.31.

April 1 23:23:45 ns kernel: Loaded 2 symbols from 3 modules.

April 1 23:23:45 ns kernel: Console: 16 point font, 400 scans

April 1 23:23:45 ns kernel: Console: colour VGA+ 0x25, 1 virtual console
(max 630

April 1 23:23:45 ns kernel: pci_init: no BIOS32 detected

April 1 23:23:45 ns kernel: Calibrating delay loop.. ok - 49.97 BogoMIPS

April 1 23:23:45 ns kernel: Memory: 30816k/32768k available (736k kerne
code, 384k reserved, 382k data)

April 1 23:23:45 ns kernel: This processor honours the WP bit even when in
supervisor mode. Good.

April 1 23:23:45 ns kernel: Swansea University Computer Society NET3.035.

April 1 23:23:45 ns kernel: Swansea University computer society TCP/IP for
NET3.034.

April 1 23:23:45 ns kernel: IP Protocols: IGMP, ICMP, UDP, TCP

April 1 23:23:45 ns kernel: VFS: Diskquotas version dquot_5.6.0 initialized

April 1 23:23:45 ns kernel:

April 1 23:23:45 ns kernel: Checking 386/387 coupling... ok,

fpu using exception 16 error reporting.

April 1 23:23:45 ns kernel: Checking 'hlt' instruction... ok.

April 1 23:23:45 ns kernel: Linux version 2.0.31

(root@porky.redhat.com) (gcc version 2.7.2.3) #1 Sun Now o 21:45:23 EST
1997

April 1 23:23:45 ns kernel: Starting kswapd v 1.4.2.2

April 1 23:23:45 ns kernel: Serial driver version 4.13 with no serial
options enabled

April 1 23:23:45 ns kernel: tty00 at 0x03f8 (irq = 4) is a 16550A

April 1 23:23:45 ns kernel: tty01 at 0x02f (irq = 3) is a 16550A

April 1 23:23:45 ns kernel: Real Time Clock Driver v1.07
```

```
April 1 23:23:45 ns kernel: Ramdisk driver initialized : 16 ramdisks of
4096K size

April 1 23:23:45 ns kernel: hda: Micropolis 2217A, 1551MB w/508kB Cache,
CHS=3152/16/63

April 1 23:23:45 ns kernel: hdb: Maxtor 72700 AP, 2583MB w/128kB Cache,
CHS=20746/15/17

April 1 23:23:45 ns kernel: ide0 at 0x1f0-0x1f7, 0x3f6 on irq 14

April 1 23:23:45 ns kernel: Floppy drive(s): fd0 is 1.44M

April 1 23:23:45 ns kernel: FDC 0 is an 8272A

April 1 23:23:45 ns kernel: md driver 0.35 MAX_MD_DEV=4, MAX_REAL=8

April 1 23:23:45 ns kernel: scsi : 0 hosts.

April 1 23:23:45 ns kernel: scsi :mdetected total.

April 1 23:23:45 ns kernel: Phân vùng check:

April 1 23:23:45 ns kernel: hda: hda1

April 1 23:23:45 ns kernel: hdb: hdb1 hdb2

April 1 23:23:45 ns kernel: VFS: Mounted root (ext2 filesystem) readonly.

April 1 23:23:45 ns kernel: Adding Swap: 3300k swap-space (priority -1)

April 1 23:23:45 ns kernel: sysct1: ip forwarding off

April 1 23:23:45 ns kernel: Swansea Univerity Computer Society IPX 0.34 for
NET 3.035

April 1 23:23:45 ns kernel: IPX Portions Copyright 1995 Caldera, Inc.

April 1 23:23:45 ns kernel: Appletalk 0.17 for Linux NET 3.035

April 1 23:23:45 ns kernel: eth0: 3c509 at 0x300 tag 1, 10baseT port,
address 00 60 97 13 30 e1, IRQ 10.

April 1 23:23:45 ns kernel: 3c509.c:1.12 6/4/97 becker@cesdis.gsfc.nasa.gov

April 1 23:23:45 ns kernel: eth0: Setting Rx mode to 1 addesses.

April 1 23:23:45 ns named[2431]: starting. named 4.9.6-REL Thu Nov 6
23:29:57 EST 1997

^Iroot@porky.redhat.com:/usr/src/bs/BUILD/ bind-4.9.6/named
```

Xem: "Tìm hiểu các tiến trình"

SysV init kích hoạt mọi tiến trình do hệ điều hành yêu cầu, chẳng hạn như cho phép việc thao tác trên mạng, sử dụng con chuột, cùng một số thao tác cơ bản khác như vào ra terminal. SysV init biết phải kích hoạt những tiến trình nào bằng cách đọc các tệp cấu hình trong thư mục /etc/rc.d. Sau này những tệp ấy sẽ được xử lý riêng lẻ tuỳ theo cấp chạy chương trình (run level) đã xác định. Cấp chạy chương trình quy định việc cung cấp các loại dịch vụ, từ chế độ chạy single-user (một người sử dụng, cấp 1) đến multiuser, multitasking và chế độ chạy tất cả các tiến trình (cấp 3). Bảng sau giới thiệu các cấp chạy của Linux. Giá trị mặc định của cấp chạy được lưu trong tệp /etc/inittab.

Bảng 9.1: Các cấp chạy của Linux

Cấp	Mô tả
0	Ngừng chạy

1	Single-user, không chạy các dịch vụ mạng NFS
2	Multiuser, không chạy các dịch vụ mạng NFS
3	Multiuser, hỗ trợ hoàn toàn các dịch vụ mạng
4	Không sử dụng (đang để dành)
5	Multiuser với các dịch vụ mạng và đăng nhập với giao diện đồ hoạ
6	Khởi động lại

Chương trình SysV init sử dụng cấu trúc thư mục như sau:

```
init.d
rc0.d
rc1.d
rc2.d
rc3.d
rc4.d
rc5.d
rc6.d
```

Những con số n trong tên thư mục rcn.d tương ứng với cấp chạy n trong bảng 9.1. Mỗi thư mục chứa nhiều shell script có khả năng kích hoạt hoặc dừng hẳn các dịch vụ cần thiết cho mỗi cấp chạy. Các script này cũng kích hoạt hệ thống tệp và khoá tệp vào trạng thái theo ý user. Thật ra các tệp trong các thư mục rcn.d chỉ là các liên kết đến các tệp script trong thư mục init.d. Do đó khi bạn muốn sửa chữa gì trong các tệp script khởi động, bạn không cần phải vào từng thư mục của chế độ khởi động mà chỉ cần vào thư mục init.d để sửa. (xem thêm ở dưới).

Xem "Làm việc với các shell script"

Mỗi thư mục chứa nhiều shell script, tên mỗi tệp script bắt đầu bằng ký tự S hoặc K (Start hoặc Kill: Bắt đầu hoặc Huỷ) theo sau là một con số có hai chữ số. Con số này chỉ để xếp thứ tự trình tự chứ không có ý nghĩa nào khác.

Mặc dù có thể chấp nhận nhiều tham số, song mỗi script thường chấp nhận một đối số dòng lệnh, đối số này dùng để kích hoạt hoặc dừng hẳn, init cung cấp một trong hai thứ, hoặc bắt đầu hoặc dừng hẳn tuỳ thuộc vào việc liệu rc có được gọi vào để thay đổi cấp chạy hay không.

Bạn có thể thực hành các script theo cách thủ công nếu cần phải cấu hình lại một dịch vụ nào đó. Chẳng hạn bạn có thể dùng sendmail với lệnh sau (bạn phải đăng nhập với tư cách root mới được phép ra lệnh thực thi các script init):

```
/etc/rc.d/init.d/sendmail stop
/etc/rc.d/init.d/sendmail stop
/etc/rc.d/init.d/sendmail start
```

Bạn lưu ý hai điều sau đây. Thứ nhất, câu lệnh được lặp lại hai lần với tham số stop là để đảm bảo hệ thống có đủ thời gian ngừng tiến trình trước khi lệnh start được gọi. Thứ hai, bạn nhận thấy script được thực thi từ thư mục init.d, mà không từ thư mục

của cấp chạy. Ngoài ra script này cũng không có chữ nào hoặc số nào. Khi liệt kê các tệp trong bất kỳ thư mục cấp chạy nào, bạn sẽ thấy rằng chúng liên kết với các tệp trong thư mục init.d.

Sau đây là một thư mục rc.3.d điển hình:

```
Irwxrwxrwx 1 root root 16 Jan 25 21:56 K08autofs->../init.d/autofs
Irwxrwxrwx 1 root root 18 Dec 14 12:17 K10pnserver->../init.d/pnserver
Irwxrwxrwx 1 root root 17 Dec 14 12:17 K20rusersd->../init.d/rusersd
Irwxrwxrwx 1 root root 15 Dec 14 12:17 K20rwhod->../init.d/rwhod
Irwxrwxrwx 1 root root 15 Dec 14 12:17 S15nfsfs->../init.d/nfsfs
Irwxrwxrwx 1 root root 16 Dec 14 12:17 S20random->../init.d/random
Irwxrwxrwx 1 root root 16 Dec 14 12:17 S30syslog->../init.d/syslog
Irwxrwxrwx 1 root root 13 Dec 14 12:17 S40atd->../init.d/atd
Irwxrwxrwx 1 root root 15 Dec 14 12:17 S40crond->../init.d/crond
Irwxrwxrwx 1 root root 14 Dec 14 12:17 S50inet->../init.d/inet
Irwxrwxrwx 1 root root 15 Dec 14 12:17 K10pnserver->../init.d/pnserver
Irwxrwxrwx 1 root root 15 Dec 14 12:17 S55named->../init.d/named
Irwxrwxrwx 1 root root 13 Dec 14 12:17 S60lpd->../init.d/lpd
Irwxrwxrwx 1 root root 13 Jar 31 20:17 S72amd->../init.d/amd
Irwxrwxrwx 1 root root 18 Dec 14 12:17 S75keytable->../init.d/keytable
Irwxrwxrwx 1 root root 18 Dec 14 12:17 S80sendmail->../init.d/sendmail
```

Bảng sau sẽ giới thiệu vài script kích hoạt quan trọng trong thư mục ấy.

Bảng 9.2: Các script của rc.3 init

Tên script	Daemon	Mô tả
S15nfsfs	nfs	Xử lý dịch vụ tệp mạng NFS
S30syslog	syslog	Ghi lại các thông báo hệ thống vào /var/log/messages
S40atd	atd	Cho phép các user thực hiện một công việc nhất định vào thời gian nhất định qua lệnh atd
S40crond	cron	Lập thời biểu **batch** cho Linux qua lệnh **crontab**
S50inet	inetd	Super server PID 1
S55named	name server	Cung cấp dịch vụ tên miền DNS
S601pd	lpd	Nạp daemon máy in để chuẩn bị in

SysV init kiểm tra tất cả các tệp trong thư mục cấp chạy được chỉ định và chuyển một trong hai tham số start hay stop dựa vào ký tự đầu tiên của tên tệp.

Xem "Kết nối hệ thống"

Thư mục rc.d còn chứa ba hay bốn tệp gọi là rc, rc.local, rc.sysinit và (hoặc không có) rc.serial. Shell script mang tên rc có nhiệm vụ khởi động lại hệ thống ở một cấp chạy khác, script này sẽ căn cứ vào tham số là con số tương ứng với cấp chạy mới. Sau khi

tiến trình khởi động kích hoạt xong tất cả các script khác thì mới đến lượt tệp rc.local. Bạn có thể đặt bất kỳ chỉ lệnh kích hoạt tại chỗ nào vào tệp này. Tệp rc.local sau đây giới thiệu một thí dụ về việc kích hoạt một tiến trình tại chỗ, gọi là secure shell, tiến trình này cho phép truy cập từ xa có bảo vệ hệ thống.

```
#!/bin/sh
# This script will be executed *after*
# all the other init scripts
# You can put your own initialization stuff
# in here if you don't
# want to do the full sys v style init stuff.
if [-f/etc/redhat-release]; then
R=$(cat / etc/redhat-release)
else
R=release 3.0.3
if
arch=$(uname-m)
a=a
case_$arch in
_a*) a=an;;
_i*) a=an;;
esac
# This will overwrite /etc/inssue at every boot.
# So, make any changes you want to make to /etc/issue
# here or you will lose them when you reboot.
echo >/etc/issue
echo RedHat Linux $R>> /etc/issue
echo Kernel $(uname-r) on $a$ (uname-m)>> /etc/issue
cp-f /etc/issue/etc/issue.net
echo>> /etc/issue
##Start sshd
/usr/local/sbin/sshd
```

rc.sysinit là tệp đầu tiên mà SysV init chạy ngay khi khởi hành. Script này thực thi một số chức năng, chẳng hạn như thiết lập các biến cho cả hệ thống (thí dụ như hostname), kiểm tra hệ thống tệp và bắt đầu sửa chữa, bật mở quota của các user và nạp hệ thống tệp /proc. Script trên cũng kích hoạt một tiến trình tại chỗ mang tên sshd, vốn là một shell daemon an toàn, chuyên hỗ trợ các lệnh từ xa và telnet.

Cấp chạy mặc định được quyết định tại /etc/initab bằng lệnh:

```
id:3:initdefault:
```

Nhận được lệnh này, khi khởi động hệ thống sẽ chạy ở cấp 3 (chế độ multiuser và multitasking hỗ trợ hoàn toàn các dịch vụ mạng).

Sau đây là một tệp /etc/inittab điển hình:

```
# initrtab This file describes how the INIT process
# hold set up the system in a certain run-level
#
# Author: Mique1 van Smoorenburg. <miquels@drinkel.n1.mugnet.org>
# Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default run level. The run levels used by RHS are:
# 0-halt (Do NOT set initdefault to this)
# 1-Single user mode
# 2-Multiuser.without NFS (The same as 3.
# if you do not have networking)
# 3-Full multiuser mode
# 4-unused
# 5-X11
# 6-reboot (Do Not set initdefault to this)
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
# Things to run in every run level.
ud::once:/sbin/update
# Trap CTRL-ALT-DELETE - Allow shutdown system with
# key combination CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
# When our UPS tells us power has failed
# assume we have a few minutes
# of power left. Schedule a shutdown
# for 2 minutes from now
# This does, of course.
# assume you have power installed and your
# UPS connected and working correctly
pf::powerfail:/sbin/shutdown -f -h +2 “Power Failure: System Shutting Down”
# If power was restored
```

```
# before the shutdown kicked in cancel it
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored: Shutdown Cancelled"
# Run gettys in standard run levels
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
# Run xdm in run level 5
x:5:respawn:/usr/bin/x11/xdm-nodaemon
```

Bạn đừng vội chạy mặc định ở cấp 0 hoặc cấp 6 nếu không hệ thống sẽ không làm gì cả vì nó sẽ dừng ngay khi khởi động lại (restart) hay mồi lại (reboot) liên tục, đôi khi còn làm hỏng tệp. Nếu vì lý do nào đó mà tệp inittab bị hỏng, hay không thể đăng nhập được (quên mật khẩu hay tệp passwd bị hỏng), bạn có thể khởi động lại ở chế độ single-user để sửa chữa. Để làm như thế, tại dấu nhắc khởi động LILO, bạn gõ tham số "linux single" như sau:

```
LILO boot: linux single
```

LILO là chương trình mồi Linux, được bàn kỹ trong giáo trình này.

Xem mục "Trình mồi LILO"

Còn nếu bạn đã cài GRUB, bạn cần thực hiện lần lượt các bước sau:

1. *Tại màn hình khởi động GRUB, chọn Red Hat Linux và bấm phím <e> để sửa.*
2. *Chọn dòng kernel và bấm <e>*
3. *Tại dấu nhắc, gõ "single" và <Enter>*
4. *Khi đã trở lại màn hình GRUB với thông tin về kernel, bấm <b> để khởi động cấp chạy 1.*

9.5 Mồi Linux bằng đĩa mềm

Nhiều người sử dụng đĩa mềm để mồi Linux. Đĩa này chứa bản sao lõi Linux ở root của hệ thống tệp Linux nằm trên phân vùng tương ứng trong ổ cứng. Các chương trình cài đặt của Red Hat và Caldera có thể giúp bạn tạo ra đĩa mềm mồi tương ứng.

Lưu ý: Ngay cả khi đã cài đặt một trình mồi trên ổ cứng, bạn vẫn nên tạo một đĩa mềm mồi Linux. Nếu chẳng may ổ cứng của bạn bị hỏng, đĩa mềm này sẽ giúp bạn khởi động hệ thống. Còn nếu sau khi hệ thống của bạn sụp rồi bạn mới tạo đĩa mềm mồi bằng một máy khác thì đĩa mềm mới này sẽ không giúp gì được cho hệ thống cũ nữa!

Ngoài ra các đĩa mềm mồi cũng giúp bạn trong trường hợp nguy cấp: Tại dấu nhắc khởi động, bạn gõ tuỳ chọn cấp cứu cho kernel. Sau khi hỏi qua vài điều, hệ thống sẽ yêu cầu bạn đưa đĩa bổ sung vào để hoàn tất tiến trình khởi động.

Sau khi khởi động, hệ thống sẽ cung cấp cho bạn một shell tối thiểu tên là bash, cùng với nhiều lệnh cứu hộ khác. Bảng 9.3 liệt kê những tiện ích này và chúng đủ giúp bạn sửa chữa hệ thống.

Bảng 9.3: Các tiện ích cứu hộ

Tiện ích	Mô tả
cat	Hiển thị nội dung tệp
chmod	Thay đổi các quyền truy cập tệp
cpio	Chép tệp từ archive
e2fsck	Kiểm tra hệ thống tệp mở rộng thứ nhì của Linux
fdisk	Thao tác bảng phân vùng Linux
gzip/gunzip	Nén hoặc giải nén tệp
insmod	Cài đặt module lõi (kernel)
ls	Liệt kê các tệp
mkdir	Tạo ra thư mục
mke2fs	Tạo ra hệ thống tệp mở rộng thứ nhì của Linux
mount	Lắp ghép (mount) một hệ thống tệp
rm	Gỡ bỏ tệp
rmmod	Gỡ bỏ một module
chkconfig	Công cụ sửa chữa và bảo dưỡng thư mục /etc/rc.d/init.d
ntsysv	giống như chkconfig nhưng có giao diện dạng văn bản
serviceconf	giống như chkconfig nhưng có giao diện đồ hoạ

Xem "Sử dụng các lệnh căn bản"

Lưu ý: Với các hệ thống Intel có phân vùng root nằm trên ổ cứng loại IDE, bạn có thể sử dụng đĩa mồi để khởi động Linux. Tại dấu nhắc khởi động, bạn gõ lệnh:

```
linux single root = /dev/hdal initrd =
```

Hãy khai báo thiết bị tích hợp hệ thống tệp root nếu nó nằm trên một thiết bị IDE khác ổ cứng /dev/hda1. Lệnh trên sẽ **mount** phân vùng root, chuyển sang chế độ single-user tức khắc, sau đó bỏ qua phần còn lại của tiến trình khởi động bằng đĩa mồi. Tiến trình này sẽ không làm việc nếu phân vùng root của bạn nằm trên ổ cứng loại SCSI.

9.6 Khởi động bằng trình mồi

Linux vốn có một chương trình quản lý mồi gọi là LILO, từ phiên bản Red Hat Linux 7.x trở đi lại có thêm trình mồi GRUB. Thông thường LILO cài đặt vào rãnh chứa trình mồi chủ (MBR: Master Boot Record) của ổ cứng đầu tiên và cho phép bạn chọn hệ điều hành để khởi động. Bạn cũng có thể thiết lập để LILO lúc đầu chạy được từ một phân vùng khác. Trình mồi sử dụng tiện lợi mà cũng bất tiện. Đương nhiên với

trình mồi bạn không cần phải có đĩa mềm mới khởi động được Linux. Ngoài ra bạn có thể chọn khởi động một hệ điều hành khác, hoặc chọn mặc định cho hệ điều hành hay dùng nhất. Điểm bất tiện là trình mồi làm cho tiến trình khởi động thêm phức tạp. Mỗi lần bạn thêm bớt, nâng cấp bất kỳ hệ điều hành nào trên ổ cứng, bạn đều phải chỉnh sửa lại, thậm chí cài đặt lại trình mồi.

Vì trình mồi thay đổi rãnh MBR trên ổ cứng, cho nên nếu xảy ra hỏng hóc bạn lại phải khởi động bằng đĩa mềm. Hơn nữa, trình mồi mà bạn chọn lại chưa chắc tương thích với vài hệ điều hành nào đó. Vậy bạn phải xác định nhu cầu sử dụng máy của mình trước khi quyết định xem dùng đĩa mềm hay trình mồi từ ổ cứng.

9.7 Đóng tắt Linux

Với Linux, bạn nên cẩn thận khi đóng tắt, không được tắt điện ngay. Linux lưu các thông tin I/O của hệ thống tệp trong vùng đệm bộ nhớ, do đó việc bạn tắt điện đột ngột có thể sẽ làm hệ thống tệp hỏng hóc.

Cách tốt nhất để đóng tắt Linux là sử dụng lệnh shutdown, với cú pháp như sau:

```
/sbin/shutdown [các cờ] thời gian [thông báo]
```

"thông báo" là thông báo cho toàn thể các user hiện đang làm việc, và "thời gian" là thời gian mà việc đóng tắt xảy ra. Đối số thời gian có những dạng như sau:

- Dạng thời gian tuyệt đối theo kiểu gg:pp, với gg là số giờ (một hoặc hai chữ số) và pp là số phút của thời gian ấy. Giá trị pp phải có hai chữ số.

- Dạng +m, với m là số phút phải chờ trước khi đóng tắt. Bạn có thể thay +0 bằng từ now khi muốn tắt ngay. Thí dụ:

```
[root@web root]#/sbin/shutdown -h now
```

*Bảng 9.4: Các cờ **(flag)** của lệnh shutdown*

Cờ	**Ý nghĩa**
-t sec	Chờ một thời gian đã được xác định bằng số giây giữa thời điểm phát lời cảnh báo và thời điểm đóng tắt tất cả mọi tiến trình. Thời gian trì hoãn này đủ để các chương trình khác kết thúc tiến trình đóng tắt riêng biệt
- k	Không thực sự đóng tắt hệ thống mà chỉ cảnh báo các user khác
- r	Khởi động lại (reboot) sau khi đóng tắt
- h	Dừng hẳn hệ thống (halt) sau khi đóng tắt
- n	Đừng đồng bộ hoá các đĩa trước khi khởi động lại hoặc dừng hẳn. Bạn cẩn thận khi dùng flag này, vì có nguy cơ làm hỏng dữ liệu
- f	Khởi động "nhanh". Flag này sẽ tạo ra tệp /etc/fastboot. Boot script có tên rc sẽ tìm tệp này và nếu có, sẽ không thực hiện fsck
- c	Huỷ một tiến trình đóng tắt đang thi hành. Flag này vô hiệu hoá đối số thời gian

Lệnh shutdown thông báo các user biết hệ thống sắp đóng tắt, đóng cổng đăng nhập hệ thống, sau đó gửi một tín hiệu SIGTERM đến tất cả các tiến trình để thực hiện việc tắt

đúng cách. Tiếp theo shutdown dựa vào sự chọn lựa của bạn ở dòng lệnh shutdown để khởi động lại hoặc dừng hẳn.

Lưu ý: Bạn muốn dừng hẳn hoặc khởi động lại bằng cách gõ trực tiếp lệnh halt hoặc reboot cũng được. Tuy nhiên nếu làm như vậy thì hệ thống sẽ đóng tắt tức khắc mà các user khác không thể biết trước. Bạn chỉ nên dùng hai lệnh vừa kể khi chỉ có một mình sử dụng hệ thống. Muốn biết trên mạng còn user nào không, bạn bấm <w> hoặc dùng lệnh who.

Chương 10. Quản lý trương khoản

Muốn làm quản trị viên Linux, bạn phải tìm hiểu các công cụ và kỹ thuật khác nhau để quản lý trương khoản của các user. Tác vụ này bao gồm việc đăng ký các user mới để họ có thể đăng nhập hệ thống; thiết lập các quyền ưu tiên cho họ; tạo ra các thư mục "nhà" (home directory) và gán chúng cho họ; sắp xếp user nào vào nhóm nào và ngược lại sẽ xoá tên của một vài user trong hệ thống khi cần thiết.

Các chủ đề chính sẽ được đề cập đến trong chương này bao gồm:

- *Làm việc với các user*
- *Làm việc với nhóm user*
- *Quản lý home directory*
- *Quản trị qua giao diện web*

10.1 Làm việc với các user

Bạn cần cho mỗi user một tên đăng nhập đặc trưng và riêng biệt. Tên này giúp bạn nhận dạng từng user và tránh được tình trạng người này xoá dữ liệu của người kia.

Mỗi user phải có mật khẩu riêng. User chỉ không cần mật khẩu khi có một mình trong hệ thống và hoàn toàn không kết nối với ai khác qua modem hoặc qua mạng.

Khi có lý do chính đáng để bỏ ai đó ra ngoài hệ thống, quản trị viên phải xoá tên đăng nhập của người ấy cùng với các tệp mà các user khác không còn nhu cầu sử dụng nữa.

Xem "Xử lý vấn đề an ninh mật khẩu"

10.1.1 Thêm vào một user

Khi thêm một user vào hệ thống, tệp mật khẩu có thêm một mục với dạng như sau:

```
logname:encrpt_passwd:userID:groupID:userInfor:login_directory:login_shell
```

Trong cú pháp trên, các trường được phân cách bằng dấu hai chấm.

Bảng 10.1: Các trường trong một mục ghi của tệp /etc/paswd

Trường	**Mô tả**
logname	Tên dùng khi đăng nhập
encrpt_passwd	Mật khẩu để nhận dạng user; đây là biện pháp đầu tiên để ngăn chặn việc vi phạm an ninh. Trường này thường được mã hoá
userID	Số định danh mà hệ thống dùng để phân biệt từng user
groupID	Số định danh hoặc tên đặc trưng dùng để nhận dạng nhóm sơ cấp gán cho user. Nếu thuộc về nhiều nhóm, một user có thể chuyển qua lại các nhóm khác khi được quản trị viên cho phép
userInfo	Thông tin về user, chẳng hạn như tên họ hoặc chức vụ

login_directory	Home directory của user, nơi user có mặt khi đăng nhập
login_shell	Shell được user dùng sau khi đăng nhập (thí dụ /bin/bash nếu user dùng shell bash)

Thí dụ:

```
[root@web etc]# cat /etc/passwd
nva:45b9d200f5f4683e569dcaa3ed6f6fb:500:231:Nva:/home/nva:/bin/bash
pla:aeswb00feø90e9cd47a01d86514a976:511:787:Pla:/home/pla:/bin/bash
bhh:4d844e2e58e14a0b377b276065c86036:515:787:Bhh:/home/bhh:/bin/bash
root:8aaa77148543346ea015cf18c57b0ce4d:0:0:root:/root:/bin/bash
```

Lệnh **adduser** hoặc **useradd** sẽ thêm user vào hệ thống. Bạn gõ lệnh này với tên của user mà bạn muốn thêm vào. Mục sau sẽ nói chi tiết hơn về chủ đề này.

Trong thực tế kể từ phiên bản 6.x, RedHat Linux đã sử dụng công cụ shadow để bảo mật hơn nữa. Khi đó tệp passwd có dạng rút gọn, thí dụ như sau:

```
gdm:x:42:42::/home/gdm:/bin/bash
squid:x:23:23::/var/spool/squid:dev/null
nva:x:500:231:Ngo Van An:/home/ nva:/bin/bash
pla:x:511:787:Phan Lan Anh (PI):/home/pla:/bin/bash
bhh:x:515:787: Bui Huy Hung (PI):/home/bhh:/bin/bash
```

Trong đó x đóng vai trò đại diện cho mật khẩu thật đang được chứa trong tệp /etc/shadow, thí dụ:

```
nva:$1$32rbaeg0d$4UtJgG/7rMqQJypA7pf0p0:10857:0:9999:7:-1:-1:134537356
pla: $1$3j86/RIF$qVUOxa.ZuPXnGLWuAC6i/:11360:0:9999:7:-1:-1:134549706
bhh:$1$EdjW7kPY$Hyn/xTNttk8fsWsZyvQo.0:10857:0:9999:7:-1:-1:134538444
```

Xem "Công dụng của mật khẩu shadow"

10.1.2 Sử dụng lệnh adduser

Trong các phiên bản cũ (RedHat Linux 5.x), khi thêm vào một user, bạn chỉ việc gõ lệnh **adduser** kèm với tên của user ấy, thí dụ:

```
[root@digital alberto] # adduser lananh
Looking for first available UID...502
Looking for first availbable GID...502
Adding login:lananh...done.
Creating home directory: /home/lananh...done.
Creating mailbox: /var/spool/mail/lananh...done.
Don't forget to set the password.
```

Lệnh **adduser** chép các tệp có tên bắt đầu bằng dấu chấm (.) từ thư mục /etc/skel sang home directory của user. Thư mục /etc/skel phải có các tệp khuôn mẫu (template) để cho tất cả các user dùng chung. Những template ấy phải có các tệp cấu hình "riêng"

như .profile, .cshrc và .login để cấu hình shell; .mailrc để thiết lập thư điện tử; .emacs để cho các user có thể sử dụng **emacs** làm trình soạn thảo v.v.

Lệnh **adduser** là một shell script nằm ở thư mục /usr/sbin, do đó bạn có thể tuỳ nghi chỉnh sửa script này chạy theo ý mình khi cần thực hiện thêm vài động tác nữa lúc thêm trương khoản mới vào. Loại chỉnh sửa thường gặp nhất là bắt **adduser** cung cấp nguyên tên user thay vì ép một tên user mặc định vào tệp mật khẩu. Nếu không thích thay đổi trong script để script hỏi tên user, bạn phải thay đổi thủ công bằng lệnh chfn như sau:

```
# chfn bhh
changing finger information for bhh.
Name [RH Linux User]: tvl
Office []:
Office Phone []:
Home Phone []:
Finger information changed.
```

Bạn không thể dùng lệnh **adduser** để thiết lập mật khẩu cho trương khoản mà phải dùng lệnh passwd.

Trong các phiên bản từ 6.x trở đi, RedHat Linux đã chuyển script này thành một lệnh với các tham số như sau:

adduser [-u uid [-o]] [-g group] [-G group...]
[-d home] [-s shell] [-c comment] [-m [-k template]]
[-f inactive] [-e expire] [-p passwd] [-n] [-r] name
adduser -D [-g group] [-b base] [-s shell]
[-f inactive] [-e expire]

Trong đó:

Trường	**Ký hiệu (ghi chú)**
logname	Name (tên đăng nhập)
userID	uid (nên để tự động)
groupID	group (có thể dùng tên của nhóm hay dùng gid)
userInfo	comment (cần đóng ngoặc đơn hoặc kép cho thông tin sẽ điền)
login_directory	Home (thư mục "nhà" mặc định dùng khi đăng nhập)
login_shell	Shell (shell mặc định dùng khi đăng nhập)

Thí dụ:

```
[root@mail /boot]# adduser -u 401 -g mail -c "thu ky" office
[root@mail /boot]# adduser -g mail -c 'trien khai' develop
```

10.1.3 Thiết lập mật khẩu cho user

Bạn dùng lệnh passwd để thiết lập mật khẩu ban đầu cho user. Sau đó từng user sẽ tự thay mật khẩu theo ý mình khi đã đăng nhập vào hệ thống. Sau đây là các bước căn bản để sử dụng lệnh passwd.

1. Gõ lệnh và tên đăng nhập (thí dụ như passwd pla) rồi bấm <Enter>

2. Tại dấu nhắc New password:, bạn gõ mật khẩu vào.

3. Khi máy nhắc phải gõ mật khẩu lần nữa, bạn chỉ việc gõ lại mật khẩu mới như sau:

```
New password (again): mật_khẩu_mới
```

Rồi mật khẩu đó được mã hoá và cất vào tệp /etc/passwd.

Mật khẩu phải đáp ứng hai điều kiện: dài ít nhất sáu ký tự (tám ký tự thì an toàn hơn), có cả chữ thường và chữ hoa cùng với các dấu phân cách và chữ số.

Xem "Xử lý vấn đề an ninh mật khẩu"

Khi đăng ký thêm nhiều user, bạn thường thích cho họ mật khẩu ngắn gọn và dễ nhớ. Việc quy định số ký tự tối thiểu là ở bạn, theo mặc định RedHat Linux 7.x yêu cầu bạn phải nhập ít nhất 6 ký tự. Bạn có thể quy định lại thông qua trình linuxconf nhưng không nên giảm bớt số ký tự này và không nên chọn mật khẩu dễ nhớ cho người dùng, bởi vì một mật khẩu tốt là tuyến phòng thủ đầu tiên chống lại tin tặc. Bạn cần giải thích điều này cho các user biết và nên đều đặn thay đổi mật khẩu.

- Có những nơi mật khẩu được yêu cầu thay đổi hàng tuần. Tuy nhiên điều đó dễ đưa đến tình trạng quên và nhầm lẫn mật khẩu. Bạn cần nhắc nhở việc ghi mật khẩu ra giấy cũng sơ hở như việc chọn mật khẩu dễ nhớ cho người dùng.

- Để tránh tình trạng quên mật khẩu, có thể cho ghi lại mật khẩu đó, bỏ vào phong bì dán niêm phong và cất trong tủ sắt.

Mỗi khi một user được gán mật khẩu, mục ghi của tệp sẽ có dạng thí dụ như sau:

```
pla: Anh.89&^0gW:123:21:Phan Lan Anh:/users/pla:/bin/bash
```

Trường thứ hai là mật khẩu ở dạng mã hoá, không phải là những ký tự lung tung.

Ghi chú: Thỉnh thoảng các user lại quên mật khẩu của mình. Bạn không thể nhắc cho từng user nhớ mật khẩu riêng. Tuy nhiên bạn có thể xoá mật khẩu bị quên bằng cách dùng lệnh passwd thiết lập một mật khẩu mới và thông báo nó cho user biết để tự đặt lại mật khẩu. Với tư cách quản trị viên, bạn nên thiết lập quy trình xử lý trường hợp vừa kể và cho các user biết quy trình ấy.

10.1.4 Gỡ bỏ một user

Bạn có thể gỡ bỏ một user theo nhiều cấp độ khác nhau. Việc gỡ bỏ một user ra khỏi hệ thống không phải là động tác "một đi không trở lại". Sau đây là một vài tuỳ chọn:

Chỉ gỡ khả năng đăng nhập. Điều này có ích khi một user nào đó phải đi xa một thời gian và sau đó sẽ trở lại vào hệ thống. Thư mục, tệp và thông tin về nhóm của user đó được giữ nguyên. Quản trị viên chỉ phải chỉnh sửa tệp mật khẩu và gõ dấu sao (*) vào trường thứ hai của mục ghi user như sau:

```
pla:*:123:21:Phan Lan Anh:/users/pla:/bin/bash
```

Cách này không còn áp dụng trên các phiên bản mới với công cụ shadow.

Gỡ bỏ user khỏi tệp mật khẩu nhưng vẫn để dữ liệu của user trên hệ thống. Hình thức này có ích khi các user khác muốn sử dụng những tệp dữ liệu ấy, hoặc có ai đó nhận nhiệm vụ thay cho user cũ. Quản trị viên xoá mục ghi của user cũ khỏi tệp mật khẩu bằng trình soạn thảo hoặc bằng lệnh:

```
userdel tên_đăng_nhập.
```

Sau đó thay đổi quyền sở hữu và vị trí các tệp của user đó bằng lệnh chown và **mv.**

Gỡ bỏ user ra khỏi tệp mật khẩu và gỡ bỏ tất cả tệp thuộc sở hữu của user ấy. Đây là hình thức cao nhất và đầy đủ nhất để xoá bỏ một user. Quản trị viên xoá mục ghi của user ở tệp mật khẩu và huỷ luôn tất cả các tệp của user ấy trong toàn hệ thống theo lệnh **find** như sau:

```
find user_home_directory exec rm{}\;
```

Ta cũng có thể dùng lệnh userdel với tham số -r như sau:

```
userdel -r tên_đăng_nhập
```

Ghi chú: Nếu hệ thống của bạn sử dụng các tệp cấu hình khác, chẳng hạn như tệp bí danh e-mail, thì vì an toàn bạn cũng phải xoá tên user ở những tệp ấy.

10.2 Làm việc với nhóm

Mỗi user là thành viên của một nhóm. Tuỳ theo tính chất của mỗi nhóm, quản trị viên sẽ chỉ định nhóm ấy có khả năng gì hoặc ưu tiên nào. Thí dụ có nhóm chuyên về việc phân tích doanh số của công ty, quản trị viên sẽ cho quyền nhóm này truy cập phạm vi tệp rộng lớn hơn so với một nhóm khác chỉ chuyên về tìm tòi sản phẩm mới.

Tệp mật khẩu chứa thông tin của một user. Trong khi đó thông tin của cả nhóm được chứa tại tệp /etc/group. Sau đây là thí dụ một mục ghi:

```
office::21:tvl, pla, nva
```

Ở thí dụ này, tên nhóm là office, số định danh nhóm là 21 và các thành viên của nhóm là tvl, pla và nva. Các thư mục và tệp có thuộc tính quyền hạn gắn liền với chủ sở hữu, nhóm và các yếu tố khác. Một user có thể là thành viên của nhiều nhóm khác nhau và quản trị viên có thể thay đổi việc tham gia các nhóm của user.

10.2.1 Thêm vào một nhóm

Bạn có thể tạo ra một nhóm mới bằng cách chỉnh sửa trực tiếp trong tệp /etc/group và đưa thông tin của nhóm ấy vào làm một mục ghi.

Trong tệp /ect/group, mỗi nhóm đều có số định danh riêng. Linux chỉ quan tâm đến con số này chứ không quan tâm đến tên nhóm. Do đó nếu bạn cấp cho hai nhóm cùng một con số định danh, Linux sẽ coi đây chỉ là một nhóm.

10.2.2 Xoá bỏ một nhóm

Muốn xoá bỏ một nhóm, bạn xoá đi mục ghi tương ứng trong tệp /etc/group. Tiếp theo bạn phải chuyển tất cả các tệp với GID tương ứng sang cho một nhóm khác. Bạn dùng lệnh **find** như sau:

```
find /-gid nhóm_A find user_home_directory -exec chgrp nhóm_mới {}\;
```

Bạn cũng có thể dùng các lệnh về nhóm như: groupdel, groupmod và groupadd để xử lý các thao tác trên.

10.3 Quản lý home directory

Nếu bạn dự kiến hệ thống của mình sẽ có nhiều user, hãy suy nghĩ trước về cách sắp xếp các home directory cho hợp lý. Cần cố gắng đặt tất cả các home directory của cùng một hệ thống vào một thư mục cấp cao nhất. Như thế, nếu sau này bạn có sắp xếp chúng lại như thế nào thì chúng cũng nằm chung với nhau.

Xem "Mount và Unmount hệ thống tệp"

Thí dụ bạn có thể chỉ định rằng thư mục /home là thư mục cấp cao nhất dành cho các thư mục của user. Dưới thư mục /home này, bạn tập hợp các user theo từng khối. Các trương khoản cho user của khối office (văn phòng) sẽ nằm ở /home/office, các trương khoản cho user của khối develop (triển khai) sẽ nằm ở /home/develop, v.v. Từ đó các home directory của những user trên hệ thống máy bạn sẽ nằm trong những thư mục ấy, hoặc bạn sẽ tạo ra thêm thư mục mới khi có yêu cầu. Vì các thư mục cho phép user sẽ chiếm nhiều chỗ trên ổ cứng, bạn phải suy nghĩ cách xếp đặt các nhóm logic trên nhiều hệ thống tệp vật lý khác nhau. Khi cần thêm khoảng trống trên ổ cứng, bạn chỉ việc tạo ra một hạng mục mới cho các home directory mới và lắp hạng mục ấy vào một hệ thống tệp như là một điểm ghép (**mount point**) trong thư mục /home.

10.4 Quản trị qua giao diện web

Từ bản phát hành Red Hat 5.1 bắt đầu có một công cụ quản trị hệ thống mang tên Linuxconf. Công cụ này giúp bạn xử lý nhiều tác vụ quản trị, chẳng hạn như làm việc với user và với nhóm. Ngoài hai giao diện thường gặp là dòng lệnh và X Windows, Linuxconf còn hỗ trợ tác vụ quản trị qua giao diện web.

Bạn cũng có thể dùng một trình duyệt sẵn có trong giao diện đồ hoạ GNOME, gọi là Nautilus. Khi khởi động GNOME, công cụ Nautilus hoạt động ngay và mở cửa sổ "START HERE" trong đó bạn có thể thấy các biểu tượng "System settings", "Server Configuration" (tương tự trong "Control Panel" của Windows) để quản lý các cấu hình của hệ thống.

Minh hoạ 10.1: Màn hình "Start here" của Nautilus

Chương 11. Sao lưu dữ liệu

Có nhiều nguyên nhân làm hỏng hóc hoặc mất mát dữ liệu như tệp chẳng may bị xoá, trục trặc phần cứng, thông tin liên quan nằm ở những tệp không còn truy cập được. Một quản trị viên giỏi phải đảm bảo sao cho các user vẫn tham khảo được những tệp "đã mất" như thế. Muốn vậy, bạn phải sao lưu dữ liệu kịp thời.

Tương lai của cơ quan bạn - và tương lai của bạn tại cơ quan - có thể tuỳ thuộc vào việc các user truy cập được những tệp đã sao lưu ấy. Tại những thời điểm nóng bỏng như thế, bản thân bạn cũng như các người khác sẽ có cơ hội nhận thức giá trị của thời gian và công sức để sao chép dữ liệu một cách đều đặn, chặt chẽ và theo một thời điểm đầy đủ. Việc sao lưu tệp không mấy hấp dẫn, nhưng quản trị viên không thể không am tường tiến trình sao lưu.

Các chủ đề chính sẽ được đề cập đến trong chương này bao gồm:

- *Vấn đề về sao lưu*
- *Các thủ thuật sao lưu*
- *Hoạch định thời biểu sao lưu*
- *Thực hiện sao lưu và phục hồi tệp*

11.1 Vấn đề sao lưu

Khi sao lưu một hệ thống, bạn lưu ý một số câu hỏi như sau:

- Sao lưu toàn bộ hay tăng dần? Sao lưu toàn bộ dĩ nhiên rất an tâm nhưng bạn sẽ tốn nhiều thời gian và dung lượng ổ cứng để lưu tất cả các tệp. Trong khi đó việc sao lưu tăng dần chỉ sao lưu những tệp nào đã có thay đổi kể từ lần sao lưu gần nhất mà thôi.

- Cần sao lưu những hệ thống tệp nào? Những hệ thống tệp nào hiện hành thì cần được sao lưu đều đặn, những gì còn lại không nhất thiết phải làm thường xuyên. Là quản trị viên, bạn phải đảm bảo luôn sẵn có bản sao lưu của tất cả các hệ thống tệp.

- Lưu tệp trên những phương tiện nào? Tuỳ thuộc vào những thiết bị sẵn có của hệ thống, bạn có thể sao lưu dữ liệu trên các loại đĩa cứng, mềm, CD, quang-từ (MOD), DVD, hoặc các loại băng từ v.v. Mỗi loại đều khác nhau về giá cả, tốc độ và dung lượng, do đó bạn nên chọn loại nào cho vừa túi tiền, song không nên quên rằng loại rẻ nhất thường lại làm cho bạn tốn nhiều thì giờ nhất.

- Việc sao lưu sẽ ảnh hưởng ra sao đối với các user? Sao lưu luôn làm hệ thống cồng kềnh thêm. Liệu đây có phải là gánh nặng không chính đáng cho các user? Hơn nữa nếu có tệp nào thay đổi trong khi đang sao lưu thì bản thân tệp ấy sẽ không được sao lưu. Điều này thật phiền phức, nhất là khi bạn đang sao lưu một cơ sở dữ liệu quan trọng. Vậy chỉ nên sao lưu khi không còn ai khác sử dụng hệ thống?

- Liệu một vài câu lệnh tương đối đơn giản và được nhiều người sử dụng như **tar** và **cpio** có đáp ứng đủ yêu cầu của bạn?

- Làm sao có thông tin về các tệp đã sao lưu? Bạn phải ghi sổ hoặc tệp nhật ký và dán nhãn trên mỗi vật mang dữ liệu sao lưu để dễ tìm khi cần dùng đến. Một vài thủ tục và câu lệnh sẽ giúp bạn chuẩn bị mục lục hoặc danh sách những gì được sao lưu.

Quản trị viên cần sao lưu theo một quy trình tự động hoặc càng ít can thiệp thủ công càng tốt. Hơn nữa nên sao lưu khi nào hệ thống không còn ai khác sử dụng để bảo đảm an toàn.

Hai điều trên phải được cân bằng với sự tiện lợi và chi phí. Liệu một quản trị viên có phải thức cho đến tận nửa đêm cuối tuần để thực hiện sao lưu toàn bộ? Liệu có nên bỏ ra nhiều tiền mua băng từ DAT nhằm tự động hoá việc sao lưu toàn bộ hệ thống vào lúc ba giờ sáng mà không cần can thiệp thủ công? Bạn phải quan tâm những vấn đề như thế và nhớ rằng thông tin sao lưu được quản lý tốt sẽ giúp chúng ta tiết kiệm được tiền bạc và công sức.

11.2 Các thủ thuật sao lưu

Mục đích của sao lưu là để khi cần sẽ phục hồi được các tệp cá nhân hoặc tệp hệ thống thật dễ dàng và nhanh chóng. Cho dù bạn làm điều gì khi sao lưu thì cũng phải ghi nhớ mục đích ấy.

Bạn nên lập một kế hoạch sao lưu. Ghi ra những tệp cần sao lưu, thời biểu xử lý chúng và sau này sẽ phục hồi chúng như thế nào. Thông báo cho các user biết thời biểu cùng với yêu cầu phục hồi tệp. Kế hoạch đã lập ra như thế nào, bạn cứ làm theo như thế.

Sao lưu xong phải kiểm tra. Bạn thử đọc bảng mục lục trên vật mang dữ liệu sao lưu, hoặc thử phục hồi một tệp nào đó. Nên nhớ rằng bản thân vật mang dữ liệu sao lưu – tức băng hoặc đĩa – đều có thể bị lỗi.

Bạn phải sao lưu như thế nào để có thể phục hồi tệp ở một máy khác hoặc ở một hệ thống tệp khác trên cùng một máy. Bạn nên sử dụng các tiện ích sao lưu và tiện ích tồn trữ (archive) có khả năng tạo ra tệp tồn trữ mà các hệ Linux hoặc hệ khác đều đọc được.

Bảo đảm tìm thấy các tệp cần thiết bằng cách dán nhãn trên tất cả các vật mang dữ liệu sao lưu. Nếu có nhiều băng hoặc đĩa, phải ghi số thứ tự và ngày tháng sao lưu.

Hãy tránh các tai hoạ nhưng vẫn dự kiến việc chúng sẽ xảy ra và sao lưu các tệp như thế nào để bạn có thể phục hồi toàn bộ hệ thống với khoảng thời gian hợp lý. Nên cất các bản sao lưu ở vài nơi xa để đề phòng trường hợp hoả hoạn thiêu trụi hoàn toàn máy. Nhiều tổ chức đã thuê nơi an toàn để chứa các băng đĩa quan trọng. Bạn cất băng đĩa chỗ nào thì cũng nên cất kèm danh sách toàn bộ cấu hình phần cứng hệ thống.

Thỉnh thoảng bạn nên đánh giá lại các thủ tục sao lưu, xem chúng có thực sự thoả mãn yêu cầu hiện hành hay không.

Nhiều công cụ có thể giúp bạn tự động hoá tiến trình sao lưu. Bạn thử xem qua các tệp Linux tại sunsite.unc.edu để có thêm thông tin. Ngoài ra Linux cũng chấp nhận phần mở rộng FTAPE. FTAPE giúp bạn sao lưu dữ liệu lên băng từ QIC-80 thông qua giao diện đĩa mềm. Xem trên Internet tệp HOWTO về FTAPE để biết thêm chi tiết.

11.3 Hoạch định thời biểu sao lưu

Như đã nói ở trên, việc hoạch định thời biểu sao lưu là rất quan trọng, cần phù hợp với yêu cầu của hệ thống và quản trị viên cần tuân theo kế hoạch ấy.

Điều lý tưởng là khi có yêu cầu phục hồi tệp thì phải thực hiện được ngay. Thật ra cũng không nhất thiết cần cầu toàn như thế, song trên nguyên tắc quản trị viên phải nắm khả năng phục hồi tệp mỗi ngày. Để đạt mục đích này, bạn chọn giải pháp chung giữa hình thức sao lưu toàn bộ và sao lưu tăng dần. Như bạn đã biết, một bản sao lưu toàn bộ chứa tất cả các tệp của hệ thống, trong khi sao lưu tăng dần chỉ chứa những tệp nào đã bị thay đổi kể từ lúc sao lưu gần nhất.

Việc sao lưu tăng dần cũng có cấp độ khác nhau: tăng dần kể từ lần sao lưu toàn bộ gần nhất, hoặc tăng dần kể từ lần sao lưu tăng dần gần nhất. Sau đây chúng ta thử đặt cấp độ cho các hình thức sao lưu vừa kể:

- Cấp 0: sao lưu toàn bộ
- Cấp 1: sao lưu tăng dần so với lần sao lưu toàn bộ gần nhất
- Cấp 2: sao lưu tăng dần so với lần sao lưu tăng dần gần nhất.

Sau đây là 2 thí dụ về thời biểu sao lưu:

A. Một ngày sao lưu toàn bộ, các ngày khác sao lưu tăng dần.

Ngày 1: Cấp 0, sao lưu toàn bộ

Ngày 2: Cấp 1, sao lưu tăng dần

Ngày 3: Cấp 1, sao lưu tăng dần

Ngày 4: Cấp 1, sao lưu tăng dần

Ngày 5: Cấp 1, sao lưu tăng dần

Nếu bạn tạo ra và lưu trữ chỉ mục của mỗi lần sao lưu, sau này bạn chỉ cần bản sao lưu của một ngày để phục hồi tệp cá nhân và bản sao lưu của hai ngày (bản của ngày 1 và bản của một ngày khác) để phục hồi toàn bộ hệ thống.

B. Sao lưu toàn bộ mỗi tháng, sao lưu tăng dần mỗi tuần và tăng dần mỗi ngày.

Thứ ba tuần đầu tiên: Cấp 0, sao lưu toàn bộ.

Những ngày Thứ ba sau: Cấp 1, sao lưu tăng dần.

Mỗi ngày khác: Cấp 2, sao lưu tăng dần.

Theo cách này, nếu cần phục hồi một tệp chưa bị thay đổi gì suốt tháng qua thì bạn phải dùng đến bản sao lưu toàn bộ. Bạn phải dùng đến bản sao lưu Cấp 1 nếu nội dung tệp đã thay đổi ở tuần trước nhưng tuần này chưa thay đổi gì. Bạn sẽ phải dùng bản sao lưu Cấp 2 nếu tệp vừa kể đã bị thay đổi ngay trong tuần này.

Thời biểu này có vẻ phức tạp hơn thí dụ trước, song tác vụ sao lưu hàng ngày sẽ chiếm ít thời gian hơn.

Ngoài ra bạn nên lưu trữ các tệp sao lưu suốt một khoảng thời gian nào đó, phòng trường hợp cần phục hồi một phiên bản cũ. Thí dụ cứ mỗi tuần bạn lại thực hiện một

bản sao lưu toàn bộ rồi lưu cho bốn tuần sau đó. Nếu cần sao và tồn trữ lâu hơn nữa thì cứ mỗi nửa tháng bạn lại sao lưu toàn bộ cho một quý.

11.4 Thực hiện sao lưu và phục hồi tệp

Linux có nhiều tiện ích giúp bạn trong tác vụ này. Một số tiện ích thật đơn giản và một số khác phức tạp hơn. Càng đơn giản thì tiện ích càng bị giới hạn. Xin giới thiệu với bạn hai chương trình sau:

* **tar** sẵn có trên mọi hệ thống Linux hoặc UNIX.

* **cpio** sẵn có trên mọi hệ thống UNIX, dùng để sao chép tệp, **cpio** dễ dùng và mạnh hơn **tar.**

11.4.1 Tiện ích tar

Từ đầu, tiện ích **tar** của UNIX đã được thiết kế để tạo ra bản sao tồn trữ trên băng từ. Bạn có thể dùng **tar** để chép sang bất kỳ thiết bị nào, **tar** có nhiều thuận lợi như: dễ sử dụng, ổn định, có thể đọc tệp tồn trữ trên mọi hệ thống Linux hoặc UNIX.

Điểm bất tiện của một vài phiên bản **tar** là đòi hỏi phải lưu toàn bộ các tệp trên cùng một đĩa hoặc một băng. Do đó nếu có hỏng hóc một phần vật mang dữ liệu, chẳng hạn như đĩa bị bad sector hoặc băng bị bad block, thì coi như bạn mất trắng toàn bộ bản sao lưu. Tiếp theo, **tar** không thể sao lưu các tệp đặc biệt như tệp thiết bị. Và bản thân **tar** chỉ có thể thực hiện sao lưu toàn bộ mà thôi. Nếu cần sao lưu tăng dần, bạn phải lập trình qua shell.

Xem "Làm việc với shell script"

Bảng 11.1 Các tuỳ chọn thường dùng với lệnh tar

Tuỳ chọn	**Mô tả**
c	Tạo ra tệp tồn trữ.
x	Khai thác hoặc phục hồi tệp từ tệp tồn trữ trên thiết bị mặc định, hoặc trên thiết bị được xác định bằng tuỳ chọn f.
f tên	Tạo ra hoặc đọc tệp tồn trữ tên, với tên là tên của tệp hoặc tên của thiết bị được xác định ở /dev, chẳng hạn như /dev/rmt0.
Z	Nén hoặc bung tệp **tar.**
z	Nén hoặc bung tệp **tar** bằng **gzip**.
M	Tạo ra bảng sao lưu **tar** nhiều tập.
t	Tạo ra chỉ mục tất cả các tệp lưu trong bản tồn trữ và liệt kê với stdout.
v	Chọn chế độ chi tiết.

Mời bạn xem vài thí dụ về cách sử dụng **tar** để sao lưu và phục hồi tệp. Lệnh sau đây sẽ chép thư mục /home vào đĩa mềm /dev/fd0:

```
tar -cf /dev/fd0 /home
```

Trong trường hợp này tuỳ chọn f cho biết tệp tồn trữ được chứa trên đĩa mềm /dev/fd0.

Lệnh sau đây cũng sao lưu thư mục /home:

```
tar -cvfzM /dev/fd0 /home | tee homeindex
```

Tuỳ chọn v chỉ định chế độ chi tiết, z cho biết phải nén tệp tồn trữ và M bắt **tar** tạo ra bản sao lưu có nhiều tập. Mỗi khi đĩa mềm đã đầy, **tar** nhắc bạn đưa đĩa khác vào. Chỉ mục homeindex chứa danh sách các tệp được sao lưu để bạn tham khảo.

Lệnh **find** có ích trong việc tìm kiếm các tệp đã có thay đổi trong khoảng thời gian nào đó, với mục đích là sao lưu chúng theo thời biểu nhất định. Thí dụ sau đây dùng lệnh **find** để tạo danh sách các tệp đã thay đổi vào ngày cuối cùng:

```
find /home -mtime -1 -type f -print > bkuplst; tar cvfzM /dev/fd0 cat bkuplst | tee homeindex
```

Muốn dùng danh sách làm nhập liệu cho lệnh **tar,** bạn hãy đặt lệnh **cat** bkuplst trong dấu nháy ngược (còn gọi là dấu huyền), thí dụ: `**cat** bkuplst`. Lúc này shell sẽ hiểu là phải thực hiện lệnh trong một shell thứ cấp và sau đó hiển thị kết quả tại vị trí của lệnh nào được đặt trong dấu nháy ngược.

Lệnh sau đây phục hồi tệp /home/dave/notes.txt từ thiết bị /dev/fd0:

```
tar xvf /dev/fd0/ home/dave/notes.txt
```

Lưu ý: Bạn có thể tự động hoá các lệnh vừa kể bằng cách đưa chúng vào tệp **crontab** ở root. Thí dụ bạn đưa mục ghi này vào tệp **crontab** để sao lưu thư mục /home mỗi ngày vào lúc 1 giờ 30 sáng:

```
30 01 * * * tar cvfz / def/fd0 /home >homeindex
```

Khi cần thực hiện các sao lưu phức tạp hơn, bạn tạo ra các shell script điều khiển. Các shell script như thế có thể chạy qua trung gian **cron.**

*Xem "Lập thời biểu thực thi lệnh bằng **cron** và **crontab**"*

Ngoài ra bạn cũng có thể dùng lệnh **tar** để tạo các tệp tồn trữ bên trong hệ thống tệp của Linux thay vì ghi lên thiết bị sao lưu. Như thế, bạn có thể sao lưu một nhóm tệp cùng với cấu trúc thư mục của chúng chỉ vào một tệp duy nhất. Bạn chỉ cần ghi tên tệp vào chỗ đối số của tuỳ chọn f thay vì ghi tên thiết bị sao lưu. Sau đây là thí dụ về việc tạo tệp tồn trữ chứa các tệp của thư mục và thư mục cấp dưới bằng lệnh tar:

```
tar cvf home /sao_luu.tar /home/dave
```

Kết quả là bạn có tệp /home/sao_lưu.tar chứa thông tin sao lưu của thư mục /home/dave cùng với tất cả các tệp và thư mục cấp dưới của /home/dave.

Ghi chú: Bản thân lệnh **tar** không thực hiện việc nén tệp. Muốn nén tệp **tar,** bạn xác định tuỳ chọn z với lệnh **tar,** hoặc bạn dùng chương trình nén, chẳng hạn như **gzip** đối với tệp **tar** sau cùng.

Khi sử dụng **tar** để tạo tệp tồn trữ, bạn nên khai báo mục ghi cao cấp nhất trong tệp **tar** là một thư mục. Như thế mỗi khi bạn khai thác, tất cả các tệp chứa trong tệp **tar** đó đều nằm trong một thư mục trung tâm của thư mục hiện hành. Nếu không, bạn sẽ rối lên với hàng trăm tệp được bung ra trong thư mục hiện hành.

Giả sử bên dưới thư mục hiện hành của bạn là thư mục mang tên data, trong đó có hàng trăm tệp. Có hai cách cơ bản để tạo tệp **tar** cho thư mục này. Cách thứ nhất, bạn chuyển sang thư mục data và tạo tệp **tar** từ đấy:

```
$ pwd
/home/dave
$ cd data
$ pwd
/home/dave/data
$ tar cvf.. /data.tar*
```

Kết quả là một tệp **tar** được tạo ra trong /home/dave chỉ chứa nội dung của thư mục data nhưng không chứa mục ghi nào cho thư mục. Khi ra lệnh khai thác tệp **tar** này, Linux sẽ không tạo ra một thư mục để chứa các tệp, ngược lại hàng trăm tệp sẽ được bung vào thư mục hiện hành.

Cách thứ hai để tạo tệp **tar** là khởi đầu từ thư mục mẹ của data và xác định đối tượng để sao lưu chính là tên thư mục, thí dụ:

```
$ pwd
/home/dave
$ tar cvf data.tar data
```

Thí dụ trên cũng tạo ra một tệp tồn trữ của thư mục data, nhưng mục ghi thư mục là điều đầu tiên được tạo ra trong tệp tồn trữ ấy. Do đó sau này khi khai thác tệp **tar,** cái đầu tiên được tạo ra chính là thư mục data và tất cả các tệp bên trong thư mục data sẽ được xếp vào trong thư mục con data.

Ghi chú: Nếu muốn tạo ra tệp **tar** chứa tất cả các tệp trong thư mục, bạn nên xác định một vị trí riêng biệt cho tệp **tar,** không nằm trong thư mục hiện hành, lệnh **tar** sẽ không bị nhầm lẫn khi cố gắng đưa tệp **tar** sẵn có trước đó vào tệp **tar** đang được tạo ra lần thứ hai.

11.4.2 Sử dụng cpio

Lệnh tổng quát để chép các tệp tồn trữ là **cpio**. Bạn có thể tạo ra các bản sao lưu bằng cách sử dụng **cpio** với tuỳ chọn -0, hoặc để phục hồi tệp bằng tuỳ chọn -i. Lệnh **cpio** sẽ lấy nhập liệu từ đầu vào tiêu chuẩn (Stdin) và gửi xuất liệu đến đầu ra tiêu chuẩn (Stdout).

cpio có những lợi điểm như:

- Có khả năng sao lưu bất kỳ nhóm tệp nào.

- Có khả năng sao lưu những tệp đặc biệt.

- Lưu trữ thông tin hiệu quả hơn **tar.**

- Khi phục hồi dữ liệu, **cpio** bỏ qua các bad sector và bad block.

- Bản sao lưu của **cpio** có thể phục hồi được trên hầu hết các hệ thống Linux hoặc UNIX.

Có người cho rằng cú pháp của **cpio** khó hơn cú pháp của **tar.** Ngoài ra khi thực hiện sao lưu tăng dần, bạn phải thực hiện lập trình qua shell.

*Bảng 11.2 Các tuỳ chọn thường dùng với **cpio***

Tuỳ chọn	**Mô tả**
-0	Chép và tạo ra một tệp tồn trữ ở đầu ra tiêu chuẩn
-B	Hạn chế xuất liệu và nhập liệu ở mức 5.120 byte cho từng mẫu tin. Có ích khi lưu trữ trên băng từ.
-I	Khai thác và sao chép các tệp từ stdin. Được dùng đặc biệt khi Stdin lại là bản sao từ đầu ra của một lệnh **cpio** khác.
-t	Tạo bảng mục lục của nhập liệu.

Sau đây là vài thí dụ sử dụng **cpio** để sao lưu và phục hồi tệp: Lệnh **ls** sẽ sao chép các tệp trong thư mục /home sang thiết bị /dev/fd0, còn lệnh **find** sẽ cho chép toàn bộ cây thư mục của /home (kể cả thư mục con):

```
ls /home | cpio -0> /dev/fd0
```

```
find /home/. | cpio -ov> dev/fd0
```

Lệnh sau đây sẽ khai thác các tệp từ thiết bị /dev/fd0, đồng thời tạo ra bảng chỉ mục trong tệp bkup.indx:

```
cpio -it</dev/fd10>bkup.indx
```

Thí dụ sau đây sử dụng lệnh **find** để tạo ra danh sách những tệp nào của thư mục /home đã thay đổi nội dung vào ngày gần nhất:

```
cpio -i /home/dave/notes.txt < /dev/fd0
```

Ghi chú: Bạn phải ghi tên đầy đủ của tệp khi phục hồi tệp ấy bằng **cpio**

Lưu ý: Muốn tự động hoá tất cả các lệnh vừa kể, bạn đặt chúng vào tệp **crontab** của root. Thí dụ, bạn đặt mục ghi sau đây vào tệp **cron** ở root để máy tự động sao lưu thư mục /home vào lúc 1 giờ 30 mỗi ngày:

```
30 01 *** find /home/. | cpio -0 > /dev.fd0
```

Nếu muốn thực hiện những sao lưu phức tạp hơn, bạn tạo ra các shell script điều khiển và chạy chúng qua **cron.**

Các bảng 11.1 và 11.2 đã liệt kê các tuỳ chọn thường dùng cho **tar** và **cpio**. Muốn biết đầy đủ chi tiết về các tuỳ chọn, mời bạn tham khảo bằng lệnh **man.**

Chương 12. An ninh hệ thống

Có phải bạn là người duy nhất sử dụng máy tính không nối mạng, dùng xong lại cất trong phòng riêng, khoá cửa luôn và cẩn thận mang theo bên mình chiếc chìa khoá? Câu hỏi này không phải để đùa. Nếu hệ thống có hơn một user, máy lại kết nối với thế giới bên ngoài qua modem hoặc qua mạng và nếu có những khoảng thời gian bạn không trực tiếp quản lý máy, thì luôn có khả năng xảy ra đột nhập trái phép.

Đôi khi sự đột nhập này không có gì trầm trọng ngoài việc làm cho bạn bực mình. Nếu ai đó bỏ công sức ra để tìm cách đột nhập máy bạn, hẳn người đó phải biết cách sao chép những thông tin bạn định giữ bí mật. Ngoài ra họ còn có thể sử dụng tài nguyên hệ thống, hoặc xoá bỏ dữ liệu của bạn.

Hầu hết các cơ quan đều giao trách nhiệm bảo vệ an ninh hệ thống cho quản trị viên. Tuy không nên quá lo lắng về việc này, song bạn vẫn phải nhận thức đầy đủ những rủi ro, từ đó có biện pháp bảo vệ hệ thống.

Chương này trình bày một số biện pháp và chính sách, cùng với các kỹ thuật hiện hành nhằm tăng cường an ninh máy tính. Một số điểm sẽ phù hợp với quy mô của các cơ quan lớn, một số khác lại có ích cho các máy riêng chỉ chạy như ở nhà.

- *Vấn đề an ninh vật lý.*
- *Vấn đề an ninh mật khẩu.*
- *Triển khai an ninh đăng nhập.*
- *Vấn đề an ninh tệp.*
- *Cảnh giác với tin tặc.*
- *Theo dõi việc sử dụng lệnh su.*
- *Triển khai một hệ thống an toàn.*
- *Các mô-đun xác thực PAM.*
- *Công dụng của shadow password.*

Ghi chú: Những năm gần đây, các báo đài thường dùng từ hacker theo nghĩa "người đột nhập hệ thống máy tính" mà quên đi nghĩa đầu là "người say mê máy tính". Trong giới tin học chuyên nghiệp thì người đột nhập hệ thống với ý xấu bị gọi là cracker (tin tặc).

12.1 Vấn đề an ninh vật lý

Vì báo đài nói nhiều về virus tin học hoặc về việc đột nhập qua modem và mạng của những tay tin tặc, cho nên ít ai để ý đến khía cạnh vật lý của vấn đề an ninh máy tính. Thực ra các thiết bị tin học khá nhạy cảm với nhiều điều kiện môi trường.

Chẳng hạn như khói lửa đương nhiên gây nguy hiểm nhanh chóng cho máy móc. Do đó nếu là quản trị viên, bạn phải để nghị lắp đặt các bộ cảm biến khói, thiết bị chữa cháy tự động, cùng với hệ thống báo động cứu hoả.

Ngoài ra bụi cũng là kẻ thù của máy. Bụi có tính mài mòn và rút ngắn tuổi thọ của thiết bị cơ học, của các ổ đĩa và băng từ. Bụi đóng kín các kênh thông gió làm cho máy bị nóng quá độ. Cuối cùng, bụi gặp hơi nước có thể dẫn điện và làm chập mạch.

Máy tính rất nhạy cảm với tình trạng điện áp chập chờn, do đó mọi máy móc tin học đều phải được nối qua thiết bị ổn áp. Phải bảo vệ cả các modem kết nối qua đường điện thoại.

Ghi chú: Mặc dù ổn áp cũng giúp máy tính chống lại tình trạng điện áp lưới tăng giảm đột ngột, song lại bất lực trong trường hợp sét đánh. Do đó khi có bão giông lớn, tốt hơn là bạn nên rút phích cắm điện và chờ cơn bão đi qua, trừ trường hợp ổn áp của bạn có thêm phần chống sét.

Ngoài ra, thiết bị tin học là những thứ đắt tiền, do đó phải phòng chống trộm cắp, ngay cả trộm cắp linh kiện như các bìa, cáp và chip.

Một việc thường được nhắc đến là ngăn cản đột nhập trái phép. Quản trị viên phải kiểm tra việc vào ra hệ thống máy để tạo rào cản cho những ai muốn đánh cắp hoặc làm hỏng dữ liệu và thiết bị. Bạn nên thiết lập chính sách, chuẩn mực sử dụng máy và truy cập thông tin, sau đó phổ biến chúng đến các user của hệ thống. Sau đây là một vài biện pháp nhằm cải tiến an ninh vật lý của hệ thống:

- Không để hệ thống, ổ băng đĩa, terminal và trạm làm việc không có người trông coi trong một thời gian kéo dài. Phải có biện pháp hạn chế ra vào phòng lưu trữ thông tin. Tuy đơn giản, song việc cửa có khoá sẽ luôn luôn mang lại hiệu quả.

- Không bỏ trống hệ thống khi có người đăng nhập với tư cách là root. Nếu một user biết rõ hệ thống, người ấy có khả năng sử dụng những ưu đãi dành cho root rồi từ đó chỉnh sửa các phần mềm quan trọng, hoặc thao tác toàn bộ thông tin của hệ thống.

- Ngoài ra với Linux, khi không biết mật khẩu của root, user vẫn có thể thay đổi nó bằng một biện pháp đơn giản là khởi động hệ thống ở chế độ Single, sau đó đổi mật khẩu của root. Do đó ta cần xác lập thêm mật khẩu khởi động và ngăn user mở máy.

- Phổ biến cho các user biết những rủi ro về an ninh vật lý, khuyến khích họ báo cáo các trường hợp đột nhập trái phép mà họ bắt gặp. Tuy mềm dẻo, song quản trị viên phải biết cương quyết ngăn cấm người lạ sử dụng hệ thống.

- Nếu có thể được, bạn đừng lưu trữ thông tin nhạy cảm trên những hệ thống nào có kết nối mạng hoặc modem.

- Giữ những bản sao lưu ở nơi an toàn và hạn chế ra vào.

12.2 Vấn đề an ninh mật khẩu

Phòng tuyến đầu tiên chống đột nhập trái phép chính là mật khẩu. Nhiều khi đây cũng chính là tuyến yếu nhất của hệ thống. Mục này sẽ mô tả vài biện pháp bảo mật.

Khuynh hướng thông thường của các user là sử dụng mật khẩu đơn giản dễ nhớ và họ cũng lười thay đổi mật khẩu. Họ lại có thói quen ghi mật khẩu ra đâu đó để tiện tham khảo và như vậy tạo thành gánh nặng cho quản trị viên.

Mật khẩu của root hay superuser là rất quan trọng. Ai nắm mật khẩu này sẽ nắm quyền vào ra và thao túng cả hệ thống, chưa kể những hệ thống khác kết nối với hệ thống

này. Do đó bạn nên chọn mật khẩu không quá đơn giản, thường xuyên thay đổi nó và cất kỹ. Các cơ quan lớn giao mật khẩu cho hai quản trị viên, nhưng chỉ hai mà thôi. Mật khẩu phải dài ít nhất sáu ký tự, tuy nhiên máy chỉ hiểu tám ký tự đầu tiên (trong đa số trường hợp, trừ khi bạn có bật chế độ bảo mật MD5).

Viết ra một chương trình đoán mật khẩu là việc không mấy khó khăn, do đó nếu mật khẩu dài hơn thì chương trình sẽ mất nhiều thì giờ hơn để đoán.

Máy tính rất hữu hiệu khi làm một việc gì đó có tính lặp đi lặp lại, chẳng hạn như so sánh từng chữ trong quyển từ điển với mật khẩu máy bạn. Do đó bạn đừng bao giờ dùng mật khẩu nào trùng với một chữ trong từ điển. Ngoài ra bạn cũng không nên chọn mật khẩu nào có dính dáng đến bản thân, chẳng hạn như tên mình, địa chỉ, tên vợ chồng, tên con cái, số điện thoại, số CMND hoặc số xe, v.v...

Một trong những kỹ thuật chọn mật khẩu là chọn hai từ ngắn bất kỳ, rồi nối chúng lại bằng một dấu hiệu riêng. Như thế bạn dễ nhớ mật khẩu, trong khi lại tạo ra xác suất đoán trúng nhỏ hơn cho người đoán. Sau đây là vài thí dụ:

```
ngovan&dongda
canhchan!phocu
chinhanh%buudien
```

Một cách nữa là tìm một câu bạn dễ nhớ, nhưng sau đó chỉ sử dụng những ký tự đầu của mỗi từ mà thôi. Thí dụ câu “em và anh cùng ngắm sao” sẽ thành “e&acn*”

Điều quan trọng là bạn phải nhớ mật khẩu chứ không được ghi ra đâu đó. Nếu các user sợ quên và cần ghi lại mật khẩu, bạn phải chỉ cho họ cách “mã hoá” như vừa rồi, hoặc cao cấp hơn. Chẳng hạn mật khẩu của bạn là “75%salary” thì bạn có thể viết ra giấy dưới dạng một câu ghi nhớ như “Đừng quên nộp 3/4 tiền lương cho vợ” để tình cờ có người đọc được cũng khó đoán ra.

12.3 Triển khai an ninh đăng nhập

Mỗi trương khoản trên hệ thống như một cửa ngõ ra vào máy. Điều mà mọi người cần là cái chìa khoá thích hợp: mật khẩu. Nếu đã truyền thụ được thói quen tốt bảo vệ mật khẩu, coi như quản trị viên đã có bước đầu khá tốt để triển khai an ninh hệ thống. Một khía cạnh luôn song hành với an ninh mật khẩu là an ninh trương khoản, hoặc an ninh đăng nhập.

An ninh đăng nhập bao gồm việc tìm tòi những trương khoản nào có tiềm năng phát hiện các vấn đề an ninh và sau đó xử lý chúng. An ninh đăng nhập đặt ra nhiều việc cho quản trị viên.

12.3.1 Trương khoản không có mật khẩu

Nhiều tin tặc đột nhập được vào hệ thống chỉ vì phát hiện ra một trương khoản user không có mật khẩu. Bạn nên đều đặn kiểm tra tệp mật khẩu nhằm phát hiện các trương khoản như thế và vô hiệu hoá chúng. Với Linux, mật khẩu được lưu tại trường thứ hai của tệp mật khẩu. Bạn có thể phát hiện một trường mật khẩu rỗng bằng nhiều công cụ, chẳng hạn như grep, awk, hoặc perl. Bạn vô hiệu hoá việc đăng nhập một trương khoản bằng cách chỉnh sửa tệp mật khẩu, đổi trường mật khẩu thành ký tự *. Kể từ

thời điểm này, chủ nhân trương khoản nói trên sẽ không thể vào hệ thống bằng mật khẩu đăng nhập cũ.

Xem "Thiết lập mật khẩu cho user"

12.3.2 Trương khoản không còn sử dụng

Bạn nên huỷ bỏ những tên đăng nhập nào không còn sử dụng. Nếu không ít ra bạn phải chỉnh sửa tệp mật khẩu và đổi trường mật khẩu thành ký tự * như vừa nói ở trên. Muốn huỷ trương khoản, bạn dùng lệnh **find** để dò tìm tất cả tệp thuộc sở hữu trương khoản ấy, sau đó chuyển quyền sở hữu hoặc huỷ bỏ chúng.

Xem "Gỡ bỏ một user", chương 10

Ghi chú: Nếu có sử dụng các tệp cấu hình khác, chẳng hạn như danh sách biệt danh của thư điện tử, bạn cũng phải gỡ bỏ trương khoản ra khỏi các tệp ấy.

12.3.3 Trương khoản mặc định

Thường có sẵn một vài trương khoản mặc định cho hệ thống. Thí dụ thoạt đầu khi vừa cài đặt Linux thì có trương khoản root nhưng chưa có mật khẩu (chỉ đúng với RedHat Linux phiên bản 5.x trở về trước, các phiên bản 6.x trở lên luôn đòi hỏi phải nhập mật khẩu ngay trong khi cài đặt). Do đó khi vừa cài đặt Linux xong, bạn nên kiểm tra tệp mật khẩu xem các trương khoản mặc định đã có mật khẩu hay chưa, nếu không thì vô hiệu hoá bằng ký tự sao.

Một số phần mềm cũng tự động tạo ra các trương khoản mặc định trong tiến trình cài đặt, vì vậy hãy nhớ đặt mật khẩu hoặc vô hiệu hoá những trương khoản đó.

12.3.4 Trương khoản công cộng

Một số cơ quan thường lập sẵn vài trương khoản công cộng cho những vị khách thỉnh thoảng đến đó sử dụng máy. Thường thì các trương khoản như thế không có mật khẩu, hoặc mật khẩu và tên đăng nhập giống nhau. Chẳng hạn như trương khoản mang tên guest hoặc là không có mật khẩu, hoặc mật khẩu cũng là guest. Tình trạng này là đầu mối của nhiều tai hoạ về mặt an ninh. Vì các trương khoản và mật khẩu kiểu này quá phổ biến nên tin tặc dùng chúng để vào hệ thống. Khi đã đột nhập xong, tin tặc sẽ tìm cách leo lên đến tận root, hoặc sẽ dùng hệ thống của bạn làm bàn đạp tấn công một hệ thống khác.

Khi truy ngược về để tìm hiểu cửa đột nhập của tin tặc mà lại gặp một trương khoản công cộng, bạn sẽ khó lần ra nguồn gốc thực sự của vụ tấn công.

Trương khoản công cộng, hoặc trương khoản mở là điều không nên có. Nếu thực sự cần thiết phải tạo ra một cửa ngõ như thế, bạn nên vô hiệu hoá trương khoản thường xuyên, đợi đến khi dùng mới kích hoạt. Quản trị viên có thể đặt cho trương khoản ấy một mật khẩu bất kỳ cần dùng và sau đó nên vô hiệu hoá mật khẩu. Và nhớ đừng gửi mật khẩu qua e-mail.

12.3.5 Trương khoản lệnh

Các hệ thống thường có nhiều trương khoản lệnh, nghĩa là loại trương khoản cho phép đăng nhập để ra một câu lệnh gì đó, rồi sau đó thoát khỏi ngay. Thí dụ finger là tên một trương khoản lệnh không mật khẩu. Khi có một user vào hệ thống với tên đăng nhập finger, chương trình finger được kích hoạt để báo cáo tên những ai đang ở trên hệ thống, sau đó liền kết thúc phiên làm việc. Một số trương khoản như thế (có thể là sync và date) cũng không có mật khẩu. Và mặc dù chỉ chạy một lệnh thay vì chạy nguyên một shell, chúng vẫn là kẽ hở về mặt an ninh.

Nếu buộc phải lập những trương khoản lệnh như thế trên hệ thống, bạn phải vô hiệu hoá việc nhận lệnh từ dòng lệnh, đồng thời phải làm sao khi lệnh vừa kết thúc và thoát khỏi thì đừng nhảy vào một shell nào đó có khả năng tương tác.

Thêm một lý do không nên dùng trương khoản lệnh là chúng cho thể cho người ở xa biết thông tin về hệ thống máy. Những chương trình như finger hoặc who lồng trong các trương khoản lệnh có thể mở cho người lạ truy cập tên đăng nhập đăng nhập của các user hệ thống của bạn.

12.3.6 Trương khoản nhóm

Trương khoản nhóm là một trương khoản dùng chung cho hơn một người, với cùng tên đăng nhập và cùng mật khẩu. Đây cũng là nguy cơ tiềm tàng. Nếu ai đó dùng trương khoản nhóm làm bàn đạp tấn công hệ thống khác, quản trị viên sẽ khó mà xác định thủ phạm.

Với Linux bạn có thể truy cập tệp theo nhóm. Như thế nếu tập thể cần truy cập một số tệp nào đó, họ có thể chia sẻ tệp thay vì chia sẻ trương khoản. Thay vì tạo ra trương khoản nhóm, một quản trị viên giỏi sẽ dùng Linux để xử lý các tệp chia sẻ với nhóm một cách an toàn hơn.

Xem "Làm việc với nhóm"

12.4 Vấn đề an ninh tệp

Hệ thống tệp Linux là một cấu trúc cây được xây dựng từ tệp và thư mục. Linux lưu trữ nhiều loại thông tin cho từng tệp trong hệ thống tệp, bao gồm những yếu tố sau:

- Tên tệp
- Loại tệp
- Kích cỡ tệp
- Vị trí vật lý của tệp trên đĩa
- Ngày giờ truy cập và chỉnh sửa
- Chủ sở hữu và định danh nhóm của tệp
- Quyền hạn truy cập gắn liền với tệp

Nếu thay đổi được thông tin tệp, user sẽ tạo ra kẻ hở an ninh.

12.4.1 Quyền hạn

Các quyền hạn (permision) về tệp của Linux quyết định xem user nào có quyền truy cập loại tệp và câu lệnh nào. Nói cách khác chúng cung cấp quyền truy cập tệp cho chủ sở hữu của tệp, cho các thành viên của nhóm có liên quan và cho các user khác.

Dùng lệnh **ls** –1 bạn sẽ nhận được danh sách các trường tham số quyền hạn.

Trường nằm phía ngoài cùng bên trái xác định các quyền hạn dùng cho loại tệp. Thí dụ các trường ấy có thể là -rw-r--r--. Dấu gạch nối (-) thứ nhất cho biết loại tệp. Những tệp bình thường có trường là -.

Chín ký tự tiếp theo cho biết các quyền hạn truy cập tệp cho người sở hữu, cho nhóm và cho những người khác. Mỗi loại quyền hạn được thể hiện bằng ba ký tự, bao gồm ký tự r (read, cho phép được đọc), w (write, cho phép được ghi) và x (execute, cho phép được thi hành).

Ký tự nào hiện diện thì quyền tương ứng được cấp. Nếu không được cấp, thì chỉ có dấu gạch nối.

Thí dụ một tệp có trường quyền hạn -rw-r--r--, thì có nghĩa là tệp bình thường (ký tự đầu tiên là -), chủ sở hữu tệp có quyền rw- (đọc và ghi nhưng không được thi hành), các thành viên khác của nhóm cùng với tất cả những người khác có quyền r-- (đọc, nhưng không được ghi và không được thi hành tệp). Lệnh chmod thay đổi được các quyền hạn của tệp.

Xem "Các quyền truy cập tệp"

Ghi chú: Bên cạnh các đối số rwx, lệnh chmod cũng chấp nhận giá trị bát phân. Bạn chỉ việc xử lý các ký tự trong một trường quyền hạn như là các bit trong một số bát phân. Nghĩa là nếu có ký tự hiện diện, thì bit tương đương là 1. Do đó -rw-r--r-- sẽ được viết là 644.

12.4.2 SUID và SGID

SUID được viết tắt từ cụm từ Set User ID (Thiết lập định danh user) và SGID viết tắt từ cụm từ Set Group ID (Thiết lập định danh nhóm).

Khi một chương trình được thực hiện, tiến trình tạo ra sẽ được gán 4 con số để chỉ rõ ai thực hiện nó: Đó là các giá trị ID thực và hiệu lực của chủ nhân hoặc nhóm của người ấy (real/effective UID or GID). Thông thường các giá trị thực và hiệu lực là như nhau. Giá trị thực của một tài khoản được gán khi nó được tạo ra và ghi lại trong tệp /etc/passwd. Còn giá trị hiệu lực được các hệ thống UNIX sử dụng để xác định quyền hạn truy xuất các tệp của một tiến trình.

Nếu giá trị hiệu lực UID của tiến trình trùng với UID của chủ sơ hữu tệp thì tiến trình sẽ có quyền truy cập tệp đó như chính chủ nhân, mặt khác, nếu giá trị hiệu lực của GID trùng với GID của tệp thì tiến trình sẽ có quyền của nhóm truy cập đến tệp. Trong các trường hợp còn lại thì tiến trình sẽ có quyền của others (những người khác).

Thông thường, khi bạn gọi thực thi một chương trình, các chỉ số thực và hiệu lực của UID và GID của tiến trình sẽ nhận giá trị UID và GID của trương khoản mà bạn đang đăng nhập. Như thế để có thể truy xuất một tệp nào đấy, trương khoản mà bạn đang

dùng phải có quyền truy cập tệp ấy. Tuy nhiên, nếu bạn ấn định SUID hay SGID cho chương trình được gọi thì tiến trình thực thi sẽ nhận UID và GID hiệu lực là UID và GID thực của chính chủ sở hữu hay nhóm của chủ sở hữu của chương trình.

Thí dụ: Nếu ta gọi thực thi một chương trình A để đọc nội dung của tệp B.txt, trong đó tệp A được phép cho tất cả mọi người thực hiện, mặc dù chủ sở hữu là chủ trương khoản C thuộc nhóm D:

```
-rwx--x--x 4 C D 1290 Apr 10 15:00:00 /usr/C/A
-rw------- 1 C D 22228 Apr 15:30:30 /ussr/C/B.txt
```

Trong trường hợp này, bạn sẽ không thể truy cập được B.txt nếu bạn không đăng nhập với trương khoản là C. Nhưng nếu xét lại cho tệp A thành:

```
-rws--x--x 4 C D 12900 Apr 10 15:00:00 /usr/C/A
-rw------- 1 C D 22228 Apr 10 15:30:30 /usr/C.B.txt
```

Tệp A đã được xét là một chương trình SUID (với bit s đã được bật). Khi đó dù bạn đăng nhập với trương khoản xyz bất kỳ, bạn vẫn có thể thực thi trình A để truy xuất tệp B.txt, vì khi đó, tiến trình tạo ra bởi A đã có UID hiệu lực là C chứ không phải là xyz (UID thực của tiến trình vẫn là xyz).

Như thế ta có thể thấy rằng mặc dù SUID và SGID có ích, song chúng lại là một kẽ hở tiềm tàng về an ninh. Thông thường chúng chỉ được dùng khi có chương trình nào đó cần quyền hạn đặc biệt để chạy, chẳng hạn như quyền của root. Những người viết chương trình đều tìm cách để cho SUID được an toàn nhất. Hầu hết các kẽ hở an ninh ở SUID xảy ra khi chương trình thực thi một dòng lệnh, khi kích hoạt một shell, hoặc khi chạy một tệp nào đó mà người sử dụng có thể thay đổi để chứa những lệnh của riêng mình.

Mặc dù một số chương trình SUID là rất cần thiết, song quản trị viên cần cố gắng hạn chế chúng đến mức tối thiểu. Bạn nên đều đặn rà quét hệ thống tệp bằng lệnh **find** để phát hiện các chương trình SUID mới.

12.5 Cảnh giác với tin tặc

Dù áp dụng mọi biện pháp an ninh trong một hệ thống, điểm yếu nhất vẫn là những user bởi vì họ đã có các trương khoản hợp lệ mà tin tặc có thể mượn đường chui vào hệ thống.

Nhìn chung thì các user tốt bụng có khuynh hướng muốn giúp đỡ người khác. Và giới tin tặc dùng kỹ năng đóng kịch, hoặc mượn danh để lợi dụng lòng tốt này.

Giả sử một trong các user của bạn là ông An, một user bình thường không giỏi giang lắm về công nghệ thông tin. Ngày kia ông An nhận được điện thoại và nội dung điện đàm đại loại như sau:

An: A-lô?

Người gọi: Chào ông An. Tôi là Bình thuộc bộ phận hỗ trợ kỹ thuật đây. Vì ổ cứng hiện hành đã hết chỗ, cho nên khoảng 7 giờ chiều nay chúng tôi sẽ dời home directory của một vài user sang ổ cứng khác. Trương khoản của ông nằm trong diện này do đó sẽ tạm thời ngưng hoạt động trong thời gian ngắn.

An: Thế à? Được, dù sao thì lúc ấy tôi cũng đã về nhà rồi.

Người gọi: Vâng. Nhớ đăng xuất trước khi về nhà ông nhé. Tôi chỉ cần kiểm tra qua loa mấy cái thôi ấy mà. Ông vui lòng nhắc lại ID đăng nhập xem, An có phải không ạ?

An: Vâng chính là An. Anh có bảo đảm là trong khi dời ổ cứng sẽ không làm hỏng tệp của tôi đấy chứ?

Người gọi: Chắc chắn là không, tôi chỉ kiểm tra lại trương khoản của ông cho chắc ăn. Thế mật khẩu là gì, tôi cần vào kiểm tra tệp xem có tốt không.

An: Mật khẩu là Thứ Sáu.

Người gọi: OK, được rồi, cảm ơn ông. Sau khi chuyển ổ cứng xong, tôi sẽ đích thân kiểm tra lại xem các tệp của ông còn nguyên vẹn không.

An: Cảm ơn anh. Chào anh.

Kết quả là người gọi đã lấy được ID và mật khẩu từ một trong các user của bạn. Nếu sáng mai ông An có gọi đến bộ phận hỗ trợ kỹ thuật thì chắc chắn sẽ không có ai có tên Bình ở đó cả. Là quản trị viên, bạn phải dự trù trước những tình huống tương tự để giáo dục các user của mình cảnh giác hơn. Nhớ nhắc đi nhắc lại với họ rằng đừng bao giờ tiết lộ mật khẩu qua điện thoại, qua e-mail hoặc hộp thư thoại.

12.6 Theo dõi việc sử dụng lệnh su

Linux dựa vào ID và mật khẩu đăng nhập để kiểm tra quyền ra vào hệ thống. Khi một user đăng nhập, hệ thống sẽ gán cho một số định danh UID. Chính UID này xác định quyền truy cập tệp và thư mục của user ấy.

Linux cho phép bạn chuyển sang một UID khác trong khi làm việc. Khi sử dụng lệnh su, user này có thể trở thành user khác, thậm chí thành root user. Tuy nhiên họ phải biết mật khẩu của user mà họ mượn định danh. Thí dụ có người muốn mượn ID của user tên ernie, người ấy cần ra lệnh như sau:

```
su ernie
```

Máy sẽ nhắc nhập mật khẩu liên quan đến tên đăng nhập ernie. Muốn chuyển sang root, phải ra lệnh sau:

```
su root
```

Máy sẽ nhắc nhập mật khẩu root.

Mọi dự định sử dụng tên su đều được ghi chép lại vào một tệp log hệ thống, chẳng hạn như tệp /var/adm/syslog. Quản trị viên thỉnh thoảng nên kiểm tra tệp này.

12.7 Triển khai một hệ thống an toàn

Quyền lực luôn đi đôi với trách nhiệm. Nếu không được quản lý tốt, thế mạnh của Linux như chia sẻ thông tin, tài nguyên xử lý cùng với các thiết bị ngoại vi sẽ bị người khác lợi dụng. Công việc của bạn là thiết lập hệ thống an ninh để chỉ những con người và hệ thống chọn lọc mới được kết nối với máy bạn và họ chỉ được phép sử dụng những vùng nào được cho phép trước.

12.7.1 Những mối đe doạ về an ninh

Bạn có thể dùng lệnh ps để kiểm tra xem những ai đang sử dụng hệ thống và hoạt động của họ là gì. Nên cảnh giác trường hợp một user nào đó sử dụng hệ thống trong thời gian kéo dài, hoặc dùng nhiều tài nguyên hơn thường lệ; biết đâu điều đó có nghĩa là một tên đăng nhập đã bị một user khác lợi dụng trái phép và chẳng hạn tin tặc đó đang chạy chương trình đoán mật khẩu.

12.7.2 Kiểm tra root

Trong Linux, tên đăng nhập root được dành riêng cho các quản trị viên. Người nào vào hệ thống với tư cách là root có quyền xoá bất kỳ tệp nào, hạn chế hoạt động của bất kỳ ai trên mạng và đương nhiên có khả năng gây nguy hại cho mạng.

Nhiều hệ điều hành thương phẩm đã cung cấp một số biện pháp cấm đoán hoặc phong toả, nhằm tránh tình trạng hỏng hóc tệp cũng như các yếu tố khác trên hệ thống. Trong khi đó các hệ UNIX và Linux lại có một cách nhìn khác đối với quản trị viên. UNIX và Linux cung cấp cho quản trị viên nhiều công cụ để kết nối với hầu hết mọi thiết bị tin học, các phần mềm để theo dõi năng suất của hệ thống, các chương trình đa dạng để thích nghi với mọi môi trường làm việc. Ngoài ra, quản trị viên có khả năng hạn chế hoạt động của user.

12.7.3 Modem và tin tặc

Nếu hệ thống của bạn cũng truy cập qua modem như những máy gia đình khác, thì đó sẽ là cửa ngõ mà tin tặc có thể đột nhập và thao túng hệ thống. Để đề phòng, có cơ quan đã thiết lập nhiều cấu trúc an ninh nghiêm ngặt đến nỗi rất khó làm việc với máy tính của họ. Một số cơ quan khác cài đặt tuỳ chọn "gọi lại" (callback), nghĩa là mỗi khi quay số modem của họ, bạn phải chờ modem đó gọi lại cho bạn rồi mới có thể tương tác với hệ thống đứng sau modem.

Đối với UNIX/Linux, biện pháp đề phòng được khuyên dùng là: tất cả các user đều phải có mật khẩu đăng nhập; hạn chế những hệ thống nào kết nối với máy gia đình; đóng quyền hạn đối với các tệp nhạy cảm; cảnh giác trước các chương trình cho phép user này chạy chương trình dưới ID của user khác.

12.7.4 Đề phòng các terminal rảnh rỗi

Đến cuối ngày làm việc, trước khi ra về các user phải đăng xuất đúng thủ tục hoặc chạy một chương trình tự động khoá terminal. Hầu hết các hệ thống UNIX đều có chương trình tự động đóng tắt terminal có hẹn giờ.

12.7.5 Các vấn đề liên quan đến con người

Tại những cơ quan liên quan đến quốc phòng hoặc có dữ liệu rất nhạy cảm, an ninh là một khái niệm phải được mọi nhân viên thuộc lòng và tuân thủ.

Song những nhân viên bình thường của một cơ quan công cộng sẽ khó mà hiểu tại sao người khác lại làm rùm beng về vấn đề an ninh như thế. Mọi nhân viên sẽ chưa nhận thức đầy đủ về an ninh cho đến khi một tệp quan trọng nào đó bị xoá mất, chẳng hạn như cơ sở dữ liệu về tài chính của cơ quan.

Nhân viên của cơ quan phải nhanh chóng nắm bắt sự nhạy cảm của các dữ liệu trên hệ thống. Đối với cơ quan, dữ liệu là khoản đầu tư quan trọng. Việc mất mát dữ liệu có khả năng làm rối loạn cơ quan. Những nhân viên nào không đủ thiện chí tham gia vào việc bảo đảm an ninh dữ liệu phải bị chuyển vị trí hoặc sa thải.

Nhiệm vụ của quản trị viên là làm sao cho các tệp và thư mục được bảo đảm đúng mức. Quản trị viên có nhiều công cụ trợ giúp, chẳng hạn như umask, **cron**, hoặc bản thân hệ điều hành Linux.

Thật ra công việc của quản trị viên cũng không dễ dàng. Biết rằng quyền hạn truy cập tệp càng rộng rãi thì càng hay bị đột nhập, cho nên hầu hết các quản trị viên lúc đầu thường có khuynh hướng hạn chế quyền hạn đó của user. Dần dà khi đã quen tay, nhiều quản trị viên lại nới lỏng phạm vi quyền hạn đến nỗi ai muốn làm gì trên hệ thống thì làm. Nếu muốn làm một quản trị viên giỏi thì bạn phải giữ đúng nguyên tắc.

12.7.6 Xử lý các khe hở về an ninh

Quản trị viên ít nhiều phải biết suy luận. Mời bạn xem qua thí dụ sau:

```
#who-u
pla ttymld        Jan 7 13:20       .secretary#1
bhh ttyp0         Jan 7 08:36       8:25 Warehouse
nva ttyøp         Jan 7 07:05       9:45 CEO Ofc
root ttyøp        Jan 7 08:36       .Modem#1
#date
Tue Jan 7 19:18:21 CST 2003
```

Giả sử bạn biết rằng user Phan Lan Anh (pla) đã rời khỏi văn phòng vào lúc 17 giờ. Vậy liệu ai đó đã bắt được mật khẩu của cô ta, hay lúc ra về cô ta vẫn chưa tắt terminal? Bằng lệnh who-u và date nói trên, bạn nhận thấy rằng Lan Anh đăng nhập hệ thống vào lúc 13:20 và bây giờ đã là 19:18 mà vẫn còn có người sử dụng hệ thống dưới tên Lan Anh.

Bạn sẽ hành động như thế nào khi có người đột nhập trái phép? Việc trước tiên là phải xác định xem có đúng như thế hay không. Nhiều khi chẳng qua chỉ là nhầm lẫn mà thôi. Nếu khẳng định được sự đột nhập, bạn có nhiều tuỳ chọn. Trước hết xác định hệ thống đã bị thiệt hại chưa và thiệt hại đến đâu. Nếu tóm được thủ phạm ta sẽ phải xử trí ra sao. Nếu có sự nghi ngờ, bạn cần bắt đầu thu thập và bảo vệ các chứng cứ.

Tiếp theo bạn phải tăng cường biện pháp an ninh hệ thống, song song với việc phục hồi dữ liệu từ các bản sao lưu. Có lẽ điều quan trọng nhất vào lúc này là ghi chép lại các thao tác của chính mình bằng một tệp ký sự (log). Tệp này sẽ có ích khi bạn thay đổi hoặc phục hồi dữ liệu. Bạn ghi ngày tháng và ký vào những biên bản chứng minh có sự đột nhập.

Hai biện pháp phòng chống khác là in ra giấy các tệp cấu hình hệ thống, chẳng hạn như /etc/fstab và thiết lập chính sách an ninh của cả mạng. Hãy báo cho các user biết về chính sách này và bảo đảm rằng họ đã thuộc lòng.

Một mối quan tâm khác là khi có nhân viên nghỉ việc hẳn. Quản trị viên phải xoá ngay tên đăng nhập của nhân viên này.

Với chừng ấy mối quan tâm về an ninh hệ thống, bạn cần tự hỏi mức độ nào là đủ. Liệu ta có nghiêm quá không?

Nói chung, cũng nên thử so sánh chi phí bỏ ra cho an ninh hệ thống với chi phí phục hồi sửa chữa khi xảy ra sự cố, nếu chi phí an ninh cao hơn thì bạn có thể tìm cách giảm bớt. Tuy nhiên cần nhớ rằng có những thứ mà tiền bạc không thể mua được. Bạn thử tính toán xem bao nhiêu thời gian và tiền bạc bỏ ra để thay thế một số tệp, cộng với tình trạng chờ máy, giảm năng suất lao động, chưa kể đến điều tiếng thị phi khi hệ thống tin học của cơ quan bạn bị hỏng.

12.7.7 Sao lưu

Trong số những vấn đề mà quản trị viên Linux phải đối diện, ít có vấn đề nào quan trọng bằng việc sao lưu hệ thống. Các dữ liệu quan trọng bị hỏng có thể làm cho quản trị viên mất việc hoặc thậm chí cả cơ quan náo loạn. Hầu hết các đĩa cứng đều có tuổi thọ trung bình khoảng 150.000 giờ (khoảng 20 năm). Tuy nhiên không có gì chắc chắn, bởi vì ổ đĩa và băng từ đều là thiết bị điện cơ, do đó chúng hoàn toàn có khả năng trục trặc.

Quản trị viên phải sao lưu thường xuyên hệ thống dữ liệu và đều đặn kiểm tra các vật chứa dữ liệu sao lưu.

12.8 Các mô-đun xác thực PAM

Dựa vào tên và mật khẩu của mình, từng user sở hữu quyền hạn riêng biệt, nói cách khác thì user phải có quyền thực hiện những điều mình muốn trong phạm vi cho phép của quản trị viên. Khi thao tác trên mạng, các user sẽ gây ảnh hưởng đến nội dung hệ thống ở nhiều cấp độ khác nhau. Do đó nói chung quản trị viên phải làm sao để mỗi user phải chạy các ứng dụng, tạo ra tệp, thay đổi và xoá các tệp mà không ảnh hưởng đến năng suất liên tục của cả hệ thống, cũng như không được truy cập những gì của người khác nếu không được phép.

Có những cách khác để kiểm tra tên và mật khẩu đăng nhập. Các mô-đun xác thực PAM (Pluggable Authentication Modules) giúp bạn thay đổi chính sách nhận dạng user trong khi vẫn giữ nguyên bản thân các ứng dụng.

Có bốn loại mô-đun PAM:

- Auth: thực hiện việc nhận dạng

- Account: xác định xem phần nhận dạng ấy có được chấp thuận không

- Pasword: thiết lập mật khẩu

- Session: cung cấp dịch vụ cho user sau khi mô-đun Account cho phép mô-đun Auth kiểm tra tên user.

Các mô đun có thể được xếp chồng (stack) thành từng bộ có thứ tự thực hiện khác nhau để tổ hợp ra nhiều cách kiểm tra hay hạn chế truy cập.

12.8.1 Tệp cấu hình PAM

Các tệp cấu hình PAM được đặt tại thư mục /etc/pam.d/. Cách tốt nhất để tìm hiểu cú pháp là xem xét tệp cấu hình.

Dưới đây là cấu hình tệp PAM cho passwd. Nếu đã cài đặt PAM là thành phần của Linux, thì đây là tệp mặc định /etc/pam.d/passwd:

```
#%PAM-1.0
auth required /lib/security/pam_pwdb.so shadow nullok
account required /lib/security/pam_pwdb.so
password required /lib/security/pam_cracklib.so retry=3
password required /lib/security/pam_pwdb.so use_authtok nullok
```

Trong tệp trên, dòng 1 là chú thích bởi ở đầu dòng có dấu thăng (#). Dòng 2 bắt máy nhắc user nhập mật khẩu và sau đó kiểm tra mật khẩu. Dòng 3 lặp lại nội dung trên nếu mật khẩu shadow không được sử dụng (sẽ bàn về mật khẩu shadow ở sau). Dòng 4 gọi một ứng dụng đoán tìm mật khẩu để thử xem mật khẩu có tốt không. Dòng 5 xác định xem phải dùng mô-đun nào để thay đổi mật khẩu.

Ghi chú: Ở những hệ Linux cũ, tệp /etc/pam.conf cung cấp các quy định về cấu hình. Hiện nay tuy tệp này vẫn được các hệ Linux mới chấp nhận, song không được khuyên dùng.

12.8.2 Thứ tự và mức độ cần thiết của các mô-đun PAM

Ở thí dụ trên, bạn nhận thấy tất cả bốn mô-đun đều được đánh dấu là "required" (ắt có). Đánh dấu "required" cho một mô-đun nghĩa là cho dù việc vượt qua các mô-đun trước đã thành công hay thất bại thì mô-đun được đánh dấu này vẫn sẽ được chương trình gọi ra "trình diện" để làm việc.

Vì mục tiêu an ninh, cả bốn mô-đun đều được gọi ra, do đó nếu không qua được cửa kiểm tra thì thông báo thất bại cho từng mô-đun sẽ giống y hệt nhau. Khi kẻ đột nhập không xác định được ngay là bị thất bại ở mô đun nào, hắn sẽ khó thực hiện bước tiếp theo.

Trong trường hợp tất cả các mô đun đều được gọi thì thứ tự trình diện không quan trọng. Tuy nhiên PAM vẫn cho phép người tùy ý lựa chọn flag (cờ hiệu) theo nhiều mức độ cần thiết khác nhau. Ngoài "required" còn có 3 mức nữa là:

- Optional (tuỳ ý)

- Sufficient (tạm đủ)

- Requisite (phải có)

Đánh dấu bằng flag "Optional" có nghĩa là cho dù việc vượt qua một mô-đun nào đó có thành công hay thất bại gì thì cũng chẳng ảnh hưởng đến khâu nhận dạng của tiến trình đăng nhập, NHƯNG với điều kiện là trong tệp cấu hình PAM còn có một mô-đun khác. Trong trường hợp tệp này chỉ có một mô đun duy nhất, thì thành công hay thất bại của mô-đun này sẽ quyết định kết quả đăng nhập. Mô-đun "Sufficient" hoạt động tương tự như mô-đun "Optional", nhưng kết quả của nó ưu tiên hơn kết quả của các mô-đun "Optional" khác.

Trong khi đó kết quả của các mô-đun đánh dấu "Required" hoặc "Requisite" sẽ lấn át quyền hạn của mô-đun "Sufficient". Do vậy, nếu kết quả kiểm tra của mô đun "Requisite" nào không đạt, quyền điều khiển sẽ được trả về cho ứng dụng.

Nếu muốn cả bộ stack PAM tạm ngừng tại một mô-đun nhất định nào đó, bạn cứ chỉnh sửa tệp cấu hình, thay đổi flag từ "required" thành "requisite".

Mời bạn xem thêm thông tin về PAM của RedHat Linux tại.

http://www.redhat.com/linux-infor/pam/

12.9 Lợi ích của mật khẩu shadow

Khi cài đặt Linux nếu bạn không cài Shadow Suite, tất cả thông tin của user (bao gồm cả mật khẩu) sẽ chỉ được lưu trong tệp /etc/passwd ở dạng đã mã hoá bằng lệnh crypt của UNIX. Phần ký tự văn bản được đặt ở chế độ rỗng [null] và mật khẩu là chìa khoá.

Cho dù hơi khó, song có thể giải mã thông tin đã mã hoá để tìm ra mật khẩu ban đầu. Nhiều user không chịu khó chọn mật khẩu phức tạp mà chỉ chọn những từ thông dụng; do đó giới tin tặc đã mã hoá tất cả các từ có trong bộ từ điển và dùng chương trình này để so sánh với danh sách mật khẩu trong tệp /etc/passwd. Mặc dù vẫn còn những hình thức bẻ khóa khác song đây vẫn là hình thức dễ dàng và đơn giản nhất.

Ngoài mật khẩu, tệp /etc/passwd còn chứa các thông tin khác như ID của user hoặc của nhóm, vốn sẵn sàng để cho các chương trình hệ thống đọc được.

Động tác shadow (che) mật khẩu sẽ chuyển mật khẩu sang một tệp khác, thường là /etc/shadow, mà chỉ có root mới đọc được. Kẻ đột nhập nào tấn công bằng danh sách kiểu từ điển sẽ gặp khó khăn với các mật khẩu đã được mã hoá trong tệp /ect/shadow.

Hầu hết các bản phát hành tiêu chuẩn Linux đều có Shadow Suite.

Tuy nhiên bạn không nên cài đặt Shadow Suite trong một số trường hợp sau:

- Hệ thống không có trương khoản của user.

- Hệ thống chạy trên mạng cục bộ LAN và sử dụng NIS (dịch vụ thông tin mạng) để gọi hoặc cung cấp username và mật khẩu cho các máy khác trên mạng.

- Hệ thống được các terminal server sử dụng để kiểm tra qua cổng NFS để gọi hoặc cung cấp username và mật khẩu cho các máy khác trên mạng.

- Hệ thống được các terminal server sử dụng để kiểm tra qua các cổng NFS, NIS, hoặc một phương pháp nào khác.

- Hệ thống chạy một phần mềm khác để nhận dạng user. Phần mềm này không có phiên bản shadow nào, ngoài ra bạn cũng không có mã nguồn.

12.9.1 Tệp /etc/passwd và /etc/shadow

Tệp không được che có dạng như sau:

```
username:passwd:UID:GID:full_name:directory:shell
```

Thí dụ:

```
username:Npje04eh3mx8e:507:200:FullName:/home/usename:/bin/csh
```

Ký tự x ở trường thứ hai là một yếu tố đứng giữ chỗ cho mật khẩu thật, vốn được lưu trong tệp /etc/shadow và tệp này có dạng như sau:

```
username:passwd:last:may:must:warn:expire:disable:reserved
```

Bảng 12.1 Các trường trong một mục ghi của tệp /etc/shadow

Trường	**Mô tả**
username	Tên dùng để đăng nhập
password	Mật khẩu được mã hoá
last	Số ngày kể từ 1-1-1970 đến khi mật khẩu thay đổi lần gần đây nhất
may	Số ngày còn lại trước khi mật khẩu có thể được thay đổi
must	Số ngày mà sau đó mật khẩu buộc phải thay đổi
warn	Số ngày còn lại trước khi mật khẩu hết hạn mà user được thông báo
expire	Số ngày sau khi mật khẩu hết hạn và trương khoản bị vô hiệu hoá
disable	Số ngày kể từ 1-1-1970 đến khi trương khoản bị vô hiệu hoá
reserved	Trường để dành

12.9.2 Thêm, đổi và gỡ bỏ user có mật khẩu shadow

Phần mềm Shadow Suite có những lệnh như sau để thêm vào, sửa đổi và gỡ bỏ user: **useradd**, **usermod** và **userdel**.

12.9.2.1 useradd

Như tên gọi, lệnh này để thêm user vào hệ thống. Ngoài ra bạn có thể gọi lệnh này để thay đổi thiết lập mặc định của máy. Đây cũng là việc đầu tiên bạn nên làm để chỉnh sửa mọi thứ cho phù hợp với hệ thống. Bạn ra lệnh như sau:

```
useradd -D
```

12.9.2.2 usermod

Tiện ích này thay đổi các thông tin của một user và rất giống lệnh **useradd**.

12.9.2.3 userdel

Giúp bạn bỏ bớt một user bằng lệnh sau:

```
userdel –r tên_user
```

Tham số -r xoá tất cả các tệp trong home directory của user bị gỡ bỏ, sau đó sẽ xoá luôn home directory. Còn một cách khác để ngăn cản không cho user nào đó vào hệ thống nữa, là dùng lệnh passwd để khoá trương khoản của user ấy.

12.9.2.4 passwd

Ngoài việc tạo ra và thay đổi mật khẩu, root user có thể dùng lệnh này để thực hành một số việc khác như:

- Mở và khoá trương khoản (với tuỳ chọn –u và –l)

- Quy định số ngày tối đa mà một mật khẩu nào đó còn hợp lệ (-x)

- Quy định số ngày tối thiểu giữa hai lần thay đổi mật khẩu (-n)

- Quy định số ngày báo trước thời hạn còn lại trước khi mật khẩu hết hạn (-w)

- Quy định rằng sau khi mật khẩu hết hạn, số ngày sẽ là bao nhiêu trước khi trương khoản bị khoá (-i).

12.9.2.5 pwch

Lệnh này giúp bạn kiểm tra tính nhất quán của các tệp /etc/passwd và /etc/shadow, bằng cách xem xét lại username và bảo đảm rằng mỗi mục ghi phải đủ các yếu tố: số trường phải đúng, username là độc nhất, cũng như sự hợp lệ của home directory, nhóm sơ cấp, yếu tố nhận dạng user và nhóm.

Và pwck sẽ cảnh báo nếu có trương khoản không khai mật khẩu.

12.9.2.6 grpck

Lệnh này dùng để kiểm tra tính nhất quán của các tệp /etc/group và /etc/gshadow. Chương trình sẽ xem xét số lượng đúng của các trường, tính độc nhất của user name, sự hợp lệ của danh sách thành viên và quản trị viên.

Cũng như đối với lệnh pwck, tuỳ chọn –r sẽ tạo ra bản báo cáo, vì vậy bạn có thể dùng **cron** để phát động lệnh kiểm tra tự động.

12.9.2.7 Kiểm tra mật khẩu qua điện thoại

Chương trình xét mật khẩu qua điện thoại (dial-up) giúp bạn quy định những ai có quyền truy cập vào hệ thống bằng đường này. Bạn dò trong tệp /ect/login.defs xem thư mục ghi DIALUPS_CHECKS_ENAB đã ở chế độ Yes hay chưa.

Có hai tệp chứa thông tin về dial-up:

- /etc/dialups chứa các ttys (mỗi dòng có một mục như thế, nhưng phía trước không có ghi /dev/). Nếu thấy có liệt kê tty, tức là việc kiểm tra dial-up đã được thực hiện.

- /etc/d_passwd chứa trọng đường dẫn của một shell, theo sau đó là mật khẩu nhiệm ý.

Nếu một user đăng nhập bằng đường điện thoại có liệt kê trong /etc/dialups, đồng thời shell của user này cũng được liệt kê trong /etc/d_paswd, thì user đó sẽ có quyền vào hệ thống sau khi khai đúng mật khẩu dial-up.

Lệnh passwd chỉ định mật khẩu cho các shell trong tệp /etc/d_passwd.

Chương 13. Thiết lập cấu hình kernel

Chương này giúp bạn lập cấu hình và cài đặt một kernel mới. Kernel là cốt lõi cung cấp các dịch vụ hệ thống cơ bản nhất cho những phần còn lại của hệ điều hành. Bạn hãy nhớ rằng mỗi phần mềm đều có thể chứa các lỗi hoặc khe hở an ninh. Tình trạng này vẫn xảy ra đối với các hệ điều hành "miễn phí" cũng như các hệ điều hành thương phẩm. Song điểm thuận tiện của Linux là mã nguồn luôn sẵn có, cho nên bạn có thể tự xử lý bất kỳ vấn đề nào khi vừa phát hiện ra, nếu không muốn chờ những người khác làm điều đó và công bố trên Internet.

Chương này bao gồm những chủ đề chính sau đây:

-Chuẩn bị một kernel mới

-Lập cấu hình cho kernel mới

-Biên dịch kernel mới

-Xây dựng một kernel mô-đun hoá

Linux còn thuận tiện ở chỗ các phiên bản kernel của nó kể từ 2.x đều có thể nạp các trình điều khiển thiết bị và các công cụ khác vào hệ thống mà không cần phải biên dịch lại phần mềm hỗ trợ để tránh tạo ra một kernel lớn hơn. Điều này cho phép nạp vào bộ nhớ những thành phần tương ứng của kernel chỉ khi nào hệ thống cần dùng đến chúng. Do đó sẽ có mục giới thiệu cách chỉnh sửa kernel nhằm giải quyết một vấn đề, hoặc để bổ sung một tính năng mà không phải tốn công biên dịch lại toàn bộ hệ thống.

13.1 Chuẩn bị một kernel mới

Để giải quyết vấn đề nào đó của hệ điều hành, đôi khi chúng ta chỉ có một giải pháp duy nhất là thay kernel. Mặc dù không dành cho người mới vào nghề, song việc tải kernel mới từ Internet về, rồi xây dựng cấu hình và cài đặt nó là điều không thể tránh. Nếu có ít nhiều kinh nghiệm lập trình và hiểu biết về ngôn ngữ lập trình C, bạn hoàn toàn có khả năng thành công. Nếu không, bạn có thể bỏ qua mục này.

Lý do cần một kernel mới có thể là:

-Một trình điều khiển thiết bị hoặc bổ sung hỗ trợ phần cứng vừa ra mắt.

-Bạn muốn bỏ bớt một số tính năng không cần thiết trong kernel, để hạ thấp nhu cầu về bộ nhớ cho hệ thống.

Việc đầu tiên phải làm là xác định hệ thống đang dùng kernel phiên bản thứ mấy, bằng cách gõ lệnh:

```
uname -a
```

Kết quả sẽ được hiển thị dưới dạng:

Số phiên bản chính	Số phiên bản phụ	Cấp bổ sung

Theo giấy phép GPL, bất cứ ai cũng có thể chỉnh sửa Linux, song Torvalds là nguồn chính thức của các kernel mới. Từ đó cộng đồng những người sử dụng và phát triển Linux có cơ sở chung để làm việc và liên lạc với nhau.

Ghi chú: Bạn nên đọc phần HOWTO liên quan đến kernel để có thông tin mới nhất trước khi bắt tay vào xây dựng một kernel mới. Ngoài ra bạn cũng nên sao lưu bản kernel hiện hành để khởi động máy trong trường hợp có rắc rối với kernel mới.

13.2 Lập cấu hình một kernel mới

Để xây dựng một kernel mới, việc đầu tiên phải làm là lập cấu hình các tệp mã nguồn thí dụ nằm trong thư mục /usr/src/linux-2.4.18-3. Bạn cũng cần có bộ biên dịch C sẵn trong máy. Nếu trước đây trong tiến trình cài đặt bạn chưa cài bộ này, giờ bạn dùng RPM để cài bằng các lệnh thí dụ như sau:

```
rpm -i kernel -source-2.4.18-3.i86.rpm\rpm -i gcc-2.96.i86.rpm
```

Có thể bạn cũng cần cài đặt kernel header cũng với một số thư viện biên dịch.

Để có được các tệp nguồn kernel và bổ sung mới, bạn đến một số website như sunsite.unc.edu (còn nếu bạn chỉ chỉnh sửa kernel hiện hành thì không cần). Thông thường các tệp nguồn được nén thành dạng **tar,** sau khi tải về bạn phải bung chúng ra.

Lưu ý: Hai lệnh trong thí dụ sau đây sẽ giúp bạn sao lưu kernel hiện hành, bằng cách chép trọn thư mục nguồn Linux và thư mục khác mang tên linux.sav:

```
cd /usr/src
cp /usr/src/linux-2.4.18-3 linux.sav -r
```

Tiếp theo bạn dùng lệnh patch để chạy các tệp bổ sung. Sau khi chuẩn bị xong các tệp nguồn, bạn có thể lập cấu hình và xây dựng hệ thống mới. Có ba giao diện để lập cấu hình kernel, tuỳ thuộc vào sở thích của bạn: chương trình chạy ở chế độ thuần văn bản; chương trình chạy ở chế độ thực đơn văn bản; và nếu bạn có X Window thì chương trình có thể chạy ở chế độ đồ họa.

Lưu ý: Muốn sử dụng các mô-đun kernel, bạn phải trả lời Yes cho phần hỗ trợ phiên bản mô-đun và kernel (CONFIG_MODVERSIONS) trong khi lập cấu hình kernel.

13.2.1. Chương trình tương tác nền văn bản

Nếu đang sử dụng chương trình chạy ở chế độ thuần văn bản, bạn gõ lệnh sau đây từ thư mục thí dụ /usr/src/linux-2.4.18-3:

```
# make config
```

Lệnh **make** sẽ hỏi bạn vài câu về các driver (trình điều khiển thiết bị ngoại vi) mà bạn muốn cài đặt và lập cấu hình. Cách thông thường nhất là cứ bấm <Enter> để chấp nhận các giá trị mặc định, hoặc bạn tự trả lời theo sở thích. Một vài câu hỏi như thế được liệt kê ở bảng sau. Tuỳ theo phiên bản kernel hoặc phần bổ sung đang xử lý, bạn có thể sẽ trả lời nhiều câu hỏi khác. Danh sách các tuỳ chọn này được chấp nhận bởi tất cả các tiện ích cấu hình đã mô tả trong chương này.

Bảng 13.1: Một vài tuỳ chọn cấu hình

Tuỳ chọn cấu hình	Mô tả
Code Maturity Level	Để sử dụng với các thành tố thử nghiệm trong kernel này.
Loadable Module Support	Sẽ cần thiết trong trường hợp bạn dùng dạng mô-đun rời thay vì kernel nguyên khối.
General Setup	Đặt một loạt các câu hỏi về những thành phần tổng quát, chẳng hạn như bộ xử lý toán học và hỗ trợ PCI BIOS.
Floppy, IDE, and Other Block Devices	Đặt câu hỏi về loại ổ cứng IDE và các thiết bị vào/ra dữ liệu theo kiểu mảng (block I/O).
Networking Options	Đặt nhiều câu hỏi về cách tương thích hoá các tính năng hỗ trợ mạng, chẳng hạn như bức tường lửa và che giấu IP.
SCSI Support	Kích hoạt hỗ trợ bộ điều khiển SCSI.
SCSI Low-Level Support	Kích hoạt hỗ trợ cấp thấp cho bộ điều khiển SCSI và báo cáo các trạng thái SCSI khác nhau.
Network device support	Kích hoạt hỗ trợ các tiến trình mạng và bộ điều khiển mạng.
ISDN subsystem	Kích hoạt hỗ trợ kernel đối với mạng ISDN.
CD-ROM drivers	Hỗ trợ các ổ CD-ROM (không phải các ổ SCSI hoặc IDE/ATAPI).
Filesystems	Cho phép lập cấu hình hỗ trợ các hệ thống tệp khác nhau.
Character Devices	Hỗ trợ các thiết bị vào/ra dữ liệu theo kiểu ký tự.
Sound	Hỗ trợ cấu hình cho nhiều loại bìa âm thanh khác nhau.
Kernel Hacking	Hỗ trợ can thiệp sâu trong kernel.

13.2.2. Chạy chương trình thực đơn văn bản

Nếu đang dùng chương trình chạy ở chế độ thuần văn bản, bạn gõ lệnh sau đây từ thư mục thí dụ /usr/src/linux-2.4.18-3 để chuyển sang chế độ thực đơn văn bản:

```
# make menuconfig
```

Sử dụng chế độ chạy ở chế độ thực đơn văn bản thuận tiện ở chỗ bạn chỉ cần lập cấu hình cho những phần kernel mà bạn muốn chỉnh sửa. Chương trình tương tác trên nền **menu** sẽ hướng dẫn bạn đi suốt tiến trình lập cấu hình.

13.2.3. Chạy chương trình trên nền X Window

Nếu đang sử dụng chương trình chạy ở chế độ thuần văn bản, bạn gõ lệnh sau đây từ thư mục thí dụ /usr/src/linux-2.4.18-3 để chuyển sang chế độ đồ họa:

```
# make xconfig
```

Công cụ chạy trên nền giao diện đồ họa giúp bạn lập cấu hình những thành phần kernel muốn chỉnh sửa và dễ dùng hơn so với các công cụ chạy trên nền văn bản. Mỗi khi bấm chuột vào một nút, một hộp thoại sẽ hiện ra hướng dẫn bạn lập cấu hình. Thí

dụ hộp thoại Module Support sẽ giúp bạn lập cấu hình về hỗ trợ mô-đun cho toàn bộ kernel. Để chọn một tiết mục nào đó, bạn chỉ việc bấm chuột vào nút tương ứng (có hình thoi). Khi cần được trợ giúp ở một chủ đề nhất định thì bạn bấm nút Help ở phía bên phải hộp thoại.

Sau khi trả lời các câu hỏi xong, bạn phải ghi lại cấu hình vừa chọn và thoát khỏi bằng cách bấm nút “Save and Exit”.

13.3 Biên dịch một kernel mới

Sau khi lập xong cấu hình, bạn phải biên dịch (compile) kernel mới. Các lệnh sau đây sẽ giúp bạn làm việc đó:

```
make dep
make clean
make
```

Tuỳ vào phần cứng của bạn, tiến trình biên dịch có thể kéo dài từ vài phút đến vài giờ. Sau đó sẽ đến đoạn thiết lập hệ thống để thích hợp với kernel mới khi khởi động. Kernel thí dụ mang tên zImage trong thư mục /usr/src/linux-2.4.18-3/arch/i386/boot và bạn phải sao chép hình ảnh tệp này vào thư mục khởi động. Tuy nhiên trước đó bạn phải sao lưu hình ảnh của kernel hiện hành để đề phòng trường hợp trục trặc. Bạn gõ lệnh sau đây để sao lưu kernel cũ:

```
mv /boot/vmlinuz /bootvmlinuz.old
```

Tiếp theo bạn đưa kernel mới vào bằng lệnh sau:

```
cp /usrs/src/linux-2.4.18-3/arch/i386/boot/zImage /boot/vmlinuz
```

Xem "Lập cấu hình LILO"

Để thay đổi thành phần kernel mà Linux sẽ dùng khi khởi động, bạn cần hiệu chỉnh tệp etc/**lilo**.conf và thêm vào mục ghi khác cho kernel mới. Khi thực hiện bước này, bạn phải đổi tên /boot/vmlinuz thành /boot/vmlinuz.old bằng các lệnh vừa nêu trên, rồi sau đó đổi nhãn thành "old" như trong tệp **lilo**.conf thí dụ sau đây:

*Một mẫu tệp /etc/**lilo**.conf*

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz
label=linux
initrd=/boot/initrd
root=/dev/hdal
read-only
image=/boot/vmlinuz.old
label=old
```

```
root=/dev/hdal
read-only
```

Sau khi thay đổi /etc/**lilo**.conf xong, bạn chạy lệnh sau đây để bản **lilo** vừa cập nhật được đưa vào phần khởi động:

```
/sbin/lilo -v
```

Từ thời điểm này trở về sau mỗi lần khởi động, máy sẽ lấy kernel mới làm mặc định và đương nhiên sẽ tạm chờ 5 giây (vì timeout=50) để bạn quyết định xem cần phải khởi động hệ điều hành nào.

13.4 Xây dựng kernel mô-đun hoá

Kể từ khi các phiên bản đời 2.0.x của lõi Linux xuất hiện với tính năng mô-đun hoá, những người sử dụng Linux có thể thay đổi kernel cho hợp với sở thích riêng biệt. Trước đây khi muốn truy cập một loại thiết bị mới hoặc một thành phần mới của hệ thống tệp, bạn phải biên dịch thêm phần mềm hỗ trợ tương ứng (driver) vào kernel; như vậy đối với cấu hình phần cứng, kích cỡ của kernel mới có thể phình to lên đến ngưỡng cuối cùng. Do đó nếu đòi hỏi hỗ trợ cho một loại thiết bị nào đó lâu lâu mới dùng một lần thì có nghĩa là ta đã sử dụng tài nguyên hệ thống không có hiệu quả. Với khả năng mới của kernel 2.4.x, nếu có thiết bị mới hoặc một thành phần mới của hệ thống tệp thỉnh thoảng mới dùng đến, thì chỉ khi gọi bạn mới nạp driver. Muốn xem các mô-đun hiện hành trong bộ nhớ, bạn gõ lệnh:

```
lsmod
```

Kết quả của lệnh trên là một màn hình cho bạn biết những mô-đun nào đang có trong bộ nhớ, chúng được nạp như thế nào và chiếm bao nhiêu bộ nhớ:

```
Module      Size        Used by
autofs      11520       0     (autoclean) (unused)
eepro100    17664       1
usb-uhci    21536       0     (unused)
usbcore     51712       1     [usb-uhci]
ext3        64624       1
jbd         40992       1     [ext3]
```

Chỉ có các bản Red Hat Linux chạy chip Intel hoặc SPARC là hỗ trợ cho các kernel dạng mô-đun. Những người sử dụng Red Hat Linux chạy chip Alpha phải xây dựng kernel nguyên khối như đã mô tả ở mục trước (Biên dịch một kernel mới).

Để xây dựng các mô-đun, bạn đến thư mục /sur/src/linux-2.4.18-3 và chạy lệnh:

```
make modules
```

Rồi chạy lệnh sau đây để cài đặt các mô-đun:

```
make modules-install
```

13.4.1. Làm việc với các mô-đun kernel

Giờ đây khi đã biên dịch và cài đặt xong các mô-đun, bạn có thể mở rộng kernel bằng các mô-đun có thể nạp được. Bảng sau trình bày những lệnh liên quan.

Bảng 13.2: Các lệnh mô-đun sẵn có trong Linux

Lệnh	**Mô tả**
lsmod	Liệt kê các mô-đun hiện hành trong kernel.
insmod	Chèn một mô-đun nhất định vào kernel.
rmmod	Gỡ bỏ mô-đun ra khỏi kernel.
depmod	Tạo ra một tệp liệt kê các phụ thuộc để dùng với **modprobe**.
modprobe	Nạp mô-đun từ danh sách do **depmod** tạo ra.

Nếu đang chạy X Window, bạn có thể tận dụng daemon kernel từ Control Panel để làm việc với mô-đun bằng GUI thay vì bằng dòng lệnh. Bạn bấm chuột vào nút như hướng dẫn ở minh hoạ, hộp thoại Kernel Configurator hiện ra như ở minh hoạ.

Muốn liệt kê các mô-đun đã nạp, bạn gõ lệnh **lsmod**. Muốn thêm vào kerneld một mô-đun đã biên dịch xong, bạn có thể dùng một trong hai cách, hoặc gõ lệnh:

```
insmod tên_mô-đun
```

Hoặc bấm nút Add của hộp thoại kerneld và xác định mô-đun lựa chọn.

Muốn xoá một mô-đun khỏi kernel, bạn gõ lệnh:

```
rmmod tên_mô-đun
```

hoặc chọn tên mô-đun từ danh sách liệt kê trên màn hình, sau đó bấm nút Remove.

13.4.2 Khởi động lại kernel

Những thay đổi mà bạn đã thực hiện bằng công cụ Kernel Daemon Configuration sẽ được thể hiện ở tệp /etc/modules.conf. Lệnh kerneld sẽ đọc những thay đổi này mỗi khi được kích hoạt. Sau đây là một tệp /etc/modules.conf thí dụ:

Một mẫu tệp /etc/modules.conf

```
alias scsi_hostadapter BusLogic
alias eth0 ne2k-pci
```

Muốn khởi động lại kerneld, bạn xem trên thanh công cụ và bấm nút Restart Kerneld. Hoặc bạn dừng và khởi động lại daemon bằng các dòng lệnh như thí dụ sau:

```
/etc/rc.d/init.d/kernel stop
/etc/rc.d/init.d/kernel start
```

Khi khởi động lại kerneld, máy sẽ không nạp lại các mô-đun hiện hành, nhưng kerneld sẽ dùng cấu hình này khi cần nạp mô-đun trong tương lai.

Chương 14. Quản lý hệ thống tệp

Hệ thống tệp bao gồm mọi dữ liệu sinh ra trên Linux. Các chương trình Linux, thư viện, tệp hệ thống, tệp user đều nằm trong hệ thống tệp, do vậy việc quản lý nó một cách đúng đắn là rất quan trọng.

Chương này nói đến nhiều công đoạn quản lý khác nhau và phần lớn chúng đều được tự động thực hiện khi bạn cài đặt Linux. Tuy nhiên bạn vẫn phải tìm hiểu cách quản lý các hệ thống tệp để biết cách tạo ra, thao tác và duy trì hệ thống Linux của mình. Việc nắm vững các tác vụ này mang tính quyết định nếu bạn muốn trở thành quản trị viên hệ thống. Sau đây là các mục chính:

-Tìm hiểu hệ thống tệp

-Mount và unmount hệ thống tệp

-Hệ thống tệp mạng NFS

-Duy trì hệ thống tệp

-Sử dụng lệnh fsck

-Tạo và định dạng hệ thống tệp

-Sử dụng tệp swap và phân vùng

14.1 Tìm hiểu hệ thống tệp

Trong Linux, những tệp mà người sử dụng nhìn thấy đều theo cấu trúc cây thư mục, với root nằm ở trên cùng. Từ điểm này các thư mục và tệp mới mọc nhánh ra và mở rộng dần xuống phía dưới. Thư mục cao nhất, ký hiệu bằng vạch /, được gọi là root directory (thư mục gốc).

Với người sử dụng bình thường thì cây thư mục có vẻ là một dải những tệp và thư mục nối liền nhau. Thực ra, những thư mục này có thể nằm ở nhiều vị trí vật lý khác nhau, trên các phân vùng khác nhau và thậm chí trên các ổ đĩa khác nhau. Khi một trong các phân vùng ấy được kết nối với cấu trúc cây tại một thư mục gọi là điểm lắp ghép (**mount point**), thì điểm này và mọi thư mục cấp dưới được gọi là hệ thống tệp.

Hệ điều hành Linux hình thành từ nhiều thư mục và tệp khác nhau. Các thư mục có thể lập thành nhiều hệ thống tệp khác nhau, tuỳ vào cách cài đặt đã chọn. Nhìn chung, phần lớn hệ điều hành nằm ở hai hệ thống tệp: hệ thống tệp root (hệ thống tệp gốc) được ký hiệu là / và một hệ thống tệp khác được kết nối theo /usr (đọc là user).

Khi dùng lệnh cd / để chuyển về thư mục gốc và gọi hiển thị danh sách thư mục, bạn sẽ thấy nhiều thư mục. Những thư mục này tạo thành nội dung của hệ thống tệp root, đồng thời cung cấp điểm lắp ghép cho các hệ thống tệp khác.

Thư mục /bin chứa các chương trình khả thi, còn gọi là binaries (tệp nhị phân). Chúng là những chương trình chủ yếu của hệ thống. Nhiều lệnh của Linux, chẳng hạn như **ls,** được thi hành bởi những chương trình nằm tại thư mục ấy.

Thư mục /sbin chứa các tệp nhị phân của chủ hệ thống. Hầu hết các tệp ở đây dùng để quản trị hệ thống (superuser-bin).

Thư mục /etc rất quan trọng vì chứa nhiều tệp cấu hình Linux. Chúng giúp cho hệ thống của bạn có "cá tính". Tệp mật khẩu passwd nằm ở thư mục này, cũng như fstab (danh sách các hệ thống tệp cần nạp vào khi khởi động).

Ngoài ra thư mục /etc còn chứa các script khởi động cho Linux, danh sách các máy mạng (host) kèm địa chỉ IP, cùng với nhiều thông tin cấu hình khác.

Các thư viện dùng chung được chứa trong thư mục /lib. Khi dùng chung thư viện, nhiều chương trình sẽ sử dụng lại các đoạn mã giống nhau; hơn nữa khi được chứa cùng chỗ, thư viện sẽ giúp giảm thiểu kích cỡ chương trình và thời gian chạy.

Thư mục /dev chứa các tệp đặc biệt gọi là device files (tệp thiết bị), được hệ thống sử dụng để điều khiển phần cứng. Thí dụ tệp /dev/mouse là thông tin từ chuột. Khi tổ chức sử dụng phần cứng theo cách này, Linux làm cho việc tương tác với chúng cũng giống như với phần mềm. Điều này có nghĩa là trong nhiều trường hợp, để thao tác phần cứng bạn có thể dùng cùng một cú pháp như với phần mềm. Chẳng hạn để tạo ra một bản tồn trữ home directory của mình trên đĩa mềm, bạn dùng lệnh:

```
tar -cdf /dev/fd0 lan_anht
```

/dev/fd0 chỉ cho lệnh **tar** biết phải dùng đĩa mềm. Nhiều thiết bị trong thư mục /dev được tập hợp thành các nhóm logic. Bảng sau liệt kê vài thiết bị phổ biến nhất trong thư mục /dev.

Bảng 14.1: Các thiết bị thường dùng có trong thư mục /dev

Tệp thiết bị	**Mô tả**
/dev/console	Màn hình kết nối vật lý với hệ thống Linux
/dev/hd*	Các ổ cứng IDE. Thiết bị /dev/hda1 chỉ phân vùng đầu tiên trên ổ cứng hda. Thiết bị /dev/hda chỉ toàn bộ ổ cứng hda
/dev/sd*	Các ổ đĩa SCSI. Những ổ đĩa và phân vùng SCSI có cùng quy ước với các thiết bị IDE /dev/hd*
/dev/fd*	Các ổ đĩa mềm. Ổ đầu tiên là /dev/fd0, ổ thứ hai là /dev/fd1
/dev/st*	Các ổ băng từ có giao diện SCSI
/dev/tty*	Nhiều loại thiết bị dùng để nhập liệu. Viết tắt là tty vì kế thừa các hệ UNIX trước kia dùng các terminal dạng teletype. Với Linux, những tệp này hỗ trợ các màn hình giao tiếp ảo, có thể truy cập bằng cách bấm từ <Alt-F1> cho đến <Alt-F6> và cho phép nhiều user đăng nhập cùng lúc
/dev/pty*	Terminal giả (pseudo), dùng cho việc đăng nhập từ xa, chẳng hạn như những phiên làm việc bằng telnet
/dev/ttyS	Các cổng nối tiếp trên PC. /dev/ttyS0 tương ứng COM1 của DOS. Nếu sử dụng chuột nối tiếp thì /dev/mouse là một liên kết tượng trưng nối với thiết bị ttyS tương ứng

/dev/cua*	Các thiết bị đặc biệt gọi ra ngoài hệ, để dùng với modem
/dev/null	Thiết bị rất đặc biệt, hoạt động như một lỗ đen. Mọi dữ liệu ghi vào /dev/null coi như bị mất vĩnh viễn. Việc này hữu ích khi bạn muốn chạy một câu lệnh và thủ tiêu stdout hoặc stderr. Và nếu /dev/null dùng làm tệp nhập, bạn sẽ tạo ra một tệp có độ dài zero.

Thư mục /proc là một hệ thống tệp ảo, dùng để đọc thông tin từ bộ nhớ.

Thư mục /tmp chứa các tệp tạm thời mà chương trình tạo ra trong khi chạy. Nếu bạn biết hệ thống của mình có chương trình tạo ra nhiều tệp tạm với kích cỡ lớn, bạn nên tạo thư mục /tmp thành một hệ thống tệp riêng thay vì đặt nó vào hệ thống tệp gốc như là một thư mục bình thường. Bởi vì với đà chất chứa các tệp tạm ngày càng nhiều, hệ thống tệp gốc sẽ nhanh chóng bị đầy.

Thư mục /home là thư mục cơ sở của các home directory cho các user. Quản trị viên thường đặt .home thành một hệ thống tệp riêng rẽ nhằm tạo nhiều khoảng trống cho user sử dụng. Ngoài ra nếu hệ thống có đông đảo user, bạn nên chia thư mục /home thành nhiều hệ thống tệp khác nhau. Thí dụ bạn có thể tạo ra /home/staff cho các thành viên của nhóm điều hành và /home/admin cho quản trị viên. Mỗi thư mục như thế sẽ là một hệ thống tệp riêng, bên dưới có home directory riêng cho các user tương ứng.

Thư mục /var lưu các tệp có thể thay đổi kích thước theo thời gian. Nhiều tệp đăng nhập hệ thống (system log file) thường nằm trong thư mục này. Thư mục /var/spool cùng với các thư mục con dùng để chứa dữ liệu tạm như tin tức hoặc thư tín mới nhận được, hoặc đang chờ gửi đi nơi khác.

Ghi chú: Bạn có thể tạo ra một số điểm lắp ghép bên dưới thư mục / theo ý thích. Thí dụ tạo điểm lắp ghép /cdrom nếu bạn thường ghép CD-ROM vào hệ thống.

Thư mục /usr và các thư mục con rất quan trọng cho hệ thống Linux, bởi vì chúng chứa nhiều chương trình cần thiết cho user. Những thư mục cấp dưới của /usr chứa các gói phần mềm mà bạn đã cài đặt. Trong hầu hết mọi trường hợp, thư mục /usr được thiết lập như là một hệ thống tệp riêng rẽ.

Bảng 14.2: Các thư mục thứ cấp quan trọng trong hệ thống tệp /usr

Thư mục thứ cấp	**Mô tả**
/usr/bin	Lưu nhiều tệp khả thi của hệ thống
/usr/etc	Lưu nhiều tệp cấu hình hệ thống
/usr/include	Tại đây và trong nhiều thư mục cấp dưới của /usr/include có lưu mọi tệp kèm theo bộ biên dịch C. Những tệp header này xác định các hằng và hàm cần cho việc lập trình bằng C
/usr/g++-include	Lưu các tệp kèm theo bộ biên dịch C++
/usr/lib	Chứa các thư viện để chương trình sử dụng trong khi kết nối
/usr/share/**man**	Chứa các trang chú giải lệnh và các chương trình. Bên dưới /usr/share/**man** là nhiều thư mục tương ứng với các chương trong chú giải

/usr/src	Chứa các thư mục mã nguồn của nhiều chương trình trên hệ thống. Nếu kiếm được một gói phần mềm thú vị, bạn nên lưu vào /usr/src/tên_gói trước khi cài đặt
/usr/local	Dành cho việc thiết kế hoặc tuỳ chỉnh các ứng dụng cho phù hợp với hệ thống của bạn. Hầu hết phần mềm dùng tại chỗ được lưu trong các thư mục cấp dưới của /usr/local. Hình thức và kích cỡ của nó thay đổi theo từng hệ thống UNIX

14.2 Mount và unmount hệ thống tệp

Đến đây, bạn đã có khái niệm khá rõ ràng về hệ thống tệp. Câu hỏi tiếp theo là bạn làm như thế nào để thiết lập một thư mục thành một hệ thống tệp riêng biệt.

Muốn Mount (lắp) một hệ thống tệp vào cây thư mục Linux, bạn phải có một phân vùng vật lý, một CD-ROM hoặc đĩa mềm. Một điều kiện nữa: thư mục mà bạn muốn ghép hệ thống tệp vào (tức điểm lắp ghép) phải là thư mục có thật.

Mount một hệ thống tệp không có nghĩa là tạo ra thư mục điểm lắp ghép. Điểm lắp ghép phải có từ trước khi bạn **mount** hệ thống tệp. Giả sử bạn muốn **mount** CD-ROM tại /dev/sr0 theo điểm lắp ghép /mnt. Thư mục mang tên /mnt phải sẵn có, nếu không thao tác **mount** sẽ thất bại. Sau khi **mount** hệ thống tệp vào thư mục này, mọi tệp và thư mục con của hệ thống tệp đều xuất hiện bên dưới thư mục /mnt. Nếu không, /mnt sẽ là thư mục rỗng (có thể tạo thêm vài thư mục con bên trong /mnt).

Ghi chú: Muốn biết thư mục hiện hành đang ở hệ thống tệp nào, bạn gõ lệnh df. Máy sẽ hiển thị hệ thống tệp và khoảng trống còn lại trên đĩa.

14.2.1 Mount hệ thống tệp có tính tương tác

Để **mount** hệ thống tệp, bạn gõ lệnh **mount** theo cú pháp sau:

```
mount thiết_bị mount_point
```

thiết_bị là thiết bị vật lý mà bạn muốn **mount**; mount_point là vị trí trong hệ thống tệp, nơi mà bạn muốn đặt thiết bị.

Ghi chú: Chỉ superuser mới có quyền ra lệnh mount. Đây là lý do an ninh hệ thống. Hiện đã có nhiều phần mềm cho phép user **mount** một số hệ thống tệp nhất định nào đó, đặc biệt là đĩa mềm.

Lệnh **mount** chấp nhận nhiều tùy chọn (xem Bảng 14.3). Khi chờ nhận một đối số nào đó mà không nhận được, **mount** sẽ cố gắng đoán ra đối số ấy từ tệp /etc/fstab.

Bảng 14.3: Các tùy chọn cho lệnh mount

Tùy chọn	**Mô tả**
-f	Làm cho mọi thứ diễn ra "như thật", song chỉ là thao tác giả
-v	Cung cấp thêm thông tin về những gì mà **mount** sẽ thực hiện
-w	Mount hệ thống tệp với các quyền hạn đọc và ghi
-r	Mount hệ thống tệp với quyền hạn chỉ đọc mà thôi

-n	Mount hệ thống tệp nhưng không ghi mục nào vào tệp /etc/mtab
-t loại	Xác định loại hệ tệp được **mount**. Những loại hợp lệ là minix, ext, ext2, xiafs, msdos, hpfs, proc, nfs, umsdos, sysv và iso9660 (mặc định)
-a	Cố gắng **mount** mọi hệ thống tệp đã khai báo trong /etc/fstab
-o danh-sách-các-tuỳ-chọn	Khi phía sau lệnh **mount** có một loạt các tuỳ chọn được cách nhau bằng dấu phẩy, **mount** sẽ áp dụng các tuỳ chọn ấy vào hệ thống tệp đang được **mount**. Muốn có danh sách đầy đủ các tuỳ chọn bạn hãy xem trang **man** của lệnh **mount.**

Lưu ý: Nhiều dạng của lệnh **mount** rất mạnh. Thí dụ lệnh **mount** /dev/hdb3 /mnt sẽ gắn hệ thống tệp trên phân vùng /dev/hdb3 vào thư mục /mnt.

Cũng giống như thế, **mount** -r -t iso9660 /dev/sr0 /mnt gắn hệ thống tệp trên ổ CD-ROM SCSI /dev/sr0 vào thư mục /mnt. Ổ CD-ROM này có thuộc tính là chỉ đọc và dạng tệp là ISO 9660. Và lệnh **mount** -vat nfs sẽ thiết lập tất cả hệ thống tệp NFS có liệt kê trong tệp /etc/fstab.

Ghi chú: Nếu thực hiện **mount** một hệ thống tệp nào đó không được, bạn dùng lệnh **mount** -vf thiết_bị mount_point để xem thông báo. Lệnh sẽ hiển thị một danh sách chi tiết và bảo **mount** làm tất cả mọi thứ chỉ trừ việc thiết lập hệ thống tệp. Như thế, bạn làm thao tác giả với lệnh **mount,** sau đó nhận được nhiều thông tin về những gì **mount** sẽ thực hiện.

14.2.2 Mount hệ thống tệp khi khởi động

Một khi đã làm việc ổn định, thường thì Linux sử dụng một số hệ thống tệp hay dùng và ít khi thay đổi. Do đó bạn có thể xác định danh sách các hệ thống tệp nào Linux cần phải **mount** khi khởi động và cần phải **unmount** (tháo) khi đóng tắt. Các hệ thống tệp này được liệt kê trong một tệp cấu hình đặc biệt gọi là /etc/fstab, viết tắt từ chữ filesystems table.

Tệp /etc/fstab liệt kê các hệ thống tệp cần được **mount,** mỗi dòng có tên một hệ tệp. Những trường ở mỗi dòng được phân cách bằng ký tự trống hoặc dấu tab.

Bảng 14.4: Các trường trong tệp /etc/fstab

Trường	**Mô tả**
Filesystem specifier	Xác định một thiết bị kiểu mảng (block) đặc biệt hoặc hệ thống tệp ở xa cần được mount
Mount point	Xác định điểm lắp ghép cho hệ thống tệp. Đối với các hệ thống tệp đặc biệt như tệp swap, bạn dùng chữ none, có tác dụng làm cho tệp swap hoạt động nhưng nhìn vào cây tệp thì không thấy
Type	Thông báo loại hệ thống tệp. Những loại hệ thống tệp sau đây được chấp nhận: minix, một hệ thống tệp tại chỗ, hỗ trợ tên tệp dài từ 14 đến 30 ký tự.

	ext, hệ thống tệp tại chỗ có tên tệp dài hơn và inode lớn hơn (hệ thống tệp này được hệ thống tệp ext2 thay thế và sau này sẽ không còn sử dụng) ext2, hệ thống tệp tại chỗ với tên tệp dài hơn, inode lớn hơn, cùng với những đặc điểm ưu việt khác xiafs, một hệ thống tệp tại chỗ msdos, hệ thống tệp tại chỗ cho các phân vùng DOS hpfs, hệ thống tệp tại chỗ cho các phân vùng OS/2 iso9660, hệ thống tệp tại chỗ cho CD-ROM nfs, hệ thống tệp tại chỗ để **mount** các phân vùng mạng swap, một phân vùng hoặc tệp đặc biệt để swap umsdos, hệ thống tệp dạng UMSDOS sysv, hệ thống tệp dạng UNIX System V ext3, hệ thống tệp mới của Red Hat
Mount options	Danh sách các tuỳ chọn để **mount** hệ thống tệp được ngăn cách bằng dấu phẩy. Danh sách này ít nhất phải có Type cho hệ thống tệp. Xem trang **man** của lệnh **mount** để biết thêm chi tiết.
Dump frequency	Xác định khoảng thời gian để lệnh dump sao lưu hệ thống tệp. Nếu trường này trống, dump sẽ giả định rằng hệ thống tệp không cần sao lưu.
Pass number	Khai báo cho lệnh **fsck** biết thứ tự kiểm tra các hệ thống tệp khi khởi động hệ thống. Hệ thống tệp gốc phải có giá trị 1. Mọi hệ thống tệp khác phải mang giá trị 2. Nếu không khai báo, khi khởi động, máy sẽ không kiểm tra tính nhất quán của hệ thống tệp.

Ghi chú: Muốn **mount** lúc khởi động máy, bạn nên sử dụng tệp /etc/fstab thay vì dùng lệnh **mount,** bởi vì chỉ các superviser mới được dùng lệnh **mount**.

Một mẫu tệp fstab

LABEL=/	/	ext3	defaults	1 1
LABEL=/boot	/boot	ext3	defaults	1 2
none	/dev/pts	devpts	gid=5 mode=620	0 0
/dev/hdc2	/mhome	vfat	defaults	0 0
none	/proc	proc	defaults	0 0
none	/dev/shm	tmpfs	defaults	0 0
/dev/hda3	swap	swap	defaults	0 0

/dev/cdrom1	/mnt/cdrom	iso9660	noauto.owner.kudzu.ro	0 0
/dev/fd0	/mnt/floppy	auto	noauto.owner.kudzu	0 0

Trong tệp fstab bạn có thể thấy nhiều loại hệ thống tệp, nhưng cần lưu ý dấu thăng đứng trước một ghi chú hoặc vô hiệu hóa một dòng trong tệp. Trong tệp fstab mẫu có hai hệ thống tệp Linux bình thường được **mount,** đó là hai phân vùng /dev/hda1 và /dev/hda2. Chúng được xem là loại ext3 và được **mount** dưới thư mục root / và /usr.

Mục ghi defaults phía dưới trường options cho biết hệ thống tệp tương ứng phải được **mount** cùng với một bộ tuỳ chọn mặc định, nghĩa là các tuỳ chọn đọc/viết được; được coi như một thiết bị block đặc biệt; mọi tệp I/O phải được thực hiện đồng bộ; cho phép thực hiện các tệp nhị phân; có thể **mount** hệ thống tệp bằng lệnh **mount** -a; hệ thống tệp này sẽ diễn giải các bit UID (định danh user) và GID (định danh nhóm) được lập trên các tệp; và user bình thường không được phép **mount** hệ thống tệp này. Như bạn đã nhận thấy, chỉ cần gõ vào chữ defaults là bạn đỡ phải khai báo hàng loạt các tuỳ chọn.

/proc là một hệ thống tệp ảo trỏ về chỗ xử lý thông tin trong bộ nhớ. Bạn nhận thấy nó không có phân vùng vật lý tương ứng nào để **mount** cả.

Phân vùng /dev/hda3 là một phân vùng swap dùng làm khoảng swap dạng bộ nhớ ảo cho kernel. Điểm lắp ghép của nó được khai là none để bạn không phải thấy nó xuất hiện trong cây hệ thống tệp. Nhưng phân vùng này vẫn phải nằm trong tệp /etc/fstab để hệ thống biết vị trí vật lý của nó. Các phân vùng swap cũng được **mount** bằng tuỳ chọn swap (xem trang **man** của lệnh **mount**).

Ghi chú: Mời bạn xem trang **man** của fstab để nắm thông tin chi tiết về mọi tuỳ chọn cho tệp /etc/fstab.

Các hệ thống tệp DOS có thể được **mount** một cách tự động. /dev/hda1 là phân vùng đầu tiên trên ổ cứng SCSI hda. Phân vùng này được tự động **mount** như là một phân vùng DOS khi xác định loại hệ thống tệp là msdos và điểm lắp ghép là /dosc. Bạn có thể đặt điểm lắp ghép cho hệ thống tệp DOS bất cứ nơi nào, không nhất thiết phải dưới thư mục root.

14.2.3 Unmount một hệ thống tệp

Sau khi đã biết mọi thứ về việc **mount** (lắp), bây giờ bạn làm quen với thao tác **unmount** (tháo). Bạn cần **unmount** một hệ thống tệp vì nhiều lý do: Để kiểm tra hoặc sửa chữa hệ thống tệp với lệnh fsck; khi gặp vấn đề về mạng, phải **unmount** các hệ thống tệp được **mount** trước đó bằng NFS; hoặc **unmount** một hệ thống tệp ở đĩa mềm.

Lệnh **unmount** có ba dạng cơ bản:

```
unmount thiết_bị /mount_point
unmount -a
unmount -t loại_fs
```

Với thiết_bị là tên của thiết bị vật lý cần được **unmount**; mount_**point** là tên thư mục mountpoint (chỉ xác định một thứ thôi).

Lệnh **unmount** chỉ có hai thông số dòng lệnh: -a để **unmount** mọi hệ thống tệp và -t loại_fs, để xử lý những hệ thống tệp nào đã được chọn.

Ghi chú: Lệnh **unmount** không **unmount** được hệ thống tệp đang sử dụng. Thí dụ nếu bạn có hệ thống tệp **mount** dưới /mnt và bạn gõ lệnh:

```
cd /mnt
unmount /mnt
```

thì máy sẽ báo lỗi rằng hệ thống tệp đang bận. Muốn **unmount** hệ thống tệp dưới /mnt, bạn phải chuyển sang thư mục khác và vào hệ thống tệp khác.

14.3 Hệ thống tệp mạng NFS

Hệ thống tệp mạng (NFS, Network Files System) là một cách giúp bạn **mount** hệ thống tệp từ một máy tính khác qua mạng TCP/IP.

Với NFS các hệ thống khác nhau như PC, Macintosh, UNIX và Linux có thể chia sẻ dữ liệu lẫn nhau. Với NFS, một hệ thống tệp nằm trên máy vi tính từ xa cũng được **mount** tại chỗ và thoạt trông cũng giống như một hệ thống tệp tại chỗ cho các user tại chỗ.

NFS có nhiều lợi ích. Thí dụ trên mạng của bạn hiện có một PC giữ vai trò máy chủ với dung lượng ổ cứng rất lớn. Ổ cứng máy này chứa mọi home directory của mọi user. Nếu **mount** những ổ cứng ấy qua NFS cho tất cả máy tính khác của hệ thống, các user có thể truy cập home directory của họ từ bất kỳ máy nào trong mạng.

NFS gồm ba thành phần chính:

-Những hệ thống tệp nào muốn **mount** bằng NFS phải nằm trên máy có khả năng liên lạc với nhau qua mạng TCP/IP.

-Bản thân chiếc máy có hệ thống tệp mà bạn định coi như là hệ thống tệp tại chỗ, thì hệ thống tệp ấy phải được mount. Chiếc máy này sẽ giữ vai trò máy chủ (server) và tiến trình làm cho hệ thống tệp của server có thể sử dụng từ xa được gọi là xuất khẩu hệ thống tệp.

-Máy nào muốn **mount** bộ hệ thống tệp được xuất khẩu gọi là máy khách (client). Máy khách phải **mount** hệ thống tệp dưới dạng tệp NFS qua tệp /etc/fstab lúc khởi động, hoặc dưới dạng tương tác qua lệnh mount.

14.3.1 Xuất khẩu hệ thống tệp NFS

Để cho máy khách **mount** hệ thống tệp NFS, bản thân hệ thống tệp này phải được server tạo điều kiện truy cập được, mà ta gọi là khả dụng (available). Trước khi khả dụng, hệ thống tệp này phải được **mount** trên server. Nếu luôn dùng hệ thống tệp dạng NFS xuất khẩu, bạn nhớ liệt kê hệ thống tệp này trong tệp /etc/fstab của server để được tự động **mount** mỗi khi server khởi động.

Một khi đã **mount** hệ thống tệp tại chỗ xong rồi, bạn có thể làm cho nó khả dụng qua NFS. Đây là tiến trình gồm hai bước.

A-Đầu tiên bạn phải bảo đảm server chạy các daemon NFS như rpc.mountd và rpc.nfsd. Các daemon này thường khởi hành từ script mang tên /etc/rc.d/init.d/nfs ở phần startup. Để đơn giản hoá, chỉ cần kiểm tra rằng script của bạn có hai dòng sau đây:

```
daemon rpc.mountd
daemon rpc.nfsd
```

Ghi chú: Là những chương trình trên nền RPC, hai daemon rpc.mountd và rpc.nfsd không chịu sự quản lý của daemon inetd, nhưng được kích hoạt ngay lúc khởi động Linux vì đã đăng ký chạy với portmap daemon. Do đó sau khi rpc.portmap đã khởi hành bạn mới chạy hai daemon kia.

B-Thứ hai bạn phải đưa hệ thống tệp NFS vào tệp cấu hình /etc/exports. Tệp này cho biết hệ thống tệp xuất khẩu được, máy nào có quyền truy cập chúng, với mức độ và loại truy cập nào.

14.3.2 Tệp /etc/exports

Tệp /etc/exports được hai daemon mountd và nfsd sử dụng để xác định xem những hệ thống tệp nào sẽ được xuất khẩu và với những giới hạn nào. Tệp này là một danh sách với mỗi dòng gồm các tham số về một hệ thống tệp. Mỗi dòng gồm có: tên của mountpoint cho một hệ thống tệp tại chỗ, tên các máy được phép **mount** hệ thống tệp ấy, tiếp sau có thể là liệt kê các tuỳ chọn **mount** nằm trong ngoặc đơn. Bảng sau trình bày các tuỳ chọn **mount** sẵn có trong tệp /etc/exports.

*Bảng 14.5: Các tuỳ chọn **mount** sẵn có trong tệp /etc/exports*

Tuỳ chọn	**Mô tả**
insecure	Cho phép truy cập từ máy này mà không cần xác thực
secure	Đòi hỏi phải xác thực RPC từ máy này
root_squash	Ánh xạ yêu cầu từ root, UID 0: client, NOBODY_UID: server
no_root_squash	Không ánh xạ yêu cầu từ UID 0 (mặc định)
ro	Mount hệ thống tệp với tính năng chỉ đọc (mặc định)
rw	Mount hệ thống tệp với tính năng đọc và ghi.
link_relative	Chuyển đổi các liên kết tượng trưng tuyệt đối (đường liên kết bắt đầu bằng dấu sổ ngược) thành liên kết tương đối, bằng cách đặt phía trước liên kết một số lượng cần thiết các ký tự ../ tính từ thư mục chứa liên kết này đến thư mục root trên server
link_absolute	Trử về nguyên thuỷ các liên kết tượng trưng tuyệt đối. Đây là mặc định cho server Sun NFS và Linux.
map_daemon	Ánh xạ các tên tại chỗ, tên từ xa và các số định danh ID, bằng cách dùng daemon mang tên lname/uid map với client, nơi mà yêu cầu NFS phát sinh. Dùng để ánh xạ giữa các khoảng trống UID client và server.

all_squash	Ánh xạ tất cả UID và GID với user vô danh. Tuỳ chọn này có ích đối với các thư mục chung xuất khẩu qua NFS, chẳng hạn như những thư mục chứa FTP và news
no_all_squash	Ngược lại với tuỳ chọn all_squash và là mặc định cho Linux
squash_uids	Xác định danh sách các UID để ánh xạ vô dạnh. Một liệt kê các ID có hiệu lực sẽ có dạng sau: squash uids=0-15. 20, 25, -50
squash_gids	Xác định danh sách các GID để ánh xạ vô danh. Một liệt kê các ID có hiệu lực sẽ có dạng sau: squash gids=0-15. 20, 25, -50
anonuid	Lập UID cho trương khoản vô danh. Tuỳ chọn này có ích đối với các client PC/NFS
anongid	Lập GID cho trương khoản vô danh. Tuỳ chọn này có ích đối với các client PC/NFS
noaccess	Dùng để loại trừ vài thư mục trước một client nào đó. Làm cho client không thể truy cập kể từ thư mục này trở xuống

Một mẫu tệp /etc/exports

```
/home A.ifi.edu(rw) B.ifi.edu(rw) C.ifi.edu(rw)
/usr/local/bin *.ifi.edu(ro)
/projects develop.ifi.edu(rw) A.ifi.edu(ro)
/pub/ (ro. insecure.root_squash)
```

Trong thí dụ trên, server xuất khẩu bốn hệ thống tệp khác nhau. /home được **mount** với tính năng đọc và ghi các cho máy A, B và C. Điều này cũng chứng tỏ rằng thư mục đang lưu home directory của các user, căn cứ theo tên của các home directory.

Còn /usr/local/bin được xuất khẩu với tính năng chỉ đọc cho mọi máy thuộc tên miền ifi.edu.

/projects lại được xuất khẩu với tính năng đọc và ghi cho máy develop.ifi.edu, nhưng A.ifi.edu chỉ đọc mà không được ghi. /pub không có danh sách những máy được phép truy cập, có nghĩa là máy nào cũng được quyền **mount** hệ thống tệp này.

/pub được xuất khẩu với tính năng chỉ đọc, không cần nhận dạng khi truy cập và server sẽ tái ánh xạ bất kỳ yêu cầu nào từ root, nếu yêu cầu phát xuất từ một máy ở xa xin truy cập hệ thống tệp này.

14.3.3 Mount các hệ thống tệp NFS

Mount một hệ thống tệp NFS cũng giống như **mount** các hệ thống tệp khác. Bạn có thể **mount** hệ thống tệp NFS từ /etc/fstab lúc khởi động hoặc tương tác qua lệnh mount.

Ghi chú: Khi sử dụng lệnh **mount** hoặc khi ghi một mục vào tệp /etc/fstab, bạn phải phân cách tên của host và các phần của tệp hoặc hệ thống hoặc đường dẫn trong tên hệ thống tệp từ xa bằng dấu hai chấm, thí dụ:

```
mailserver:/var/spool/mail
```

Nếu không có dấu hai chấm phân cách, hệ thống của bạn sẽ không **mount** được thư mục từ xa đúng cách.

14.3.4 Mount NFS qua /etc/fstab

Khi xác định một hệ thống tệp NFS trong tệp /etc/fstab, bạn nhận dạng hệ thống tệp ấy qua dạng thức:

```
tên_host:/tệp/hệ_thống/đường_dẫn
```

với tên_host là tên của server đang lưu hệ thống tệp đó và:

/tệp/hệ_thống/đường_dẫn chính là hệ thống tệp trên server.

Loại hệ thống tệp được xác định là nfs trong trường tuỳ chọn **mount** của mục ghi hệ thống tệp.

*Bảng 14.6: Các tuỳ chọn **mount** phổ dụng nhất với NFS*

Tuỳ chọn	**Mô tả**
rsize=n	Xác định kích thước datagram bằng byte, khi có client NFS yêu cầu đọc và yêu cầu này được thoả mãn. Mặc định là 1024 byte
wsize=n	Xác định kích thước datagram bằng byte, khi có client NFS yêu cầu ghi và yêu cầu này được thoả mãn. Mặc định là 1024 byte
timeo=n	Lập thời lượng mà client NFS phải chờ sau khi yêu cầu. Được tính bằng đơn vị phần mười của giây và mặc định là 0.7 giây
hard	Mount hệ thống tệp bằng cách lắp cứng (mặc định)
soft	Mount hệ thống tệp bằng cách lắp mềm (xem mục sau)
intr	Ngắt một yêu cầu NFS để dừng khi server NFS không hồi đáp

14.3.5 Mount mềm và mount cứng

Mount mềm và **mount** cứng xác định ứng xử của client NFS khi server NFS ngưng không hồi đáp hay đáp ứng yêu cầu.

Mặc định thì các hệ thống tệp NFS được **mount** cứng. Dù cho **mount** cứng hay mềm, mỗi khi server không hồi đáp, client sẽ chờ cho hết thời gian timeout (xác định bằng giá trị của tuỳ chọn timeo), sau đó sẽ phát đi yêu cầu lần nữa (gọi là timeout phụ).

Nếu yêu cầu gửi đến server tiếp tục bị timeout và tổng thời gian timeout lên đến 60 giây thì sẽ xảy ra timeout tổng.

Nếu hệ thống tệp được **mount** cứng, client phát thông điệp đến console và bắt đầu gửi yêu cầu **mount** lần nữa và lần này thì cho giá trị timeout lớn gấp đôi lần trước.

Client có thể lặp đi lặp lại thao tác này. Client sẽ cố gắng **mount** đi **mount** lại hệ thống tệp NFS từ server cho đến khi nào hài lòng thì thôi.

Trong khi đó, nếu xảy ra timeout tổng, **mount** mềm chỉ phát đi một báo lỗi I/O đến tiến trình yêu cầu và Linux vẫn tiếp tục hoạt động.

Thường thì các gói phần mềm và tiện ích quan trọng **mount** qua NFS phải được **mount** cứng. Do đó **mount** cứng trở thành mặc định. Chẳng hạn bạn sẽ không muốn hệ thống của mình chạy phập phù một khi Ethernet tạm ngưng kết nối trong thời gian ngắn, hoặc bạn muốn Linux tiếp tục chờ và vận hành trong khi đang sao lưu mạng.

Mặt khác, bạn có thể **mount** mềm những dữ liệu không quan trọng, chẳng hạn như các phân vùng tin tức từ xa, để nếu xảy ra trường hợp máy ở xa có sự cố thì phiên đăng nhập hiện hành của bạn không bị treo.

Một mục ghi hệ thống tệp NFS điển hình trong tệp /etc/fstab có thể là như sau:

```
mailserver:/var/spool/mail /var/spool/mail nfs
timeo =20, intr
```

Mục ghi này **mount** hệ thống tệp mang tên /var/spool/mail đang nằm trong mailserver ở mountpoint tại chỗ /var/spool/mail. Mục ghi còn cho biết loại hệ thống tệp là nfs, đặt timeout bằng 20 x 1/10= 2 giây và quy định rằng hệ thống tệp có thể ngắt được.

14.3.6 Mount NFS kiểu tương tác

Cũng như bất kỳ hệ thống tệp nào khác, hệ thống tệp NFS có thể **mount** theo chế độ tương tác. Song bạn cần nhớ rằng lệnh **mount** theo kiểu này với NFS không hấp dẫn tí nào vì có rất nhiều tuỳ chọn từ dòng lệnh.

Lấy thí dụ kể trên, lệnh **mount** tương tác để **mount** hệ thống tệp /var/spool/mail được viết lại thành:

```
#mount -t nfs -o timeo=20, intr mailserver:/var/spool/mail /var/spool/mail
```

Nếu phải xác định kích thước datagram và timeout, các lệnh **mount** tương tác khá phức tạp. Tốt hơn bạn nên đặt các lệnh **mount** loại này trong tệp /etc/fstab để máy tự động nạp chúng khi khởi động.

14.4 Bảo trì hệ thống tệp

14.4.1 Nguyên tắc

Quản trị viên hệ thống chịu trách nhiệm duy trì tính nhất quán của các hệ thống tệp. Công việc thường làm là thỉnh thoảng kiểm tra xem có tệp nào hỏng không.

Linux sẽ tự động kiểm tra hệ thống tệp lúc khởi động nếu chúng có giá trị lớn hơn 0 và được xác định trong trường pass number của tệp /etc/fstab.

Ghi chú: Hệ thống tệp mang tên ext2 có một flag gọi là clean bit (bit sạch). Nếu hệ thống tệp được đồng bộ hoá và **unmount** một cách đúng đắn, thì bit này sẽ được đánh dấu vào hệ thống tệp. Khi Linux khởi động và thấy đánh dấu thì sẽ không kiểm tra tính nhất quán của hệ thống tệp ấy nữa.

14.4.2 Sử dụng lệnh fsck

Thỉnh thoảng quản trị viên nên kiểm tra hệ thống tệp xem có tệp nào bị hỏng không. Với các bản phát hành Linux, bạn dùng lệnh **fsck** (filesystem check). Lệnh này thực sự

là mặt trước của một loạt các lệnh khác đứng phía sau nhằm kiểm tra hệ thống tệp. Cú pháp của lệnh **fsck** với rất nhiều tùy chọn (xem bảng dưới) là như sau:

```
fsck [-A] [-V] [-t loại_fs][-a][-l][-r][-s] filesys
```

Trong đó filesys là đối số xác định hệ thống tệp nào cần kiểm tra. Đối số này có thể là tên của một thiết bị block đặc biệt, chẳng hạn /dev/hda1, hoặc của một mountpoint như /usr.

Tuy nhiên hình thức cơ bản nhất của lệnh **fsck** là:

```
fsck filesys
```

Bảng 14.7: Các tùy chọn của lệnh fsck

Tùy chọn	**Mô tả**
-A	Duyệt khắp tệp /etc/fstab và cố gắng kiểm tra mọi hệ thống tệp chỉ trong một lần duyệt. Tuỳ chọn này thường được dùng trong tiến trình khởi động Linux để kiểm tra mọi hệ thống tệp được **mount** bình thường. Nếu đã chọn -A thì không được dùng đối số filesys nữa
-V	Chế độ thông báo chi tiết, cho biết lệnh **fsck** đang làm gì
-t loại_fs	Xác định loại hệ thống tệp cần kiểm tra
-a	Tự động sửa chữa những hỏng hóc trong hệ thống tệp mà không cần hỏi. Hãy cẩn thận khi sử dụng tuỳ chọn này
-l	Liệt kê mọi tên tệp trong hệ thống tệp
-r	Hỏi trước khi sửa chữa hệ thống tệp
-s	Liệt kê các superblock trước khi kiểm tra hệ thống tệp

Lệnh **fsck** thực ra dùng để gọi các lệnh khác kiểm tra hệ thống tệp nào khớp với loại hệ mà bạn muốn kiểm tra. Để làm được điều này đương nhiên Linux cần biết loại hệ thống tệp sẽ phải kiểm tra. Cách dễ nhất để bảo đảm rằng **fsck** gọi đúng lệnh cần thiết, là bạn phải xác định loại hệ thống tệp bằng tuỳ chọn -t. Nếu bạn không xác định, Linux sẽ tự đi tìm loại hệ bằng cách lần lượt duyệt hết mọi hệ thống tệp có trong /etc/fstab và sử dụng loại hệ được tệp này xác định. Nếu đến đây vẫn không xác định được thì Linux sẽ giả định rằng bạn đang sử dụng loại hệ thống tệp Minix.

Ghi chú: Bởi vì có nhiều khả năng hệ thống tệp lại là loại khác chứ không phải Minix, bạn phải cẩn thận và báo cho **fsck** biết bằng tuỳ chọn -t. Điều này đặc biệt quan trọng khi bạn muốn kiểm tra một hệ thống tệp không được liệt kê trong tệp /etc/fstab.

Bạn nên **unmount** hệ thống tệp trước khi kiểm tra để chắc chắn thành công với lệnh fsck. Nhưng việc kiểm tra hệ thống tệp gốc (root) có phần đặc biệt hơn. Bạn không thể trực tiếp **unmount** hệ thống tệp gốc vì Linux cần phải sử dụng hệ thống tệp này để hoạt động. Muốn kiểm tra hệ thống tệp gốc, bạn phải khởi động từ một đĩa mềm bảo trì có lưu hệ thống tệp gốc. Từ đĩa mềm này bạn ra lệnh **fsck** và xác định tên thiết bị đặc biệt của hệ thống tệp gốc. Trong trường hợp **fsck** có thay đổi gì đó trong hệ thống tệp, bạn phải lập tức khởi động lại hệ thống. Như thế Linux sẽ đọc các thông tin quan trọng về hệ thống tệp và giữ cho nó không bị hỏng hóc.

Muốn khởi động lại, bạn có thể sử dụng lệnh shutdown -r hoặc reboot.

14.5 Tạo ra và định dạng hệ thống tệp

Khi gắn thêm một ổ cứng mới vào hệ thống hoặc khi muốn thay đổi thông tin về phân vùng trên ổ cứng có sẵn, bạn cần thực hiện tiến trình tạo ra hệ thống tệp như từ một ổ cứng nguyên xi.

Giả sử đã thêm một ổ cứng vào hệ thống, bạn phải thiết lập thông tin về phân vùng đĩa cứng, rồi tạo ra các hệ thống tệp, sau đó Linux mới đọc và hiểu được đĩa mới. Muốn thay đổi thông tin về phân vùng, bạn sử dụng lệnh **fdisk**. Sau khi phân vùng xong, bạn dùng lệnh **mkfs** để tạo các hệ thống tệp.

14.5.1 Phân vùng ổ đĩa cứng bằng lệnh fsdisk

Lệnh **fdisk** dùng để tạo ra các phân vùng trên đĩa cứng, sau đó lập các thuộc tính và báo cho Linux biết trên phân vùng nào có loại hệ thống tệp gì.

Nếu muốn cài đặt Linux trên một đĩa cứng từng chạy DOS trước đó, bạn phải dùng **fdisk** để thay đổi các thông tin về phân vùng. Khi chuẩn bị tiến hành việc này, bạn hãy sao lưu toàn bộ đĩa cứng, bởi vì **fdisk** sẽ xoá tất cả dữ liệu sẵn có trên đó. Ngoài ra vì **fdisk** cũng là chương trình tương tác theo **menu** chứ không phải là một lệnh đơn thuần, bạn chỉ được chạy **fdisk** trên hệ thống tệp chưa được mount.

Để khởi động **fdisk**, bạn gõ:

```
fdisk [ổ_đĩa]
```

Với ổ_đĩa là tên ổ đĩa vật lý mà bạn muốn thao tác. Nếu nó không được xác định, **fdisk** sẽ giả định đó là ổ /dev/hda. Thí dụ muốn chạy **fdisk** trên ổ cứng IDE thứ hai, tại dấu nhắc lệnh dành cho superuser bạn gõ:

```
fdisk /dev/hdb
```

Như một lệnh chạy bằng **menu**, **fdisk** có nhiều tùy chọn được tóm tắt ở bảng sau.

*Bảng 14.8: Các tùy chọn sẵn có trong **menu fdisk***

Tùy chọn	Mô tả
a	Bật tắt cờ boot (bootable) để có thể khởi động từ một phân vùng.
c	Bật tắt cờ báo tương thích với DOS trên một phân vùng.
d	Xoá một phân vùng.
l	Liệt kê các loại phân vùng mà **fdisk** hiểu được.
m	Hiển thị danh sách các tùy chọn khả dụng.
n	Thêm vào một phân vùng mới.
p	In bảng phân vùng của đĩa hiện hành.
q	Thoát ra mà không lưu lại thay đổi nào cả.
t	Thay đổi loại hệ thống tệp của một phân vùng

u	Thay đổi các đơn vị hiển thị hoặc mục ghi.	
v	Kiểm tra bảng phân vùng.	
w	Ghi bảng phân vùng vào đĩa và thoát khỏi.	
x	Liệt kê các hành động bổ sung dành cho chuyên gia (xem các ô dưới)	
	b	Di dời vị trí bắt đầu của dữ liệu trong một phân vùng.
	c	Thay đổi số của các cylinder.
	d	In ra các dữ liệu sống của bảng phân vùng.
	e	Liệt kê các phân vùng mở rộng (extended) trên đĩa.
	h	Thay đổi số đầu đọc trên đĩa.
	r	Trở về **menu** chính.
	s	Thay đổi số cung (sector) trên đĩa.

fdisk có thể thiết lập bất kỳ loại hệ thống tệp nào cho phân vùng. Với các phân vùng dùng cho Linux, bạn chỉ nên dùng lệnh **fdisk** của Linux. Đối với các phân vùng dùng cho DOS hoặc OS/2, bạn phải dùng công cụ **fdisk** riêng của các hệ điều hành ấy, rồi sau đó mới dùng **fdisk** của Linux để đánh dấu đó là Linux native hoặc Linux swap.

Bảng 14.9 liệt kê các phân vùng được **fdisk** của Linux chấp nhận. Mỗi loại phân vùng có riêng một mã dạng Hex (thập lục phân) để nhận dạng. Muốn lập một loại phân vùng, bạn phải gõ mã tương ứng với **fdisk.**

*Bảng 14.9: Mã và loại phân vùng cho lệnh **fdisk** của Linux*

Kiểu	**Giá trị**	**Kiểu**	**Giá trị**
Empty	00	Novell Netware 386	65
DOS 12 bit FAT	01	PIC/IX	75
XENIX root	02	Old MINIX	80
XENIX usr	03	Linux/MINUX	81
DOS 16-bit <=32 M	04	Linux swap	82
Extended	05	Linux native	83
DOS 16-bit >=32 M	06	Linux extended	85
OS/2 HPFS	07	Amoeba	93
AIX	08	Amoeba BBT	94
AIX bootable	09	BSD/386	a5
OS/2 Boot Manager	0a	OpenBSD	a6
Win95 FAT 32	0b	NEXTSTEP	a7
Win95 FAT 32 (LBA)	0c	BSDI fs	b7

Win95 FAT 16 (LBA)	0e	BSDI swap	b8
Win95 FAT Extended (LBA)	0f	Syrinx	c7
Venix 80286	40	CP/M	db
Novell	51	DOS access	e1
Microport	52	DOS R/O	e3
GNU HURD	63	DOS secondary	f2
Novell Netware 286	64	BBT	ff

Những mục sau sẽ hướng dẫn bạn cách dùng **fdisk**. Đầu tiên là thí dụ dùng **fdisk** để lập các phân vùng trên đĩa cứng cho Linux.

Giả sử bạn muốn lập cấu hình của ổ IDE đầu tiên dành cho Linux. Trước hết bạn phải sao lưu tất cả dữ liệu. Tên của ổ đĩa cứng IDE đầu tiên là /dev/hda, vốn là một mặc định của Linux.

Linux có quy ước chung về ký hiệu các ổ cứng như sau:

Các ổ có giao diện IDE, EIDE, ... bắt đầu bằng chữ hd

Các ổ có giao diện SCSI bắt đầu bằng chữ sd

Ổ master thuộc IDE 0: ký hiệu là a

Ổ slave thuộc IDE 1: b

Ổ master thuộc IDE 1: c

Ổ slave thuộc IDE 1: d

Cuối cùng các phân vùng trên mỗi đĩa sẽ có số thứ tự tương ứng. Thí dụ: hdd2 là phân vùng thứ 2 trên ổ đĩa cứng slave thuộc IDE 1

14.5.2 Chạy lệnh fdisk

Bạn gõ:

```
fdisk /dev/hda
```

và màn hình hiển thị như sau:

```
Using /dev/hda as default device!
Command (m for help):
```

Có nghĩa là **fdisk** đang hiểu ổ /dev/hda là thiết bị mà bạn đang thao tác. Bạn luôn phải coi lại xem đúng đó là ổ đĩa mình muốn xử lý không. Sau đó Linux hiển thị tiếp dấu nhắc lệnh **fdisk.**

Lưu ý: nếu bạn sử dụng các ổ cứng lớn, bạn sẽ nhận được cảnh báo (vô hại) sau:

```
[root@localhost root]# fdisk /dev/hda
The number of cylinders for this disk is set to 2842.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
```

```
software that runs as boot time (e.g., old versions of LILO)
booting and partitioning software from other Oss (e.g., DOS FDISK, OS/2 FDISK)
```

14.5.3 Hiển thị bảng phân vùng hiện hành

Bạn gõ lệnh (tùy chọn) p để **fdisk** hiển thị bảng phân vùng hiện hành:

```
Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 2482 cylinders
Units = cylinders of 16065 * 512 bytes
```

Device Boot	Start	End	Blocks	Id	System
/dev/hda1	1	6	48163+	83	Linux
/dev/hda2	7	159	1228972+	82	Linux swap
/dev/hda3	160	2482	18659497+	83	Linux

Màn hình trên cho biết đĩa cứng hiện hành /dev/hda có 255 đầu đọc, 63 cung và 2482 trụ. Đơn vị tính là các trụ có kích cỡ 16065 * 512 = 8225280 byte. Bởi vì có 2482 trụ và mỗi trụ gồm 8225280 byte, có thể suy ra dung lượng đĩa là 2482*8225280 = 20415144960 byte, tương đương khoảng 20 GB.

14.5.4 Tạo ra phân vùng mới

Giả sử bạn muốn tạo một phân vùng 48 MB cho home directory của user. Bạn gõ lệnh (tùy chọn) n để tạo phân vùng mới:

```
Command (m for help): n
Command action:
e extended
p primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-2482):1
Last cylinder or + size or +sizeM or + sizeK (1-2482): +48M
```

Lệnh n sẽ hiển thị một **menu** khác. Bạn phải lựa chọn xem mình muốn tạo ra phân vùng mở rộng (extended) hoặc sơ cấp (primary). Trừ khi đĩa cứng của bạn phải có hơn 4 phân vùng, nếu không bạn chỉ tạo ra phân vùng sơ cấp thôi. Sau đó **fdisk** hỏi xem bạn muốn tạo ra bao nhiêu phân vùng. Và bởi vì đấy là phân vùng đầu tiên trên đĩa, bạn trả lời là 1. Tiếp theo máy hỏi bạn về cylinder đầu tiên cho phân vùng, để xác định xem vùng dữ liệu bắt đầu nơi nào. Và trong thí dụ này bạn chỉ mới có một phân vùng nên bạn trả lời là cylinder 1.

Dòng tiếp theo hỏi bạn về kích thước phân vùng. Bạn có thể trả lời bằng đơn vị byte, kilobyte, hoặc megabyte. Và vì bạn muốn tạo phân vùng 48 MB nên bạn gõ vào +48MB.

14.5.5 Kiểm tra lại bảng phân vùng

Đến đây bạn nên kiểm tra bảng phân vùng xem **fdisk** đã làm được những gì, bằng cách gõ lệnh (tùy chọn) p:

```
Command (m for help):p
Disk /dev/hda: 255 headers, 63 sectors, 2482 cylinders
Units = cylinders of 16065 * 512 bytes
```

Device Boot	Start	End	Blocks	Id	System
/dev/hda1	1	6	48163+	83	Linux

Bảng trên cho biết đĩa cứng có một phân vùng tên /dev/hda1 trải dài từ cylinder 1 đến cylinder 6, chiếm 48163 block và là phân vùng loại 83 của Linux.

14.5.6 Tạo ra phân vùng swap

Giả sử bạn muốn tạo ra một phân vùng swap khoảng 1200 MB ở phần còn lại của đĩa cứng. Cách làm cũng giống như đối với phân vùng trước (xem mục 14.5.4), bằng cách gõ lệnh (tùy chọn) n:

```
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (7-2482):7
Last cylinder or + size or +sizeM or + sizeK (8-2482): 159
```

Ở thí dụ này bạn gõ số 2 là số phân vùng thứ hai. Khi **fdisk** hiện dấu nhắc cho cylinder đầu tiên, bạn nhận thấy đó là từ 7 đến 2482, bởi vì phân vùng đầu tiên đã chiếm hết những gì ở phía trước cylinder 7. Vì vậy bạn gõ 7 là số cylinder bắt đầu phân vùng thứ hai và dùng phân vùng này làm swap. Bạn gõ 159 là cylinder cuối cùng của phân vùng swap và nó sẽ có kích cỡ khoảng 1200 MB (159-6 = 153 cylinder).

Ghi chú: Đến đây bạn nên khai báo kích thước của phân vùng cuối cùng, cũng bằng đơn vị cylinder để bảo đảm rằng sẽ sử dụng hết dung lượng đĩa cứng. Đĩa cứng của bạn còn khoảng 18000 MB cho phân vùng thứ ba, song nếu bạn chọn đơn vị kích thước bằng MB thì các tính toán nội bộ của lệnh **fdisk** sẽ có thể để thừa vài cylinder, vì thế bạn nên chọn đơn vị kích thước bằng cylinder và gõ 2482 cho số cylinder cuối cùng của phân vùng thứ 3.

Lưu ý: Đôi lúc bạn sẽ thấy một cảnh báo như sau (với xxx là một con số nào đó):

```
Warning: Linux cannot currently use the last xxx sectors of this partition.
```

Bạn không cần quan tâm đến cảnh báo này, vì nó được sinh ra vào thời mà Linux không xử lý được các hệ thống tệp lớn hơn 64MB.

14.5.7 Kiểm tra lại kích thước

Bạn đã tạo xong hai phân vùng cần thiết. Bạn nên hiện lại bảng phân vùng lần nữa để kiểm tra xem các kích thước đã như ý chưa:

```
Command (m for help):p
Disk /dev/hda: 255 headers, 63 sectors, 2482 cylinders
Units = cylinders of 16065 * 512 bytes
```

Device Boot	Start	End	Blocks	Id	System
/dev/hda1	1	6	48163+	83	Linux
/dev/hda2	7	159	1228972+	82	Linux swap

Như bạn đã nhận thấy, /dev/hda1 sử dụng từ cylinder 1 đến cylinder 6 với kích thước 48163 block tương đương khoảng 48 MB. Phân vùng /dev/hda2 dùng phần còn lại là từ cylinder 7 đến cylinder 159 với kích thước 1228972 block, tương đương khoảng 1200 MB.

14.5.8 Thay đổi loại phân vùng

Điều tiếp theo bạn cần làm là thay đổi loại cho từng phân vùng. Tại dấu nhắc **fdisk**, bạn gõ lệnh (tùy chọn) <t>. Mã Id phổ quát nhất cho hệ thống tệp Linux tiêu chuẩn là 83, tức Linux native, trong khi đó phân vùng swap được tạo lập với Id=82.

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 83
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
```

Khi bạn gõ lệnh t, máy sẽ hỏi muốn thay đổi phân vùng thứ mấy. Tiếp theo bạn phải gõ một mã Hex (thập lục phân) để máy biết bạn muốn gán loại nào cho phân vùng. Như đã nói trên, hệ thống tệp Linux bình thường có mã 83 tức là Linux native và phân vùng swap có mã 82. Nếu muốn xem danh sách mã, bạn gõ L.

14.5.9 Kiểm tra lần cuối

Đến đây khi đã tạo và gán phân vùng xong, bạn hiển thị lại bảng phân vùng lần chót xem mọi thứ đã như ý muốn chưa.

```
Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 2482 cylinders
Units = cylinders of 16065 * 512 bytes
```

Device Boot	Start	End	Blocks	Id	System
/dev/hda1	1	6	48163+	83	Linux
/dev/hda2	7	159	1228972+	82	Linux swap
/dev/hda3	160	2482	18659497+	83	Linux

Như bạn nhận thấy, mọi phân vùng đã ở đúng vị trí, đúng kích thước vào được gán đúng loại. Bạn cần thực hiện thao tác cuối cùng là gõ lệnh w để ghi bảng phân vùng vào đĩa và thoát khỏi:

```
Command (m for help): w
#
```

Nhớ phải gõ lệnh này, nếu không những gì bạn mới thiết lập xong sẽ không được ghi vào đĩa. Ngược lại nếu đang làm nửa chừng bạn muốn thoát khỏi mà không thay đổi gì cả, thì gõ lệnh q. Sau khi đã dùng **fdisk** để thay đổi nội dung đĩa cứng, bạn phải khởi động lại hệ thống để Linux cập nhật các thông tin về phân vùng trong kernel.

14.6 Xây dựng hệ thống tệp bằng lệnh mkfs

Bạn đã tạo ra hệ thống tệp bằng lệnh fsdisk nhưng như thế vẫn chưa chứa dữ liệu được. Bạn phải xây dựng hệ thống tệp bằng lệnh mkfs. Lệnh **mkfs** cũng gọi nhiều chương trình khác nhau, cú pháp như sau:

```
mkfs [-V][-t loại_fs][tuỳ chọn_fs] filesys[blocks]
```

với filesys là tên thiết bị chứa hệ thống tệp mà bạn muốn xây dựng, chẳng hạn như /dev/hda1.

Lưu ý: **mkfs** cũng chấp nhận nếu bạn dùng tên của một **mount point**, chẳng hạn như /home là tên của hệ thống tệp. Nếu chạy **mkfs** trên một hệ thống tệp "sống", bạn sẽ có nguy cơ làm hỏng tất cả dữ liệu trên hệ thống tệp ấy.

Bảng 14.10: Các đối số của lệnh mkfs

Đối số	**Mô tả**
-V	Chế độ thông báo chi tiết, liệt kê cả các lệnh thuộc hệ thống tệp đang được thực thi. Nếu gõ tuỳ chọn này hơn một lần, mọi lệnh liên quan đến hệ thống tệp sẽ bị dừng hẳn.
-t loại_fs	Xác định loại hệ thống tệp cần được xây dựng. Nếu không có tuỳ chọn này, **mkfs** sẽ đi tìm filesys trong /etc/fstab và sử dụng mục ghi tương ứng. Đến đây nếu vẫn không suy ra được loại hệ thống tệp, máy sẽ tự động tạo ra hệ thống tệp dạng MINIX.
-tuỳ_chọn_fs	Xác định các tuỳ chọn liên quan đến hệ thống tệp cần được giao cho chương trình hiện hành. Các tuỳ chọn sau đây được hầu hết các chương trình xây dựng hệ thống tệp hỗ trợ:
	-c: Trước khi xây dựng hệ thống tệp, kiểm tra thiết bị xem có bị bad block không
	-l tên tệp: Dựa vào tên tệp để liệt kê danh sách các bad block.
	-v: Bắt chương trình cho xem báo cáo chi tiết.
filesys	Xác định thiết bị đang chứa hệ thống tệp. Tham số này bắt buộc phải có.
blocks	Xác định số lượng các block dùng cho hệ thống tệp.

14.7 Sử dụng tệp swap và phân vùng swap

Có một khoảng swap (Swap space) trên hệ thống của bạn dùng để làm bộ nhớ ảo (giáo trình này không thể bàn kỹ các vấn đề liên quan đến bộ nhớ ảo, nếu quan tâm, xin bạn tham khảo những tài liệu nói về hệ điều hành máy tính nói chung).

Linux hỗ trợ hai loại khoảng swap là phân vùng swap và tệp swap.

Phân vùng swap là một phân vùng vật lý, mang mã số Id=82.

Tệp swap là một tệp lớn nằm trên hệ thống tệp bình thường.

Thay vì dùng tệp swap, bạn nên dùng phân vùng swap. Lý do là mọi truy cập vào tệp swap đều phải đi qua con đường hệ thống tệp bình thường, trong khi các block hình thành nên tệp swap lại có thể không nằm kề sát bên nhau, do đó tốc độ truy cập sẽ không bằng phân vùng swap. Thao tác vào ra phân vùng swap tác động trực tiếp đến thiết bị và các block của phân vùng swap luôn nằm sát bên nhau. Hơn nữa, việc đặt khoảng swap bên ngoài một hệ thống tệp bình thường sẽ giảm thiểu rủi ro hỏng hóc hệ thống tệp một khi tệp swap bị sự cố.

14.7.1 Tạo ra phân vùng swap

Trước đó bạn phải dùng **fdisk** tạo một phân vùng trên đĩa cứng và gán cho nó loại 82. Sau khi tạo xong phân vùng swap, bạn kích hoạt theo hai bước như sau:

A. Trước hết, bạn chuẩn bị phân vùng giống như chuẩn bị hệ thống tệp. Thay vì dùng **mkfs**, bạn dùng **mkswap** theo cú pháp:

```
mkswap [-c] thiết_bị [số_lượng_block]
```

Với thiết_bị là tên của phân vùng swap, chẳng hạn như /dev/hda2, hay cũng có thể là một tệp.

Số_lượng_block là kích cỡ của phân vùng mà bạn định tạo ra. Thông số này nay là một tuỳ chọn dùng để tương thích với các phiên bản cũ của Linux. Muốn có khái niệm về kích thước bằng block, bạn chạy lệnh **fdisk** và nhìn vào bảng phân vùng. Trong thí dụ ở mục "14.5.9 Kiểm tra lại kích thước", kích thước của /dev/hda2 là 1228972 block.

Lấy thí dụ ấy, lệnh xây dựng phân vùng swap trên /dev/hda2 như sau:

```
mkswap -c /dev/hda2 1228972
```

B. Sau khi **mkswap** chạy xong, bạn kích hoạt bằng lệnh swapon theo cú pháp:

```
swapon thiết_bị
```

với thiết_bị là thiết bị mà bạn định dùng làm swap space. Khi máy khởi động, Linux sẽ gọi swapon -a để **mount** mọi phân vùng swap khả dụng được liệt kê trong tệp /etc/fstab.

Ghi chú: Thông tin về phân vùng swap và tệp swap vừa được tạo ra, bạn luôn nhớ đưa vào mục ghi trong tệp /etc/fstab để Linux biết tự động truy cập chúng lúc khởi động.

14.7.2 Tạo ra tệp swap

Các tệp swap đặc biệt hữu ích khi bạn muốn mở rộng swap space nhưng không còn khoảng trống trên đĩa để tạo ra phân vùng swap chuyên dụng. Xây dựng tệp swap chỉ khác xây dựng phân vùng swap ở chỗ là bạn phải tạo ra tệp trước khi chạy **mkswap** và swapon.

Để tạo tệp swap, bạn dùng lệnh dd, thường dùng để chép những lượng dữ liệu lớn. Xin bạn tham khảo trang **man** của dd để biết thêm chi tiết. Trước khi tạo tệp swap, bạn phải xác định tên và kích thước. Với Linux, một block bằng 1024 byte. Chẳng hạn để tạo ra một tệp swap khoảng 10 MB mang tên /swap, bạn gõ:

```
#dd if=/dev/zero of=/swap bs=1024 count =10240
```

of=/swap xác định rằng tệp sẽ mang tên /swap và count 10240 lập kích thước của tệp là 10240 block, tương đương 10 MB. Tiếp theo bạn dùng **mkswap** để biến tệp thành swap space:

```
#mkswap /swap 10240
```

Nên nhớ là phải báo kích thước tệp cho **mkswap** (với các phiên bản Red Hat 6.x trở về trước). Trước khi chạy swapon, bạn phải ghi tệp vào đĩa bằng lệnh /etc/sync.

Đến đây, cũng giống như với phân vùng swap, bạn kích hoạt tệp swap bằng lệnh swapon:

```
#swapon /swap
```

Khi muốn huỷ một tệp swap, bạn phải giải ngũ cho nó bằng lệnh swapoff:

```
#swapoff /swap
```

Rồi mới xoá tệp.

Chương 15. Sử dụng Samba

Chương này giúp bạn cài đặt, lập cấu hình và sử dụng Samba, một ứng dụng hỗ trợ giao thức SMB (Session Message Block) trên Linux, cho phép chia sẻ hệ thống tệp và cả máy in giữa Linux với Windows 95, 98, NT, hoặc 2000.

Microsoft và Intel phát triển giao thức SMB vào năm 1987, rồi SMB được tương thích với các hệ UNIX khác nhau và sau cùng là với Linux.

Chương này bao gồm những chủ đề chính sau đây:

-*Cài đặt Samba*

-*Lập cấu hình Samba trên Linux*

-*Chạy server Samba*

-*Sử dụng smbclient*

Samba bao gồm các thành phần như smbd, nmbd, smbclient, smbstatus và testparm. Daemon mang tên smbd cung cấp dịch vụ in và tệp cho client SMB, chẳng hạn như Windows **for** Workgroups, Windows NT, Windows 2000 hoặc Lan Manager. Tệp cấu hình của daemon này được mô tả trong smb.conf.

Daemon nmbd hỗ trợ dịch vụ tên NETBIOS (NetBIOS Name Service), cho phép các máy tính khác truy cập và sử dụng các tài nguyên được cấp bởi máy chủ Samba. Bạn có thể tương tác với daemon này để truy cập các daemon chuyên về dịch vụ tên khác.

Trình smbclient hoạt động như một client bình thường, tương tự FTP. Việc này có ích khi bạn truy cập những phần mềm SMB dùng chung trên các server tương thích khác, đồng thời còn cho phép máy UNIX gửi tệp in về một máy in kết nối với bất kỳ server SMB nào, chẳng hạn như một PC chạy Windows 98.

Tiện ích testparm giúp bạn kiểm nghiệm tệp cấu hình smb.conf. Còn tiện ích smbstatus báo cho bạn biết ai đang sử dụng server smbd.

Ghi chú: Microsoft đã đệ trình lên IETF (Internet Engineering Task Force) yêu cầu thúc đẩy việc tiêu chuẩn hoá giao thức SMB, như là một Hệ thống tệp Internet phổ quát (Common Internet File System, CIFS).

15.1 Cài đặt Samba

Bạn có thể cài Samba cùng lúc với Linux hoặc cài lúc khác bằng RPM.

Ngay trong bộ cài đặt Red Hat, đã có sẵn Samba, vấn đề là bạn có chọn nó hay không, bởi vì Samba không phải là tuỳ chọn mặc định của Red Hat. Tuy nhiên phiên bản Samba đi kèm Red Hat thường hơi bị cũ. Bạn có thể cài mới hay cập nhật Samba chép xuống từ Website của Samba.

Muốn cài gói phần mềm Samba, đầu tiên bạn tải nó xuống từ website của Samba http://va.samba.org/samba/ftp/Binary_Packages/redhat/RPMS/. Thí dụ phiên bản tải xuống là samba-2.2.3a.200202060i386.rpm, bạn cài đặt mới bằng lệnh:

```
rpm -ivh samba-2.2.3a-20020206.i386.rpm
```

Trong trường hợp bạn muốn cập nhật lại bản Samba cũ đã có, bạn có thể thực hiện lệnh sau:

```
rpm -Uvh samba -2.2.3a.20020206.i386.rpm
```

Để biết có cần cập nhật hay không, bạn kiểm tra phiên bản của gói Samba đã cài bằng lệnh:

```
rpm -q samba
```

Gói phần mềm này chứa đủ các tệp cần thiết chạy Samba, kể cả hai daemon sơ cấp smbd và nmbd. Tuy nhiên nếu đang dùng bản phát hành khác, bạn có thể biên dịch lại các tệp chương trình.

15.2 Lập cấu hình Samba trên Linux

Tệp cấu hình smb.conf nằm trong thư mục /etc/samba/ là một tệp văn bản theo kiểu các tệp cấu hình của Windows (*.INI). Có thể chia thành 2 phần chính:

A. Cấu hình tổng quát (Global Settings): cho bạn xác định các tham số kết nối

B. Phân định chia sẻ (Share Definitions): cho bạn xác định các tham số chia sẻ, đó là các thư mục có thể được truy cập từ một số người dùng nào đó trên mạng và thông qua mạng. Trong thành phần B bạn còn có thể chia làm 3 mục con:

-Homes: xác định thư mục gốc của người dùng.

-Printers: Xác định các máy in dùng chung.

-Shares: Xác định các chia sẻ khác.

Danh sách 15.1 giới thiệu các mặc định đi kèm với Red Hat (trong bộ cài đặt của máy chủ IBM)

Ghi chú: Dấu thăng (#) hay chấm phẩy (;) ở đầu dòng cho biết đấy là dòng chú thích, để server Samba không xử lý nội dung của dòng này. Trong thí dụ sau, dấu "#" sẽ dùng cho chú thích còn dấu ";" sẽ dùng cho các giá trị của cấu hình mà bạn có thể bỏ qua (hay cho hoạt động nếu xóa ";" đi).

Danh sách 15.1 Tệp cấu hình smb.conf của Samba

```
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
```

```
# to check that you have not made any basic syntactic errors.
#
#==================Global Settings=====================
[global]
# workgroup = NT-Domain-Name or Workgroup-Name
workgroup = MYGROUP

# server string is the equivalent of the NT Description field
server string = Samba Server

# This option is important for security. It allows you to restrict
# connections to machines which are on your local network. The
# following example restricts access to two C class networks and
# the I'loopback" interface. For more examples of the syntax see
# the smb.conf man page

; hosts allow = 192.168.1. 192.168.2. 127.

# if you want to automatically load your printer list rather
# than setting them up individually then you'll need this
printcap name = /etc/printcap
load printers = yes

# It should not be necessary to spell out the print system type unless
# yours is non-standard. Currently supported print systems include:
# bsd, sysv, plp, lprng, aix, hpux, qnx
printing = lprng

# Uncomment this if you want a guest account, you must a~d this to
/etc/passwd # otherwise the user "nobody" is used

;guest account = pcguest

# this tells Samba to use a separate log file for each machine
# that connects
log file = /var/log/samba/%m.log

# Put a capping on the size of the log files (in Kb).
max log size = 0
```

```
# Security mode. Most people will want user level security. See
# security_level.txt for details.
security = user

# Use password server option only with security = server
# The argument list may include:
#     password server = My_PDC_Name [My-BDC_Namel [My_Next_BDC-Namel
# or to auto-locate the domain controller/s
#     password server = *
;     password server = <NT-Server-Name>

# Password Level allows matching of - n - characters of the password for
# all combinations of upper and lower case.
;password level = 8
;username level - 8

# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba documentation.
# Do not enable this option unless you have read those documents
encrypt passwords = yes
smb passwd file = /etc/samba/smbpasswd
# The following is needed to keep smbclient from spouting spurious errors
# when Samba is built with support for SSL.
;     ssl CA certFile = /usr/share/ssl/certs/ca-bundle.crt

# The following are needed to allow password changing from Windows to
# update the Linux sytsem password also.
# NOTE: Use these with 'encrypt passwords' and 'smb passwd file' above.
# NOTE2: You do NOT need these to allow workstations to change only
#           the encrypted SMB passwords. They allow the Unix password
#           to be kept in sync with the SMB password.
      unix password sync = Yes
      passwd program = /usr/bin/passwd Xu
      passwd chat = *New*password* Xn\n *Retype*new*password* Xn\n
*passwd:*all*authentication*tokens*updated*successfully*

# You can use PAM's password change control flag for Samba. If
# enabled, then PAM will be used for password changes when requested
# by an SMB client instead of the program listed in passwd program.
# It should be possible to enable this without changing your passwd
```

```
# chat parameter for most setups.

pam password change = yes

# Unix users can map to different SMB User names
;username map = /etc/samba/smbusers

#Using the following line enables you to customise your configuration
# on a per machine basis. The %m gets replaced with the netbios name
# of the machine that is connecting
; include = /etc/samba/smb.confJm

# This parameter will control whether or not Samba should obey PAM's
# account and session management directives. The default behavior is
# to use PAM for clear text authentication only and to ignore any
# account or session management. Note that Samba always ignores PAM
# for authentication in the case of encrypt passwords = yes
obey pam restrictions = yes

# Most people will find that this option gives better performance.
# See speed.txt and the manual pages for details
      socket options = TCP-NODELAY SO-RCVBUF=8192 SO-SNDBUF=8192

# Configure Samba to use multiple interfaces
# If you have multiple network interfaces then you must list them
# here. See the man page for details.
; interfaces = 192.168.12.2/24 192.168.13.2/24

# Configure remote browse list synchronisation here
# request announcement to, or browse list sync from:
# a specific host or from / to a whole subnet (see below)
; remote browse sync = 192.168.3.25 192.168.5.255

# Cause this host to announce itself to local subnets here
; remote announce = 192.168.1.255 192.168.2.44

# Browser Control Options:
# set local master to no if you don't want Samba to become a master
# browser on your network. Otherwise the normal election rules apply
; local master = no
```

```
# OS Level determines the precedence of this server in master browser
# elections. The default value should be reasonable
      os level = 33

# Domain Master specifies Samba to be the Domain Master Browser. This
# allows Samba to collate browse lists between subnets. Don't use this
# if you already have a Windows NT domain controller doing this job
; domain master = yes

# Preferred Master causes Samba to force a local browser election on
startup
# and gives it a slightly higher chance of winning the election
; preferred master = yes

# Enable this if you want Samba to be a domain logon server for
# Windows95 workstations.
; domain logons = yes

# if you enable domain logons then you may want a per-machine or
# per user logon script
# run a specific logon batch file per workstation (machine)
; logon script = %m.bat

# run a specific logon batch file per username
; logon script = XU.bat

# Where to store roving profiles (only for Win95 and WinNT)
#           %L substitutes for this servers netbios name, %U is username
#           You must uncomment the [Profiles] share below
; logon path = \\%L\Profiles\%U

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable it's WINS
Server
; wins support = yes

# WINS Server - Tells the NMBD components of Samba to be a WINS Client
# Note: Samba can be either a WINS Server, or a WINS Client, but NOT both
; wins server = w.x.y.z
```

```
# WINS Proxy - Tells Samba to answer name resolution queries on
# behalf of a non WINS capable client, for this to work there must be
# at least one WINS Server on the network. The default is NO.
; wins proxy = yes

# DNS Proxy - tells Samba whether or not to try to resolve NetBIOS names
# via DNS nslookups. The built-in default for versions 1.9.17 is yes.
# this has been changed in version 1.9.18 to no.
; dns proxy = no

# Case Preservation can be handy - system default is -no-
# NOTE: These can be set on a per share basis
; preserve case = no
; short preserve case = no
# Default case is normally upper case for all DOS files
; default case = lower

# Be very careful with case sensitivity - it can break things!
; case sensitive = no
#==========================Share Definitions=====================
[homes]
      comment = Home Directories
      browseable = no
      writable = yes
      valid users = XS
      create mode = 0664
      directory mode = 0775

#If you want users samba doesn't recognize to be mapped to a guest user
; map to guest = bad user

# Un-comment the following and create the netlogon directory for Domain
Logons

; [netlogon] comment = Network Logon Service
; path = /usr/local/samba/lib/netlogon
; guest ok = yes
; writable = no
; share modes = no
```

```
# Un-comment the following to provide a specific roving profile share
# the default is to use the user's home directory
; [Profiles]
; path = /usr/local/samba/profiles
; browseable = no
; guest ok = yes

# NOTE: If you have a BSD-style print system there is no need to
# specifically define each individual printer
[printers]
comment = All Printers
path = /var/spool/samba
      browseable =      no

# Set public yes to allow user 'guest account' to print

      guest ok = no
      writable = no
      printable yes

# This one is useful for people to share files

; [tmpl comment = Temporary file space
; path = /tmp
; read only = no
; public = yes

# A   publicly accessible directory but read only, except for people in
# the "staff" group
;[public]
;     comment = Public Stuff
;     path = /home/samba
;     public = yes
;     writable = yes
;     printable = no
;     write list = @staff

# Other examples.
# A   private printer. usable only by fred. Spool dabạn will be placed in
fred's
```

```
# home directory. Note that fred must have write access to the spool directory.
# wherever it is.

;[fredsprn]
;     comment = Fred's Printer
;     valid users = fred
;     path = /home/fred
;     printer = freds_printer
;     public = no
;     writable = no
;     printable = yes
# A private directory. usable only by fred. Note that fred requires write
# access to the directory.
;[fredsdir]
;     comment = Fred's Service
;     path = /usr/somewhere/private
;     valid users = fred
;     public = no
;     writable = yes
;     printable = no

# a service which has a different directory for each machine that connects
# this allows you to tailor configurations to incoming machines. You could
# also use the %U option to tailor it by user name.
# The %m gets replaced with the machine name that is connecting.
;[pchome]
;     comment = PC Directories
;     path = /usr/local/pc/%m
;     public = no
;     writable = yes

# A publicly accessible directory. read/write to all users. Note that all files # created in the directory by users will be-owned by the default user. so
# any user with access can delete any other user's files. Obviously this
# directory must be writable by the default user. Another user could of course # be specified, in which case all files would be owned by that user instead.
;[public]
;     path = /usr/somewhere/else/public
;     public = yes
```

```
;     only guest = yes
;     writable = yes
;     printable = no


# The following two entries demonstrate how to share a directory so that
two
# users can place files there that will be owned by the specific users. In
this # setup. the directory should be writable by both users and should
have the
# sticky bit set on it to prevent abuse. Obviously this could be extended
to
# as many users as required.
# as many users as required.
;[myshare]
;     comment = Mary's and Fred's stuff
;     path = /usr/somewhere/shared
;     valid users = mary fred
;     public = no
;     writable = yes
;     printable = no
;     create mask = 0765
```

Tệp smb.conf gồm một loạt những đoạn có tên. Mỗi đoạn bắt đầu bằng tên của chính mình đặt trong ngoặc vuông, chẳng hạn như [global]. Tham số trong mỗi đoạn được xác định bằng từng cặp key=value (khoá=giá trị), chẳng hạn như comment=RedHat Samba Server.

15.2.1 Đoạn [global]

Đoạn [global] kiểm soát tham số của toàn bộ server smb và cung cấp giá trị mặc định cho những đoạn khác. Sau đây là những dòng (trích từ trong danh sách 15.1) có thể cần giải thích kỹ hơn.

```
[global]
workgroup = LINUXRULZ
server string = Samba Server
hosts allow - 192.168.1.192.168.2.127.
include = /etc/samba/smb.conf %m
socket options = TCP_NODELAY SO_RCVBUF = 8192 SO_SNDBUF=8192
```

Dòng đầu tiên của đoạn [global] cho biết chiếc máy này sẽ thuộc về nhóm làm việc LINUXRULZ. Kế đến là các thông tin về máy chủ Samba như địa chỉ IP của nó v.v.

```
load printers = yes
printcap name = /etc/printcap
```

```
printing = lprng
```

Dòng đầu ở bên trên cho biết máy chủ của bạn cần tải nạp hệ thống in ấn lên mạng, dòng tiếp theo cho biết vị trí của tệp cấu hình máy in, dòng cuối cho biết kiểu in của hệ thống (một trong các giá trị: bsd, sysv, plp, lprng, aix, hpux, qnx)

Xem: "Tệp /etc/printcap", Chương 20

```
guest account = pcguest
log file = /var/log/samba/smb.%m
max log size = 50
```

Những dòng trên cung cấp:

-username cho một trương khoản client trên server của bạn. Trương khoản này dùng để nhận diện những user nào được dùng các dịch vụ Samba dành cho client.

-vị trí của tệp ký sự (log file) cho từng client nào muốn truy cập dịch vụ Samba. Biến viết tắt %m bắt server Samba tạo ra cho mỗi client một tệp log riêng biệt.

- kích cỡ tối đa cho những tệp log đã được tạo ra (giá trị tính bằng kB).

Tuỳ theo phiên bản và hệ điều hành, các biến viết tắt có thể khác nhau. Bảng sau giới thiệu các biến số sử dụng cho Samba:

Biến	**Ý nghĩa**
%S	Tên của dịch vụ hiện hành, nếu có.
%P	Thư mục gốc của dịch vụ hiện hành, nếu có.
%u	Tên user của dịch vụ hiện hành, nếu có.
%g	Tên của nhóm chính của %u.
%U	Tên phiên làm việc của user
%G	Tên của nhóm chính của %U.
%H	Thư mục gốc của user (%u)
%v	Phiên bản của Samba.
%h	Tên của host mà samba đang chạy.
%m	Tên NetBIOS của máy client (rất thường dùng)
%L	Tên NtBIOS của máy chủ.
%M	Tên Internet của máy client.
%N	Tên của máy chủ NIS.
%p	Đường dẫn đến thư mục gốc của dịch vụ. Giá trị auto.map có thể coi là gồm %N:%p
%R	Mức giao thức đã chọn sau khi trao đổi về giao thức. Có thể nhận các giá trị CORE, COREPLUS, LANMAN1, LANMAN2 hay NT1.
%d	Chỉ số (ID) của tiến trình máy chủ hiện hành.

%I	Địa chỉ IP của máy client.
%a	Kiến trúc của máy từ xa. Chỉ có một số máy từ xa là được công nhận Samba, WfWg, Win95, WinNT và Win2k. Các hệ khác sẽ được coi là "UNKNOWN".
%T	Ngày và giờ hiện hành.
%$	(envvar) Giá trị của biến môi trường envar.

Tiếp theo là những dòng xác định chế độ bảo mật của hệ thống Samba:

```
security = user
; password server = WINDOWSNT
; password level = 8
encrypt passwords = yes
smb passwd file = /etc/samba/smbpasswd
; unix password sync = Yes
; passwd program = /usr/bin/passwd %u
;     passwd     chat     =     *New*UNIX*password*%n\n*ReType*new*UNIX*pass
word*%n\n*passwd:*all*authentication*tokens*updated*successfully*
```

Mục security có một trong 4 giá trị:

share: Tương tự chế độ bảo mật của Windows 9x File/Print server. User chỉ cần cung cấp password của tài nguyên.

user: Cặp user/password sẽ được máy chủ kiểm tra.

server: Việc lượng giá user/password sẽ do một máy chủ password xác định (được chọn thông qua mục password server - ở đây là có tên là WINDOWSNT).

domain: Về cơ bản, mức bảo mật này giống như mức bảo mật server trừ một việc là máy chủ Samba phải là thành viên của một Windows NT domain.

Mục encrypt passwords cho biết cần hay không cần mã hoá các mật khẩu khi đăng nhập vào máy chủ Samba. Từ Windows 98 SE cùng với các Security Patch, Microsoft mã hoá mọi mật khẩu khi gửi từ một máy trạm Windows 9x/2000/NT đến một máy chủ bất kỳ. Do đó, nếu bạn chọn giá trị "no" thì máy chủ server sẽ không chấp nhận sự đăng nhập của bất kỳ user nào. Chú ý rằng, nếu giá trị là yes (true) thì chỉ có các user có password trong tệp /etc/samba/smbpasswd là có thể thấy máy chủ Samba.

Tiếp theo là các tham số môi trường cho máy chủ Samba:

```
; interfaces = 192.168.12.2/24 192.168.13.2/24
; remote browse sync = 192.168.3.25 192.168.5.255
; remote announce = 192.168.1.255 192.168.2.44
; local master = no
; os level = 33
; domain master = yes
; preferred master = yes
; domain controller = <NT-Domain-Controller-SMBName>
; domain logons = yes
```

```
; logon script = %m.bat
; logon script = %U.bat
; logon path = \\%L\Profiles\%U
; name resolve order = win lmhosts bcast
; wins support = yes
; wins server = w.x.y.z
; wins proxy = yes
; dns proxy = no
; preserve case = no
; short preserve case = no
; default case = lower
; case sensitive = no
```

Các giá trị trên dùng để xác định môi trường hoạt động của máy chủ Samba:

-Có hoạt động như một máy chủ WINS không ("wins support = "),

-Chỉ rõ hay không địa chỉ IP và netmask của giao tiếp mạng ("interfaces="),

-Sử dụng một máy chủ WINS khác ("wins server="),

-v.v.

Các tham số preserve case và short preserve case cho server Samba biết phải ghi lại chữ thường hoặc chữ hoa khi có thông tin được ghi vào server. Việc này quan trọng bởi vì tên tệp của Windows không nhất thiết phải phân biệt dạng chữ thường hay chữ hoa, trong khi Linux lại phân biệt hẳn hoi.

15.2.2 Đoạn [homes]

Đoạn [homes] giúp các client kết nối với home directory của mình trên server mà không cần phải có mục ghi rõ ràng trong tệp smb.conf. Khi user phát đi yêu cầu sử dụng dịch vụ, server Samba sẽ tìm trong tệp smb.conf đoạn tương ứng với dịch vụ được yêu cầu. Nếu không tìm ra mục ghi đó, Samba kiểm tra xem có đoạn [homes] không. Nếu có, Samba xem tên của mục được yêu cầu như là tên của user đó và tiếp tục dò tìm trong tệp mật khẩu.

Nếu tên của user đó tồn tại và mật khẩu tương ứng là đúng, thì home directory của user ấy sẽ được mang ra chia sẻ trên mạng dựa vào các giá trị trong đoạn [homes].

```
[homes]
comment = Home Directories
path = %H
read only = No
valid users = %S
only users = yes
browseable = no
writable = yes
create mask = 0775
```

```
directory mask = 0775
```

Mục comment hiển thị cho client biết những phần nào họ được dùng chung (phần chia sẻ). Mục browseable bảo Samba cách hiển thị phần chia sẻ trên danh sách duyệt mạng. Mục read-only kiểm tra xem một user nào đó có quyền tạo ra và thay đổi tệp trong home directory của mình khi home directory ấy đang được chia sẻ trên mạng. Mục create mask sẽ kiểm tra xem trong số những tệp được tạo ra trong thư mục chia sẻ, thì tệp nào được phép (quyền hạn) gì.

15.2.3 Đoạn [printers]

```
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
# Set public = yes to allow user 'guest account' to print
public = no
guest ok = no
writable = no
printable = yes
create mask = 0700
```

Đoạn [printers] cho biết các dịch vụ in sẽ được kiểm tra ra sao trong trường hợp không có mục tương ứng trong tệp smb.conf. Giống như ở đoạn [homes], nếu không có mục đặc biệt nào, Samba dùng đoạn [printers] để cho phép user kết nối với các máy in được xác định trong tệp /etc/printcap.

Các mục ghi comment, browseable và create mask cũng cùng ý nghĩa với những gì đã giải thích tại đoạn [homes]. Mục path cho biết vị trí của tệp spool để sử dụng khi cung cấp dịch vụ in qua SMB.

Xem "Chọn máy in làm việc với Linux", Chương 20.

Nếu được bật sang "yes", giá trị printable cho biết máy in sử dụng được. Mục public cho biết trương khoản "guest" có được phép in hay không (theo phiên bản mới bạn thay bằng mục guest ok = '')

15.2.4 Chia sẻ thư mục

Sau khi lập cấu hình mặc định cho server Samba, bạn có thể tạo ra nhiều thư mục dùng chung (thư mục chia sẻ) và quyết định xem cá nhân nào, hoặc nhóm nào được phép sử dụng chúng.

Thí dụ bạn muốn thư mục pladir chỉ dành riêng cho user lan_anh mà thôi. Bạn cần viết ra một đoạn mới và ghi các thông tin cần thiết vào: khai báo user, đường dẫn đến thư mục, cùng với thông tin cấu hình cho server SMB như sau:

```
[pladir]
comment = Pla's remote source code directory
path = /usr/local/src
```

```
valid users = lan_anh
browsable = yes
public = no
writable = yes
create mask = 0700
```

Đoạn trên đây đã tạo ra một thư mục chia sẻ mang tên plasdir. Đường dẫn đến thư mục này trên server tại chỗ là /usr/local/src. Vì mục browseable được khai báo "yes", danh sách duyệt mạng sẽ có tên là plasdir. Nhưng vì mục public lại là "no" nên chỉ có user tên là lan_anh mới có quyền dùng Samba để vào ra thư mục. Muốn cho ai được truy cập, bạn chỉ cần liệt kê họ tại thư mục valid users.

15.2.5 Kiểm nghiệm tệp smb.conf

Sau khi tạo ra tệp cấu hình, bạn nên dùng chương trình testparm để kiểm tra lại xem tệp có đúng đắn hay không. Nếu testparm báo cáo không vấn đề gì, bạn có thể yên tâm rằng smbd sẽ nạp tệp cấu hình một cách đúng đắn.

Ghi chú: Chương trình testparm không bảo đảm rằng những dịch vụ đã khai báo trong tệp cấu hình sẽ sẵn có trên mạng hoặc sẽ chạy như mong muốn.

Dùng lệnh testparm như sau:

```
testparm [tệp_cấu_hình][hostname hostip]
```

Với tệp_cấu_hình là vị trí thực sự của tệp smb.conf nếu tệp ấy không nằm ở vị trí mặc định (/etc/samba/smb.conf). Tham số hostname hostIP bắt testparm kiểm tra xem máy chủ có quyền truy cập các dịch vụ được khẳng định trong tệp smb.conf hay không.

Thí dụ sau đây trình bày những gì máy sẽ hiển thị sau khi chạy lệnh testparm. Nếu tệp smb.conf có lỗi, chương trình sẽ cho biết đó là những lỗi gì.

```
[tl@submail t1]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[home]"
Processing section "[printers]"
Loaded services file OK
Press enter to see a dump of your service definitions
```

Khi bạn bấm phím <Enter>, testparm sẽ bắt dầu đánh giá từng đoạn được khai báo trong tệp cấu hình.

15.3 Chạy server Samba

Server Samba gồm hai daemon: smbd và nmbd. Deamon smbd cung cấp tệp và các dịch vụ chia sẻ in ấn. Deamon nmbd hỗ trợ NetBIOS name service.

Bạn có thể chạy server Samba hoặc từ các script init được mô tả chi tiết ở Chương 9, hoặc từ inetd như là một dịch vụ hệ thống.

Xem: "Tìm hiểu tiến trình khởi động"

Vì cả Red Hat và Caldera đều khởi động các dịch vụ SMB từ những script init thay vì dùng inetd làm dịch vụ, bạn có thể sử dụng mẫu lệnh sau đây để khởi động hoặc dừng hẳn server SMB.

```
/etc/rc.d/init.d/smb {start|stop|restart|status|condrestart}
```

15.4 Sử dụng smbclient

Chương trình smbclient cho phép các user của Linux truy cập những phần chia sẻ SMB trên máy khác, nhất là các máy chạy Windows. Vì muốn dùng các tệp trên những máy Linux khác, bạn có thể sử dụng nhiều phương pháp như FTP, NFS và các lệnh r- (như rcp).

smbclient có một giao diện giống như FTP để bạn chuyển tệp sang một máy khác đang dùng server SMB. Điều đáng tiếc là smbclient không làm được như NFS, nghĩa là không cho phép bạn **mount** một chia sẻ khác vào thư mục tại chỗ.

Để **mount** một thư mục cho dịch vụ Samba, bạn thường dùng smb**mount** hơn là dùng **mount** -t smbfs.

smbclient cung cấp các tuỳ chọn dòng lệnh để yêu cầu server chia sẻ thư mục hoặc trao đổi tệp. Để biết thêm thông tin về các tuỳ chọn này, mời bạn xem trang **man** dành cho smbclient. Lệnh sau đây sẽ hiển thị mọi phần chia sẻ trên máy win.netwharf.com:

```
smbclient - L -I win.netwharf.com
```

Tham số -L hiển thị danh sách.

Tham số -I bắt smbclient xử lý máy win.netwharf.com như là một mục ghi DNS chứ không phải mục NetBIOS.

Muốn chuyển tệp, trước tiên bạn phải kết nối với server Samba bằng lệnh:

```
smbclient '\\WORKGROUP\PUBLIC' -I win.netwharf.com -U lan_anh
```

Tham số '\\WORKGROUP\PUBLIC' xác định dịch vụ từ xa trên máy phía bên kia. Thông thường, đó là một thư mục hệ thống tệp hoặc một máy in. Tuỳ chọn -U giúp bạn xác định username mà bạn muốn kết nối. Samba sẽ hỏi bạn mật khẩu (nếu trương khoản ấy đòi mật khẩu), sau đó sẽ hiện ra dấu nhắc.

```
smb:\
```

với \ là thư mục hiện hành.

Từ dòng lệnh này, bạn có thể ra bất kỳ lệnh nào được liệt kê ở bảng sau.

Bảng 15.1: Các lệnh smbclient

Lệnh	Tham số	Mô tả
? hoặc help	[lệnh]	Hiển thị thông báo trợ giúp tương ứng với lệnh, hoặc trong trường hợp không có lệnh thì cho thông báo trợ giúp tổng quát.
!	[lệnh dạng shell]	Thực thi lệnh shell hoặc đưa user về dấu nhắc shell.
cd	[thư mục]	Chuyển về thư mục trên server (chứ không

		phải trên máy tại chỗ). Nếu thư mục không được xác định, smbclient sẽ báo thư mục hiện hành.
lcd	[thư mục]	Chuyển về thư mục trên máy tại chỗ. Nếu thư mục không được xác định, smbclient sẽ báo thư mục hiện hành trên máy tại chỗ.
del	[các tệp]	Những tệp được khai báo sẽ bị xoá khỏi server, nếu user được quyền thực hiện thao tác này. Có thể dùng ký tự wildcard.
dir hoặc ls	[các tệp]	Liệt kê các tệp được chọn. Có thể dùng lệnh ls để có danh sách các tệp.
exit hoặc quit	không có	Thoát khỏi chương trình smbclient.
get	[thư mục][tên tại chỗ]	Truy cập tệp từ xa và sao lưu vào server tại chỗ. Nếu có tên tại chỗ, tệp sẽ được sao lưu với chính tên tại chỗ, thay vì sao lưu với tên trên server từ xa.
mget	[các tệp]	Sao chép mọi tệp được xác định vào máy tại chỗ.
md hoặc mkdir	[thư mục]	Tạo thư mục trên máy từ xa.
rd hoặc rmdir	[thư mục]	Xoá thư mục trên máy từ xa.
put	[tệp]	Sao chép tệp từ máy tại chỗ vào server.
mput	[các tệp]	Sao chép mọi tệp từ máy tại chỗ vào server.
print	[tệp]	In tệp trên máy từ xa.
queue	không có	Liệt kê mọi công việc in ấn đang xếp hàng chờ trên server từ xa.

15.5 Samba trong môi trường X

Trong môi trường đồ hoạ Windows, bạn có thể truy cập vào các thư mục của Samba trên máy Linux thông qua Network Neighborhood hay Windows Explorer.

Trong môi trường Linux dạng văn bản, bạn sử dụng smbclient (xem mục 15.4)

Trong môi trường X (GNOME hay KDE), bạn có thể gọi trình duyệt Nautilus. Với GNOME, bạn chọn Main Menu Button => Programs => Applications => Nautilus. Khi cửa sổ của trình Nautilus hiện ra, bạn gõ smb: tại thanh địa chỉ (location toolbar))

Trên cửa sổ bạn sẽ thấy nhiều biểu tượng mà mỗi biểu tượng biểu hiện cho một SMB workgroup trên mạng. Bạn chỉ cần bấm đúp lên biểu tượng thích hợp để truy cập.

Minh hoạ 15.1: Truy cập thư mục chia sẻ bởi Samba.

Nếu thư mục đó đòi phải đăng nhập như một user với mật khẩu, bạn cần nhập vào thanh địa chỉ theo mẫu sau:

```
smb://user:password@servername/sharename/
```

Bạn cần thay user, password, servername, sharename bằng các giá trị tương ứng.

15.6 Mã hoá mật khẩu với Windows NT/2000

Nguyên thuỷ, giao thức SMB của Microsoft chỉ sử dụng mật khẩu tường minh (plain text password). Tuy nhiên, Windows NT 4.0 (Service Pack 3 trở lên) và Windows 2000 lại yêu cầu mật khẩu Samba được mã hoá (encrypted Samba password). Do đó:

A. Hoặc là bạn chỉnh lại registry của Windows để sử dụng mật khẩu tường minh, hơn nữa, bạn cần chỉnh lại tất cả registry của mọi máy Windows và điều này có nguy cơ đem lại một số xung đột và có thể sai sót.

Ghi chú: Để biết cách chỉnh trong registry của Windows, bạn xem thêm về Windows 2000, Windows NT Policy hoặc xem các tệp ENCRYPTION.txt, WinNT.txt và Win95.txt trong thư mục /usr/share/doc/samba-version-number/docs/textdocs

B. Hoặc là bạn cấu hình lại cho Samba trong Linux chấp nhận mật khẩu mã hoá. Bạn cần làm theo các bước sau:

Tạo một tệp mật khẩu riêng cho Samba. Từ tệp /etc/passwd có sẵn, tạo một tệp mới bằng cách dùng lệnh:

```
cat /etc/passwd | mksmbpasswd.sh > /etc/samba/smbpasswd
```

Với mksmbpasswd là một script đã cài sẵn trong hệ thống. Dùng lệnh:

```
chmod 600 /etc/samba/smbpasswd
```

để chỉ cấp quyền đọc và ghi cho root.

Một user chưa được sử dụng Samba khi user đó chưa được cấp mật khẩu và ghi vào tệp trên. Bạn dùng lệnh dưới đây để cấp password cho user:

```
smbpasswd username
```

Với username là định danh của user đó.

Chỉnh lại trong tệp cấu hình (samba.conf) như encrypt password :

```
smb passwd file = /etc/samba/smbpasswd
```

Để các thay đổi có tác dụng, cần gõ tại dấu nhắc lệnh (shell prompt) dòng lệnh sau để khởi động lại Samba:

```
#service smb restart
```

hay:

```
#/etc/rc.d/init.d/smb restart
```

Chương 16. Hệ thống tệp Linux

Cụm từ Linux File System (hệ thống tệp Linux) có hai nghĩa khác nhau:

1- hệ thống tệp vật lý nằm trong các ổ đĩa (cứng, mềm, CD v.v.) của máy tính

2- hệ thống tệp logic mà user nhìn thấy và thao tác.

Chúng ta sẽ chỉ bàn ở đây về hệ thống tệp logic của Linux. Nếu đã biết hệ điều hành DOS, bạn sẽ thấy nhiều chủ đề trong chương này rất quen thuộc bởi vì cấu trúc tệp của DOS (từ phiên bản 2.0 trở về sau) đều dựa theo khuôn mẫu của UNIX và cấu trúc này cũng áp dụng cho các tệp của Linux.

Mỗi thực thể vật lý hay logic trong Linux đều được coi như một tệp trong hệ thống tệp Linux. Các thực thể vật lý bao gồm các loại đĩa, máy in và terminal. Các thực thể logic bao gồm các thư mục và các tệp vẫn dùng để chứa dữ liệu và chương trình.

Các chủ đề chính sẽ được đề cập đến trong chương này bao gồm:

- *Tên tệp và tên đường dẫn*
- *Các thư mục chuẩn của Linux.*

16.1 Tên tệp và tên đường dẫn

Trong Linux cũng như trong các hệ điều hành khác như DOS, bạn phải phân biệt tên tệp và tên đường dẫn.

Tên tệp là một chuỗi ký tự bao gồm những chữ, số và dấu phân cách đi liền nhau.

Tên tệp được phép chứa ký tự trống, hoặc bất kỳ ký tự nào dùng để thay mặt cho một dấu phân cách các trường với nhau. Thí dụ tên tệp "lan_anh.letter" và "lan_anh letter" đều hợp lệ.

Tên tệp không được chứa bất kỳ ký tự nào mang ý nghĩa đặc biệt với các shell của hệ điều hành. Thí dụ bạn không được dùng dấu sổ ngược "/" là một ký tự bị cấm trong tên tệp, bởi vì theo quy ước thì dấu này dùng để chỉ tên thư mục gốc và thường có mặt trong các tên đường dẫn tuyệt đối (xem ở dưới).

Ghi chú: Thực ra bạn vẫn có thể dùng bất kỳ "ký tự bị cấm" nào, nếu chúng được bao bọc trong dấu ngoặc đơn (trong một số trường hợp có thể dùng ngoặc kép), thí dụ như: "!lan_anh.letter". Tuy nhiên, phần lớn các chương trình sẽ trục trặc khi gặp phải những tên tệp như thế và một số các hệ UNIX khác cũng không chấp nhận.

Hầu hết các phiên bản đầu tiên của UNIX (mà Linux lấy làm cơ sở) chỉ chấp nhận tên tệp dài nhất là 14 ký tự. Tuy nhiên Linux chấp nhận tên tệp dài đến 256 ký tự. Một vài phiên bản UNIX gần đây như BSD chấp nhận 64 ký tự, song chỉ 14 ký tự đầu là có ý nghĩa. Vì một trong những mục đích của Linux là tính hỗ trợ và vì để cho việc biên soạn chương trình hoặc các shell script được chấp nhận rộng rãi (với cả UNIX), bạn nên hạn chế tên tệp ở mức 14 ký tự. Song tên đường dẫn có thể chứa bao nhiêu ký tự cũng được.

Với Linux cũng như với nhiều hệ điều hành khác, tệp phải nằm trong thư mục. Thư mục cấp cao nhất của Linux gọi là root (gốc) và được tượng trưng bằng dấu sổ ngược (/). Nếu tệp lan_anh có mặt trong thư mục gốc, tên đường dẫn tuyệt đối của tệp này sẽ là /lan_anh.

Khi quản trị viên dùng lệnh **adduser** để thêm một user vào hệ thống, user ấy được giao một home directory. Theo quy ước, home directory này nằm trong một thư mục cũng thường lấy tên là home, nằm dưới thư mục root.

Như vậy, khi một user tên lan_anh được giao cho một thư mục mang tên /home/lan_anh, mọi tệp do lan_anh tạo ra đều nằm trong thư mục đó. Đường dẫn đến một trong các tệp của lan_anh có thể là /home/lan_anh/lan_anh1.file. Tên đầy đủ như thế gọi là đường dẫn tuyệt đối, bởi vì nó thông báo vị trí chính xác của một tệp trong hệ thống tệp.

Còn một loại đường dẫn khác gọi là đường dẫn tương đối, bởi vì nó cho biết vị trí tương đối của tệp so với thư mục hiện hành. Thí dụ nếu lan_anh đang ở trong home directory của mình, thì tên tệp lan_anh2.file cũng là một đường dẫn tương đối, so với thư mục hiện hành. Muốn biết mình đang ở trong thư mục hiện hành nào, bạn dùng lệnh pwd. Ngoài ra bạn cũng có thể kiểm tra nội dung biến môi trường $PWD bằng lệnh **echo** $PWD để biết thư mục hiện hành.

Nhờ đường dẫn tương đối, bạn có thể xác định một tệp ở bất cứ nơi nào trong hệ thống tệp của Linux bằng hai "bí danh" (**alias**) có mặt ở mọi thư mục: 1) dấu chấm (.) chỉ thư mục hiện hành và 2) dấu chấm chấm (..) chỉ thư mục mẹ (DOS cũng tuân theo quy ước này). Riêng thư mục gốc (root) là thư mục duy nhất không có thư mục mẹ.

Thí dụ, nếu muốn dời thư mục /lan_anh sang thư mục hiện hành, bạn dùng lệnh **mv** với đường dẫn tuyệt đối như sau:

```
mv /lan_anh lan_anh
```

hoặc đơn giản hơn bạn có thể thay lệnh trên bằng lệnh sau:

```
mv /lan_anh .
```

Những lệnh Linux khi xử lý các tệp thì phải dựa vào tên đường dẫn. Trong phần lớn trường hợp, tên đường dẫn bạn dùng là tên của tệp trong thư mục hiện hành, vì vậy tên đường dẫn cũng mặc định chỉ về thư mục hiện hành. Nếu user lan_anh đang ở trong thư mục /home/lan_anh thì ba lệnh command sau đây sẽ có cùng một hiệu quả:

```
command lan_anh1.letter
command /home/lan_anh/lan_anh1.letter
command ./lan_anh1.letter
```

Ghi chú: Mặc dù tên tệp và tên đường dẫn là hai thứ khác nhau, song thư mục cũng là một loại tệp. Khi đặt tên cho thư mục, bạn cần làm theo quy định đối với tên tệp thông dụng.

Ngoài ra cũng nên nhớ rằng, khác với nhiều hệ điều hành trên máy PC, Linux không dùng mẫu tự đặt tên cho ổ cứng, mà chỉ dùng tên đường dẫn thư mục. Linux chỉ xử lý các mẫu tự đại diện cho ổ cứng khi làm việc với các hệ thống tệp của DOS trên những đĩa mềm với các lệnh kiểu m- (chẳng hạn như **mcopy**).

Xem "Tìm hiểu hệ thống tệp", chương 14.

16.1.1 Phân loại các tệp

Linux có bốn loại tệp cơ bản: tệp thông thường, thư mục, kết nối và tệp đặc biệt. Mỗi loại như thế lại có nhiều kiểu, chưa kể rất nhiều thư mục chuẩn. Chúng sẽ được mô tả ở các mục sau.

Bạn có thể dùng lệnh file để xác định kiểu của một tệp như tệp khả thi, tệp văn bản, hay một tệp có khuôn dạng dữ liệu khác v.v... Nhiều lệnh UNIX đơn thuần chỉ là shell script hoặc những chương trình giống như tệp bó (**batch** file) của DOS. Lệnh file có thể cho bạn biết đó là tệp nhị phân hay là shell script. Ngoài ra bạn cũng cần biết xem tệp có phải đang ở dạng văn bản hay không, nghĩa là có thể xem và sửa hay không. Cú pháp lệnh file như sau:

```
file [-vczL] [-f tệp_tên] [-m tệp_magic] danh_sách_tệp
```

Bảng 16.1: Các đối số của lệnh file

Đối số	**Mô tả**
-c	Hiển thị hình thức phân tích của tệp magic (/usr/lib/magic), vốn là một con số đặc biệt trong phần đầu của một tệp nhị phân giúp nhận dạng loại tệp. Thông thường được dùng với -m để gỡ lỗi một tệp magic mới trước khi cài đặt
-z	Xem xét nội dung tệp nén để đoán ra loại tệp.
-L	Bắt chương trình đi theo một kết nối tượng trưng
-f tệp_tên	Cho biết rằng danh sách các tệp cần được nhận dạng đang nằm trong một tệp văn bản có tên là tệp_tên. Có ích khi cần nhận dạng nhiều tệp
-m tệp_magic	Xác định một tệp magic khác, gồm những số magic, dùng để nhận dạng loại tệp. Tệp mặc định là: /usr/lib/magic
Danh_sách_tệp	Danh_sách_tệp là một chuỗi tên các tệp (cách nhau bằng ký tự giới hạn) mà bạn đang muốn biết chúng thuộc loại gì

16.1.2 Các tệp thông thường

Các tệp thông thường là những tệp mà bạn thường thao tác nhất.

Nội dung của tệp thông thường có thể chứa văn bản, mã nguồn ngôn ngữ C, chương trình khả thi, shell script (tức một tệp văn bản bao gồm tên các lệnh hoặc chương trình khả thi có đối số và được diễn dịch bằng một trong các shell của Linux) và nhiều kiểu dạng dữ liệu khác.

Cho dù nội dung các tệp có tên trong đối số của một lệnh là gì đi nữa thì khi xử lý Linux đều xem mọi tệp như nhau và chỉ chú ý một đặc điểm duy nhất là những tệp đó có thuộc loại khả thi hay không.

Các tệp khả thi là những chương trình dạng nhị phân có thể được thi hành trực tiếp, đương nhiên khi chúng có chứa những mã lệnh cần phải thi hành và lại nằm ngay

trong đường tìm kiếm (search path) của bạn. Về cơ bản, đường tìm kiếm là danh sách các đường dẫn mà Linux phải xem xét khi cần tìm một tệp khả thi.

Xem "Tìm hiểu về shell"

Lệnh file xem xét các dữ liệu trong tệp rồi cho biết những thông tin cần thiết, tuỳ theo các đối số trong bảng nói trên. Thí dụ nếu bạn gõ file * (dấu sao nghĩa là mọi tên tệp) thì máy sẽ hiển thị đại loại như trong khung dưới đây, trong đó tất cả các tệp có tên ở cột thứ nhất đều là những tệp thông thường nhưng chứa nhiều thứ dữ liệu khác nhau và mọi tệp đều đang ở trong thư mục nơi lệnh file được thi hành:

```
a2ps.cfg:   ASCII English text
a3ps0-site.cfg:   ASCII English text
adjtime:    ASCII text
alchemist:  directory
aliases:    ASCII English text
aliases.db: Berkeley DB (Hash, version 7, native byte-order)
amanda:     directory
amandates:  empty
amd.conf:   ASCII text
amd.net:    ASCII text
anacrontab: ASCII text
at.deny:    ASCII text
auto.master:      ASCII English text
auto.misc:  ASCII English text
bashrc:     ASCII text
cdrecord.conf:    ASCII English text
cipe:       directory
conf.linuxconf:   ASCII text
dico:       directory
```

16.1.3 Các tệp thư mục

Thư mục thực chất là một tệp văn bản chứa tên của các tệp và của các thư mục cấp dưới, cùng với tên của các đường liên kết đến những tệp và thư mục cấp dưới ấy. Trong toàn bộ hệ thống, Linux chỉ chứa tên của mọi tệp trong những tệp thư mục. Khi dùng lệnh **ls** để liệt kê nội dung một thư mục, bạn chỉ hiển thị nội dung của tệp thư mục tương ứng mà không đụng chạm gì đến bản thân các tệp của nó.

Khi dùng lệnh **mv** để đặt lại tên cho một tệp (và tệp ấy đang ở thư mục hiện hành), tức là bạn đang thay đổi mục ghi trong tệp thư mục. Khi dời một tệp từ thư mục này sang thư mục khác, tức là bạn thay đổi sự mô tả của tệp đó từ một tệp thư mục này sang một tệp thư mục khác - đương nhiên với điều kiện là thư mục đích phải nằm trên cùng đĩa hoặc phân vùng vật lý, nếu không Linux sẽ chép lại từng byte sang đĩa khác.

16.1.4 Thư mục và đĩa vật lý

Linux chỉ định cho mỗi tệp trong hệ thống một con số duy nhất gọi là inode, được lưu trong một bảng gọi là bảng inode. Bảng này được thành lập khi bạn định dạng đĩa. Mỗi đĩa hoặc phân vùng vật lý đều có bảng inode của riêng mình. Inode chứa tất cả thông tin về tệp, bao gồm địa chỉ của dữ liệu trên đĩa và loại tệp Hệ thống tệp của Linux chỉ định inode 1 cho thư mục /proc. Tệp này chứa danh sách tên mọi tệp và thư mục, vùng với các số inode tương ứng. Linux có thể tìm ra bất kỳ tệp nào trong hệ thống khi xem xét một chuỗi thư mục, bắt đầu là thư mục gốc mà nội dung có dạng thí dụ như sau:

inode	**Tệp**
2	.
2	..
12	.autofsk
1234856	.automount
392474	bin
65409	dev
245291	etc
833985	home
981152	initrd
1193728	lib
11	lost+found
278597	misc
1308187	mnt
1553472	opt
1	proc
392476	root

Hai tệp (.) và (..) có mặt trong mọi thư mục. Bảng trên là thư mục gốc, nên thư mục hiện hành (.) và thư mục mẹ (..) như nhau. Nội dung của /home sẽ khác, thí dụ:

833985	.
2	..
245382	a2ps.cfg
245381	a2ps-site.cfg
245366	adjtime
245361	aliases
245364	aliases.db

Trong bảng trên, inode của thư mục hiện hành (.) khớp với inode của /home trong tệp thư mục gốc root và inode của thư mục mẹ (..) cũng giống như inode của thư mục root.

Linux điều hướng hệ thống tệp của mình bằng cách "móc xích" hệ thống các tệp thư mục. Nếu bạn muốn dời một tệp sang thư mục nằm trên đĩa vật lý khác, Linux sẽ đọc bảng inode. Trong trường hợp này tệp được dời thực sự sang đĩa đích và được giao cho một inode mới trước khi inode cũ của nó ở đĩa nguồn được xoá bỏ.

Cũng như với lệnh **mv**, khi bạn xoá một tệp bằng lệnh rm, bạn chưa hề đụng đến bản thân tệp. Trên thực tế Linux đánh dấu rằng inode ấy giờ đã trống và trả nó về nhóm những inode sẵn sàng được sử dụng. Mục ghi của tệp trong thư mục được xoá đi.

Xem "Di dời và đặt lại tên tệp"

16.1.5 Các (tệp) kết nối

Các kết nối thông thường (kết nối cứng) không thực sự là những tệp, chúng là các mục ghi thư mục trỏ về cùng một inode. Bảng inode theo dõi xem một tệp có bao nhiêu kết nối và chỉ khi nào kết nối cuối cùng với một thư mục được xoá đi, thì inode ấy mới được đánh dấu là trống. Hiển nhiên là các kết nối thông thường không thể nào vượt qua giới hạn của thiết bị, bởi vì tất các mục tham khảo của thư mục đều trỏ về cùng một inode.

Muốn tạo ra kết nối, bạn dùng lệnh ln như sau:

```
ln [tuỳ_chọn] nguồn đích
```

Thí dụ để tạo kết nối giữa tệp mainfile.txt và tệp tempfile.txt, bạn ra lệnh:

```
ln mainfile.txt tempfile.txt
```

Cũng như đa số các phiên bản hiện đại của UNIX, Linux còn một loại kết nối khác gọi là kết nối tượng trưng (kết nối mềm). Đối với loại kết nối này, mục ghi thư mục chứa inode của một tệp, mà bản thân tệp này lại trỏ về một tệp khác nữa nằm trong hệ thống logic của hệ thống tệp. Một kết nối tượng trưng có thể trỏ về một tệp khác hoặc thư mục khác trên cùng một đĩa, hoặc trên một đĩa khác, hoặc trên một máy khác.

Điểm khác biệt chủ yếu giữa kết nối thông thường và kết nối tượng trưng là với kết nối thông thường, mỗi kết nối đều bình đẳng (nghĩa là hệ thống xử lý từng kết nối như thể đó là tệp nguyên thuỷ) và dữ liệu hiện hành chỉ được xoá khi nào kết nối sau cùng với tệp ấy đã được xoá bỏ xong. Với kết nối tượng trưng, khi tệp nguyên thuỷ bị xoá thì dữ liệu của mọi kết nối tượng trưng với tệp ấy liền bị xoá theo, mặc dù các tệp này vẫn tồn tại. Các tệp được kết nối tượng trưng không được hưởng tiêu chuẩn của một tệp nguyên thuỷ.

Muốn tạo kết nối tượng trưng bạn dùng tuỳ chọn -s với lệnh ln. Thí dụ để tạo kết nối tượng trưng giữa tệp /etc/rc.d/inid/named với tệp slnamed trong thư mục hiện hành, bạn gõ như sau:

```
ln -s /etc/rc.d/initd/named slnamed
```

Linux xử lý và truy cập các kết nối giống như đối với tệp, chứ không phân biệt những điểm dị biệt nhỏ giữa kết nối và tệp. Muốn biết tệp có phải là một kết nối không, bạn

dùng lệnh **ls** -l. Nếu phải, máy sẽ hiển thị tên tệp tại chỗ và tiếp theo đó cho biết là tệp được kết nối thí dụ như sau:

```
l rwxrwxrwx 1 root root 4 Oct 17 15:27 Info -> /info/r_info
```

Quyền hạn của tệp này bắt đầu bằng flag l, cho biết đây là tệp kết nối.

16.1.6 Các tệp đặc biệt

Mọi thiết bị ngoại vi như các đĩa, terminal và máy in, là những tệp đặc biệt trong hệ thống tệp Linux. Hầu hết chúng đều nằm trong thư mục /dev.

Thí dụ nếu bạn đang làm việc với terminal console hệ thống, thiết bị ngoại vi này sẽ mang tên /dev/tty01. Các terminal, hoặc cổng nối tiếp, được gọi là những thiết bị tty (viết tắt từ chữ teletype, vốn là tên tiếng Anh của những terminal UNIX nguyên thuỷ). Muốn xác định tên của thiết bị tty hiện hành, bạn gõ lệnh tty. Máy in và terminal được gọi là các thiết bị ngoại vi kiểu ký tự. Chúng có khả năng tiếp nhận và xuất ra hàng loạt ký tự.

Các thiết bị ngoại vi kiểu mảng như đĩa lại lưu trữ dữ liệu thành từng mảng (block), với địa chỉ được khai báo theo số hiệu trụ (cylinder) và cung (sector) của đĩa. Khi truy cập đĩa, bạn không thể chỉ đọc mỗi một ký tự, mà phải đọc hoặc viết cả một mảng. Rắc rối hơn, đĩa và các thiết bị đặc biệt kiểu mảng khác (băng từ) phải có khả năng hành động như là thiết bị đặc biệt kiểu ký tự, do đó mỗi thiết bị kiểu mảng phải khớp với một thiết bị kiểu ký tự. Linux giải quyết việc đó bằng cách đọc những dữ liệu từ thiết bị kiểu ký tự rồi phiên dịch cho thiết bị kiểu mảng.

Còn có ít nhất một loại tệp đặc biệt nữa: vùng đệm FIFO (vào trước ra trước), còn gọi là named pipe (ống dẫn có tên). FIFO khá giống như tệp thông thường, nếu viết vào đấy, kích thước của nó sẽ tăng lên; tuy nhiên khi đọc nội dung thì nó co lại. FIFO được hệ thống dùng để điều tiết việc nhiều chương trình cùng lúc gửi thông tin về một tiến trình xử lý duy nhất. Thí dụ khi bạn ra lệnh **lp** để in, **lp** thiết lập tiến trình in ấn và báo hiệu cho daemon lpshed bằng cách gửi thông báo đến FIFO (daemon là một tiến trình hệ thống có thể được cấu hình để chạy tự động mà không cần user yêu cầu).

Còn một tệp đặc biệt nữa gọi là bit bucket (thùng đựng bit) hay /dev/null. Tệp này rất có ích bởi vì bất cứ những gì bạn ném vào đấy đều bị lãng quên và máy không quan tâm tới. Thí dụ nếu bạn không muốn nhìn thấy các báo cáo trên thiết bị báo lỗi chuẩn (stderr), bạn có thể ném chúng vào thùng bit bằng lệnh sau:

```
ls -la 2 > /dev/null
```

16.1.7 Quyền hạn của tệp

Trong hệ thống Linux, quyền hạn của tệp (file permission) còn quan trọng hơn là quyền hạn thông thường mà bạn có đối với tệp hoặc thư mục. Ngoài việc xác định xem ai có quyền đọc, ghi, hoặc thi hành một tệp, nó còn xác định loại tệp và cách mà tệp được thi hành.

Lệnh **ls** -l giúp bạn hiển thị quyền hạn chi tiết của một tệp, thí dụ như sau:

```
drwx ------ 2 offices   doc   512         Jan   1     13:44 Mail
drwx ------ 5 offices   doc   1024        Jan   17    08:22 News
```

```
-rw-------- 1 offices   doc   1268        Dec   7     15:01 biblio
drwx ------ 2 offices   doc   512         Dec   15    21:28 bin
-rw-------- 1 offices   doc   44787       Oct   20    06:59 books
-rw-------- 1 offices   doc   22801       Dec   14    22:50 bots.msg
-rw-------- 1 offices   doc   105990      Dec   27    21:24 duckie.gif
```

Liệt kê trên đây cho biết gần như là tất cả thông tin về các tệp:

Cột thứ nhất cho biết quyền hạn và cờ chỉ thị loại tệp.

Cột thứ hai là số kết nối đến một tệp (còn gọi là block phụ thêm trong thư mục).

Cột thứ ba chỉ chủ sở hữu (Linux có ba khả năng sở hữu: cá nhân sở hữu, nhóm của các cá nhân sở hữu và những chủ sở hữu khác. Chúng ta sẽ bàn về chủ đề này sau).

Cột thứ tư cho biết tệp thuộc nhóm nào.

Cột thứ năm là kích thước tệp.

Cột thứ sáu chỉ ngày giờ tạo ra tệp.

Cột thứ bảy là tên tệp.

Trường quyền hạn (cột thứ nhất) được chia chi tiết thành bốn trường con như sau:

```
-rwx rwx rwx
```

Trường con thứ nhất xác định loại tệp: tệp thông thường mang dấu gạch nối (-) như là một mốc giữ chỗ (placeholder), còn thư mục được đánh dấu bằng d. Bảng sau trình bày các giá trị của trường con xác định loại tệp.

Bảng16.2: Các mục ghi hợp lệ cho trường con xác định loại tệp

Ký tự	**Ý nghĩa**
-	Tệp thông thường
b	Tệp đặc biệt kiểu mảng
c	Tệp đặc biệt kiểu ký tự
d	Thư mục
l	Liên kết

Ba trường con tiếp theo cho biết quyền đọc, ghi và thi hành của tệp. Thí dụ cụm rwx trong trường con thứ nhất có nghĩa là chủ sở hữu tệp có quyền đọc, ghi và thi hành tệp ấy. Cụm ba ký tự tiếp theo cung cấp cùng thông tin đối với nhóm sở hữu tệp. Cuối cùng, cụm ký tự thứ ba cho biết quyền hạn đối với những người khác.

Các trường con này còn mang thông tin khác nữa, vì thực ra nhiều thuộc tính được bao hàm trong ba trường con ấy. Tuy nhiên, ý nghĩa của các thuộc tính tuỳ thuộc vào từng phiên bản Linux và tuỳ vào việc đó có phải là tệp khả thi hay không.

Ghi chú: Thông thường, một chương trình đang chạy thì thuộc sở hữu của người ra lệnh chạy. Nhưng nếu bit SUID được kích hoạt (bật sang "on"), chương trình đang chạy sẽ thuộc sở hữu của chủ tệp, có nghĩa là chương trình đang chạy ấy có tất cả quyền hạn dành cho chủ sở hữu của tệp. Nếu bạn là một user bình thường và chương

trình đang chạy lại thuộc về superuser (root user), chương trình ấy sẽ tự động có quyền hạn đọc và ghi bất kỳ tệp nào trong hệ thống, mà không cần quan tâm xem bản thân bạn có những quyền hạn nào. Điều này cũng tương tự đối với bit GUID cho nhóm. Đây chính là một kẽ hở bảo mật tiềm tàng.

Bạn có thể lập bit dính (sticky bit) trong những trường con ấy. Bit dính bắt máy lưu một bản sao của chương trình đang chạy vào bộ nhớ đệm (cache) khi chương trình hoàn tất. Nếu chương trình này được chạy thường xuyên, bit dính sẽ giúp máy tiết kiệm thời gian truy cập vào lần sau, bởi vì máy sẽ không phải lấy chương trình từ đĩa rồi nạp vào RAM.

Dùng lệnh chmod, bạn có thể thay đổi quyền hạn của bất kỳ tệp nào mà bạn có quyền ghi. Lệnh này có hai cú pháp: tuyệt đối và tương đối.

A. Cú pháp tuyệt đối. Bạn khẳng định quyền hạn của tệp bằng một con số hệ bát phân (hệ đếm octa, hay cơ sở 8, được dùng vì ban đầu UNIX ra đời trên những máy tính PDP vốn sử dụng hệ bát phân). Mỗi chữ số (đơn vị) bát phân có một giá trị trong khoảng từ 0 đến 7. Các chữ số bát phân được cộng lại với nhau để hình thành con số xác định tổng quyền hạn. Bảng sau liệt kê các quyền hạn bát phân hợp lệ.

Bảng 16.3: Các quyền hạn tuyệt đối dùng với lệnh chmod

Giá trị bát phân	**Quyền hạn**
0001	Quyền hạn thi hành cho cá nhân sở hữu
0002	Quyền hạn ghi cho cá nhân sở hữu
0004	Quyền hạn đọc cho cá nhân sở hữu
0010	Quyền hạn thi hành cho nhóm sở hữu
0020	Quyền hạn ghi cho nhóm sở hữu
0040	Quyền hạn đọc cho nhóm sở hữu
0100	Quyền hạn thi hành cho những chủ sở hữu khác
0200	Quyền hạn ghi cho những chủ sở hữu khác
0400	Quyền hạn đọc cho những chủ sở hữu khác
1000	Bật bit dính lên chế độ "on"
2000	Bit GUID nhóm được bật lên chế độ "on" nếu tệp là dạng thi hành. (bit của GUID)
4000	Bit SUID cho user ở chế độ "on" nếu tệp dạng thi hành (bit của SUID)

ID (chỉ số) của nhóm hoặc ID của user xác định xem ai có quyền sử dụng, đọc, hoặc thi hành một tệp. Các quyền hạn nguyên thuỷ này được quản trị viên hệ thống xác lập khi lần đầu tiên tạo ra trương khoản cho user.

Chỉ một số user nhất định của một nhóm nhất định mới được truy cập các tệp chung của nhóm, với điều kiện là những user ấy có quyền hạn nhóm sở hữu của những tệp

đó. Thí dụ, muốn cấp quyền hạn đọc và ghi một tệp cho tất cả mọi người, trước hết phải cộng các quyền hạn cho các loại chủ sở hữu thành dòng cuối cùng ở bảng sau:

0002	Quyền hạn ghi cho cá nhân sở hữu
0004	Quyền hạn đọc cho cá nhân sở hữu
0020	Quyền hạn ghi cho nhóm sở hữu
0040	Quyền hạn đọc cho nhóm sở hữu
0200	Quyền hạn ghi cho những chủ sở hữu khác
0400	Quyền hạn đọc cho những chủ sở hữu khác
0666	Quyền hạn đọc và ghi cho tất cả mọi người

Rồi bạn dùng lệnh chmod để cấp tổng quyền hạn đó cho tệp cần thiết:

```
chmod 666 tên_tệp
```

B. Cú pháp tương đối. Cú pháp tương đối hơi khác một chút, bạn phải xác định những việc sau:

-Cấp quyền hạn cho ai.

-Định thực hiện thao tác gì (thêm, bớt, hoặc lập quyền hạn).

-Lập quyền hạn nào.

Thí dụ nếu gõ chmod a=rwx file, thì nghĩa là bạn cấp quyền hạn đọc, ghi và thi hành cho mọi user. Bảng dưới đây tổng kết các lệnh cho quyền hạn tương đối.

Bảng16.4: Các quyền hạn tương đối dùng với lệnh chmod.

Giá trị	**Mô tả**
Cấp cho ai ?	
a	Mọi user (user, nhóm user và những chủ sở hữu khác)
g	Nhóm user
o	Những chủ sở hữu khác
u	Chỉ một mình user thôi
Thao tác gì ?	
+	Thêm quyền
-	Bớt quyền
=	Lập quyền một cách tuyệt đối
Quyền hạn nào ?	
x	Lập quyền thi hành
r	Lập quyền đọc
w	Lập quyền ghi

s	Lập bit SUID hay GUID
t	Lập bit dính

Nếu tệp được đánh dấu "on" ở bit SUID của user, lệnh **ls** -l sẽ hiển thị quyền hạn như sau:

```
-rws------ l office 31336 Jan 17 15:42 x
```

Nếu thêm vào bit GUID nhóm, quyền hạn sẽ thay đổi như sau:

```
-rws--s--- l office 31336 Jan 17 15:42 x
```

Và nếu lúc này bạn bật bit dính lên, quyền hạn sẽ biến thành:

```
-rws--s--t l office 31336 Jan 17 15:42 x
```

Hai ký tự **s** và **t** cho biết trạng thái "on" của các bit SUID, GUID và bit dính.

16.2 Các thư mục chuẩn của Linux

Giờ này bạn đã quen với khái niệm thư mục. Khi đăng nhập, hệ thống sẽ đưa bạn vào home directory của bạn. Biến môi trường PATH được lập để trỏ về các thư mục chứa những chương trình khả thi. Những thư mục này là thành phần trong cấu trúc thư mục chuẩn "cổ điển" của Linux được kế thừa từ UNIX. Chúng được mô tả ở những mục tiếp theo sau đây, dù rằng không được áp dụng nguyên xi trong phiên bản Red Hat.

16.2.1 Thư mục UNIX cổ điển

Trước UNIX System V Release 4 (thí dụ UNIX System V Release 3.2 và các bản trước đó), hầu hết các bản UNIX đều có dạng tổ chức thư mục như sau:

```
/
/etc
/lib
/tmp
/bin
/usr
      /spool
      /bin
      /include
      /tmp
      /adm
      /lib
```

Thư mục /etc chứa hầu hết các dữ liệu liên quan đến hệ thống và cần thiết để khởi động máy. Nó chứa các tệp như passwd và inittab, rất cần thiết để hệ thống hoạt động tốt.

Thư mục /lib chứa thư viện các hàm (chức năng) dành cho trình biên dịch C. Ngay cả khi hệ thống của bạn không có trình biên dịch này, thư mục /lib cũng quan trọng bởi vì nó chứa mọi hàm C dùng chung mà nhiều chương trình ứng dụng sẽ truy cập. Thư

viện các hàm C dùng chung (cũng gọi là thư viện chia sẻ) chỉ được nạp vào bộ nhớ khi nào có chương trình gọi nó được thực hiện. Như thế các chương trình khả thi sẽ nhỏ gọn, nếu không chúng lại phải cộng thêm những đoạn mã chức năng giống nhau làm tốn cả đĩa cứng lẫn bộ nhớ.

Thư mục /tmp dùng để lưu trữ tạm thời. Thông thường, chương trình nào dùng đến /tmp sẽ tự động xoá sạch các tệp mà nó tạm lưu trong thư mục này sau khi chạy xong. Và bởi vì thỉnh thoảng hệ thống lại tự động dọn /tmp, bạn đừng lưu những gì quan trọng trong thư mục này.

Thư mục /bin và /sbin chứa mọi tệp khả thi cần thiết để khởi động hệ thống và những lệnh thường dùng nhất của Linux. Tuy nhiên, bạn nên nhớ là tệp khả thi không nhất thiết phải là tệp nhị phân. Một số tệp nhỏ trong /bin thực ra chỉ là shell script.

Thư mục /usr chứa tất cả mọi thứ còn lại. Biến PATH của bạn có chuỗi /bin:/usr/bin bởi vì trong thư mục /usr/bin chứa mọi lệnh Linux nào không có mặt trong thư mục /bin. Việc này có lý do lịch sử của nó. Linux cần ít nhất ba thư mục /etc, /tmp và /bin để tự thi hành. Vào thuở Linux còn sơ khai, đĩa cứng không có dung lượng lớn và chỉ đủ sức chứa ba thư mục ấy mà thôi, tất cả những gì còn lại đều phải đặt trên đĩa khác, đợi sau khi Linux chạy được rồi mới nạp vào. Trước đây khi Linux hãy còn là một hệ điều hành tương đối nhỏ, việc đặt thêm các thư mục con vào thư mục /usr không mấy khó khăn, do đó Linux có thể chạy với hai đĩa: đĩa root và đĩa /usr.

Thư mục /usr/adm chứa mọi thông tin thống kê và chẩn đoán cần thiết cho quản trị viên hệ thống. Hiện nay đó là thư mục /var. Nếu cả hai chương trình thống kê và chẩn đoán đều không được bật lên, thư mục này sẽ trống.

Thư mục /include, hiện nay là /usr/include, chứa mọi đoạn mã nguồn cần thiết cho những biểu thức #include trong các chương trình C. Bạn phải có ít nhất quyền hạn đọc đối với thư mục này bởi vì trong đó có tất cả đoạn mã và cấu trúc hình thành hệ thống của bạn. Bạn không nên thay đổi các tệp trong thư mục này.

Thư mục /usr/spool, nay là /var/spool, chứa thông tin tạm thời, cần thiết cho trình in ấn **lp**, cho daemon **cron** và cho trình liên lạc UUCP. Những tệp nào được "tuồn" (spooled) cho máy in đều nằm tạm trong thư mục spool cho đến khi chúng được in xong. Mọi chương trình chờ để được **cron** điều khiển, kể cả các tệp **crontab**, các tệp **batch** và at đang trong thời gian chờ đợi đều tạm trú ở đây.

Thư mục /usr/lib chứa tất cả những gì còn lại thuộc về hệ thống Linux chuẩn. Nói chung, thư mục này chứa mớ tệp hỗn độn nằm ở hậu trường Linux.

Thư mục /usr/bin chứa tệp cấu hình cho máy in và terminal, chứa các trình được gọi bởi trình khác của /bin và /usr/bin, trình thư tín, **cron** và trình liên lạc UUCP.

Thư mục /usr, nay là thư mục /home chứa mọi thư mục con được cấp cho user. Quy ước chung là nếu ID đăng nhập là "phan_an" thì home directory sẽ là /usr/phan_an.

Cách sắp xếp thư mục như bạn vừa xem qua được hình thành xưa kia khi đĩa cứng vừa nhỏ vừa đắt tiền. Giờ đây khi đĩa cứng có dung lượng lớn và rẻ hơn, bạn có thêm một cách tổ chức Linux tốt hơn sẽ được bàn đến ở mục sau.

16.2.2 Các thư mục trong Linux

Cấu trúc cổ điển của UNIX gặp khó khăn khi phải sao lưu các tệp dữ liệu với thư mục /usr bị phân mảnh. Nhìn chung một hệ thống đòi hỏi phải có ba cấp sao lưu: bản thân hệ thống cơ bản; dữ liệu của user; và bất kỳ thay đổi nào trong những bảng định nghĩa hệ thống cơ bản.

Bạn chỉ nên sao lưu hệ thống cơ bản một lần sau khi có thay đổi trong những bảng điều khiển và những thay đổi này đã được sao lưu trước đó rồi. Dữ liệu của user luôn thay đổi do đó nên sao lưu thường xuyên.

Dưới đây là cấu trúc điển hình của Linux, tuy nhiên cấu trúc trong máy bạn có thể khác đôi chút, tuỳ theo bạn đã cài đặt những gói phần mềm nào. Các thư mục /bin, /etc và /tmp có cùng chức năng như ở cấu trúc cổ điển. Các bảng định nghĩa hệ thống được chuyển sang thư mục /var, nhờ đó mỗi khi trong thao tác của hệ thống có gì thay đổi, bạn chỉ cần sao lưu thư mục này mà thôi. Nét mới là mọi chương trình hệ thống đều chuyển sang thư mục /sbin. Tất cả những chương trình chuẩn Linux đều nằm trong /usr/bin và thư mục này được kết nối với /bin. Để dễ dàng tương thích, mọi thư mục cổ điển đều được duy trì bằng những kết nối tượng trưng. Thư mục /usr, giờ đây không còn chứa dữ liệu user, được tổ chức lại để chứa mớ tệp hỗn độn trước kia nằm trong thư mục /usr/lib.

```
/
/etc
      /passwd (user database)
      /rc.d (system initialization scripts)
/sbin
/bin
/tmp
/var
/lib
/home
/<tên user của bạn ở đây> (trương khoản của user)
/install
/usr
/bin
/sbin
/src
/local
/bin
/sbin
/lib
/proc
```

Chương 17 Quản lý tệp và thư mục

Hầu hết các lệnh Linux (và do đó cả các shell script) đều tập trung vào thao tác trên các tệp và thư mục. Những thao tác này được các shell Linux diễn dịch dễ dàng nhờ một cú pháp thích hợp. Nói chung có thể chia các lệnh ấy thành hai hạng: 1) Những lệnh thao tác bản thân tệp như là đối tượng; 2) Những lệnh chỉ thao tác nội dung tệp.

Chương này cung cấp cho quản trị viên và độc giả quan tâm những lệnh coi bản thân tệp như là đối tượng, nghĩa là những lệnh có chức năng di dời, đổi tên, xoá bỏ, định vị và thay đổi các thuộc tính của tệp và thư mục. Ngoài ra ở đây cũng lướt nhanh qua những lệnh thao tác nội dung tệp.

Các chủ đề chính sẽ được đề cập đến bao gồm:

-Liệt kê tệp

-Tổ chức tệp

-Sao chép tệp

-Di dời và đặt tên lại tệp

-Xoá tệp hoặc thư mục

-Xem nội dung của tệp

-Duyệt tìm tệp

-Thay đổi nhãn ngày giờ

-Nén và nới tệp

17.1 Liệt kê tệp

Lệnh cơ bản để liệt kê tệp là **ls**, hoạt động tuỳ theo cách ra lệnh. Nếu dùng trong ống dẫn (pipe), mỗi tệp sẽ hiển thị trên một dòng như mặc định của vài phiên bản UNIX, chẳng hạn như SCO. Những phiên bản khác liệt kê tệp thành nhiều cột. Hầu hết người sử dụng đều thích hình thức cột. Phiên bản nào mặc định **ls** là lệnh liệt kê thành từng dòng thường có riêng một lệnh khác, chẳng hạn như lc, để liệt kê thành từng cột.

Hành động của lệnh **ls** thay đổi tuỳ theo các cờ (flag) đi kèm, dạng như -abcd. Nhìn chung các phiên bản của lệnh **ls** có hai dạng: dạng xuất phát từ UNIX System V và dạng xuất phát từ UNIX Berkeley. Bởi vì các hệ Berkeley dần dần hội tụ với System V cho nên chương này tập trung phân tích các cờ của System V. Muốn biết mình đang dùng phiên bản **ls** nào, bạn xem tài liệu kèm theo hệ hoặc gõ lệnh **man ls.**

Các cờ được lệnh **ls** sử dụng có thể viết liền hay viết rời, kết quả như nhau. Thí dụ **ls** -l -F và **ls** -lF đều cho cùng một kết quả.

*Bảng 17.1: Các cờ dùng với lệnh **ls***

Cờ	**Mô tả**
-a	Liệt kê mọi tệp vì **ls** mặc định không liệt kê những tệp bắt đầu bằng

	dấu chấm (thường đó là các tệp cấu hình có rất nhiều trong home directory nên sẽ làm rối mắt nếu **ls** lại mặc định là liệt kê mọi tệp)
-A	Như –a, song không liệt kê 2 bí danh "." và "..". Vì tên những tệp này bắt đầu bằng dấu chấm nên -a liệt kê cả chúng
-b	Buộc các ký tự phi đồ hoạ phải xuất hiện ở dạng bát phân \ddd, cờ -b tiện dùng hơn -q vì nó giúp bạn hình dung các ký tự ấy ra như thế nào
-c	Sử dụng thời gian của lần truy cập, chỉnh sửa gần nhất đề phân loại. Linux dán 3 loại nhãn ngày giờ vào tệp: lúc tạo ra tệp, lần truy cập gần nhất và lần chỉnh sửa gần nhất. Các tệp thường liệt kê theo thứ tự abc, nhưng chữ in hoa được xếp trước chữ thường
-d tên_tệp	Nếu đối số là một thư mục, cờ này chỉ liệt kê tên (không liệt kê nội dung); thường dùng với -l để xem trạng thái thư mục. Thường nội dung thư mục chỉ được liệt kê khi nào được khẳng định là phải liệt kê, hoặc dùng chung với wildcard (ký tự đại diện). Do đó lệnh **ls** chỉ liệt kê tên thư mục, trong khi **ls** * lại liệt kê tệp, thư mục và nội dung của bất kỳ thư mục nào có mặt trong thư mục hiện hành
-F	Đánh dấu thư mục bằng dấu chéo /, đánh dấu tệp khả thi bằng dấu sao, các kết nối tượng trưng bằng dấu a vòng @, các FIFO bằng dấu | và các rãnh cắm (socket) bằng dấu -
-i	In các inode tại cột đầu tiên của bản báo cáo. Nếu bạn liệt kê những tệp kết nối, nên lưu ý là chúng đều có cùng số inode.
-l	Liệt kê mục ghi thư mục ở dạng dài, bao gồm cả quyền hạn, số kết nối, chủ sở hữu, kích thước tính bằng byte và ngày giờ chỉnh sửa gần nhất. Nếu đấy là tệp đặc biệt, trường kích thước sẽ chứa số hiệu thiết bị trưởng và thứ. Nếu thời gian chỉnh sửa cách nay hơn sáu tháng, sẽ hiển thị tháng, ngày và năm; ngược lại sẽ chỉ hiện ngày giờ. Nếu tệp là kết nối tượng trưng, tên đường dẫn của tệp đích sẽ được hiển thị sau dấu mũi tên. Bạn có thể phối hợp -l với các tuỳ chọn khác, thí dụ như -n
-n	Thay vì tên, sẽ hiển thị số ID của user và nhóm kết hợp với từng tệp và thư mục. Thông thường **ls** chỉ liệt kê tên. –n rất tiện lợi khi thiết lập mạng, chẳng hạn như TCP/IP, vì bạn cần biết số ID để cấp quyền hạn qua nhiều hệ thống khác nhau đang nối mạng
-q	Hiển thị các ký tự phi đồ hoạ trong tên tệp thành dấu hỏi. Đối với **ls**, đây là hành động mặc định khi xuất ra terminal. Nếu vô tình tệp được tạo ra bằng những ký tự không in được, cờ -q sẽ hiển thị tệp ấy
-r	Đảo ngược thứ tự sắp xếp abc hoặc thời gian
-s	Hiển thị kích thước tệp, gồm cả những block gián tiếp ánh xạ tệp. Nếu có biến môi trường POSIX_CORRECT, mỗi block sẽ là 512 byte
-t	Sắp xếp theo thứ tự thời điểm chỉnh sửa (gần nhất xếp trước) thay vì theo tên. Nếu muốn xem những tệp lâu đời trước, bạn gõ -rt.

-u	Sử dụng thời điểm truy cập gần nhất, thay vì thời điểm thay đổi gần nhất để sắp xếp (với -t) hoặc để in (với –l).
-x	Buộc Linux xuất mục ghi ra thành dạng nhiều cột dàn hàng ngang, thay vì ở cuối trang.

Nếu có bản Linux Slackware, bạn sẽ nhận thấy rằng **ls** cũng hiện các màu khác nhau cho từng loại tệp. Màu được xác định ở tệp cấu hình DIR_COLORS trong thư mục /etc (từ phiên bản 6.1 trở đi, RedHat cũng sử dụng màu khi liệt kê tệp).

Cấu hình mặc định bao gồm:

- Tệp khả thi: màu xanh lá cây (green)

- Thư mục: màu xanh biển (blue)

- Kết nối tượng trưng: màu lục lam (cyan).

Muốn căn chỉnh màu, bạn chép tệp DIR_COLORS về home directory rồi đổi tên thành .dir_colors.

Bảng sau liệt kê các định nghĩa màu khả dụng. Mời bạn tham khảo thêm các trang **man** về tệp DIR_COLORS.

Bảng 17.2: Các giá trị DIR_COLORS thể hiện màu sắc.

Giá trị	**Mô tả**
0	Phục hồi các màu mặc định
1	Cho vùng có màu sáng hơn.
4	Cho vùng văn bản gạch dưới.
5	Cho vùng văn bản nhấp nháy.
30	Cho vùng tiền cảnh màu đen.
31	Cho vùng tiền cảnh màu đỏ.
32	Cho vùng tiền cảnh màu xanh lá cây.
33	Cho vùng tiền cảnh màu vàng (hoặc nâu).
34	Cho vùng tiền cảnh màu xanh biển.
35	Cho vùng tiền cảnh màu đỏ tía.
36	Cho vùng tiền cảnh màu lục lam.
37	Cho vùng tiền cảnh màu trắng hoặc xám.
40	Cho vùng nền màu đen.
41	Cho vùng nền màu đỏ.
42	Cho vùng nền màu xanh lá cây
43	Cho vùng nền màu vàng hoặc nâu.
44	Cho vùng nền màu xanh biển.

45	Cho vùng nền màu đỏ tía.
46	Cho vùng nền màu lục lam.
47	Cho vùng nền màu trắng hoặc xám.

Còn nhiều tuỳ chọn nữa xin xem ở các trang **man** dành cho lệnh **ls.**

17.2 Tổ chức tệp

Linux không có quy định bất biến cho việc tổ chức tệp. Tệp của Linux không có phần mở rộng (thí dụ như phần đuôi .EXE cho những tệp khả thi của DOS). Bạn có thể và bạn nên tổ chức cách đặt tên tệp cho riêng mình.

Xin nhắc bạn rằng ở thuở ban đầu của Linux mỗi tên tệp đều ghi đầy đủ đường dẫn với các thư mục của nó.

Tuy nhiên, phải nhìn nhận việc ngày càng có nhiều ứng dụng Linux đến từ thế giới của DOS và các ứng dụng này mang theo những quy ước của mình đến cho Linux. Giới cung cấp phần mềm thường khuyên bạn nên đặt tên tệp có phần mở rộng và thích hợp với ứng dụng của tệp.

Nếu muốn tạo ra những lệnh riêng của mình, bạn nên tổ chức thư mục dựa theo cách của Linux, với những thư mục /bin, /lib và /etc. Hãy tạo cấu trúc thư mục cấp dưới trong thư mục /home của bạn. Để cho mọi việc được đơn giản, bạn nên đặt các lệnh thi hành trong /bin, các lệnh phụ trong /lib và các tệp cấu hình khởi động trong /etc. Và đây chỉ là một gợi ý chứ không phải là bắt buộc.

Bạn tạo ra thư mục bằng lệnh **mkdir** với cú pháp đơn giản như sau:

```
mkdir tên_thư_mục
```

Với tên_thư_mục là tên đặt cho thư mục mới. Trên nguyên tắc, để tạo ra một thư mục cấp dưới, bạn phải có quyền hạn ghi trong thư mục hiện hành. Tuy nhiên ở đây bạn tạo ra thư mục cấp dưới trong chính /home của bạn, do đó sẽ không gặp rắc rối gì cả.

Giả sử bạn tạo ra ba chương trình mang tên prog1, prog2 và prog3, tất cả đều đặt trong $HOME/bin. Xin nhớ $HOME chính là home directory của bạn.

Nếu muốn những chương trình riêng của mình được chạy như thể chúng là thành phần tiêu chuẩn của bộ lệnh Linux, bạn phải ghi $HOME/bin vào biến môi trường PATH. Để thực hiện việc này, bạn dùng lệnh sau đây trong shell của Bourne và Korn.

```
PATH = $PATH :$HOME/bin; export PATH
```

Với shell C, bạn sẽ gõ như sau:

```
setenv PATH $PATH $HOME/bin
```

Ghi chú: $HOME đại diện cho đường dẫn đầy đủ đến home directory của bạn. Nếu đó là /home/ams, thì $HOME/bin sẽ được viết là /home/ams/bin.

Nếu các chương trình của bạn gọi đến chương trình phụ, bạn nên tạo thư mục cấp dưới trong thư mục $HOME/lib. Bạn có thể tạo ra thư mục cấp dưới cho từng chương trình phụ. Thí dụ lệnh riêng pgm1 có thể gọi $HOME/lib/pgm1a. Tương tự, nếu lệnh prog1

yêu cầu phải có bảng cấu hình khởi động, bạn có thể đặt tên bảng đó là $HOME/etc/pgm1.rc. Dữ liệu của bạn có thể đặt trong thư mục $HOME/data/pgm1.

17.3 Sao chép tệp

Lệnh sao chép tệp có dạng:

```
cp tệp_nguồn tệp_đích
```

Bạn phải có quyền hạn đọc từ tệp_nguồn mà bạn định sao chép và quyền hạn ghi vào thư mục đích (và tệp mà bạn định ghi đè lên phải có thực).

Bạn nên lưu ý một số điểm như sau:

-Nếu tệp_đích là tên của một tệp có sẵn, thì bạn sẽ ghi đè lên tệp có sẵn ấy.

-Nếu gõ tên thư mục đích sau lệnh **cp**, tệp sẽ được chép vào thư mục ấy và vẫn giữ nguyên tên cũ. Thí dụ bạn gõ lệnh:

```
cp tệp_nguồn thư_mục_đích
```

thì tệp sẽ được chép vào thư_mục_đích dưới dạng thư_mục_đích /tên_tệp_nguồn.

Bạn có thể chép một danh sách các tệp1, tệp2, tệp3 vào thư_mục_đích bằng lệnh:

```
cp tệp1, tệp2, tệp3 … thư_mục_đích
```

Nếu thư_mục_đích không phải là một thư mục, máy sẽ báo lỗi. Ngoài thư_mục_đích, nếu bất kỳ thành phần nào của danh sách tệp lại là một thư mục, thì máy cũng báo lỗi.

Hãy cẩn thận khi dùng wildcard (ký tự thay thế) với lệnh **cp**, bởi vì có khả năng rủi ro là bạn sẽ chép nhiều thứ hơn là dự định.

Ghi chú: Những người sử dụng Linux thường để trong máy nhiều tệp dạng DOS, đồng thời làm cho Linux truy cập được hệ thống tệp DOS. Hầu hết các lệnh Linux nhận biết khi nào một phân vùng DOS là đích hoặc nguồn, cho nên trong tiến trình chép tệp, DOS có thể xử lý việc diễn dịch tệp. Thông thường khi mở ra và lưu lại một tệp văn bản của UNIX/Linux trong DOS bằng WordPad, WordPad luôn lưu thêm ký tự về đầu dòng (CR) cạnh ký tự xuống dòng (LF), trong khi các hệ Linux và UNIX chỉ dùng một ký tự LF để báo hiệu về đầu dòng mới.

17.4 Di dời và đặt tên lại tệp

Lệnh **mv** của Linux giúp bạn di dời và đặt lại tên cho cả tệp lẫn thư mục. Cú pháp và quy định của **mv** cũng giống như của **cp**, nghĩa là bạn toàn quyền di dời bao nhiêu tệp cũng được, miễn là cuối lệnh phải có tên thư mục và bạn phải có quyền ghi đối với thư mục đó. Chỉ khác là **cp** không thể di dời và đặt lại tên thư mục.

Khi bạn ra lệnh di dời hoặc đặt lại tên tệp, thực ra chỉ có mục ghi trong tệp thư mục là thay đổi. Ngoại trừ trường hợp vị trí mới nằm trên ổ đĩa hoặc phân vùng vật lý khác, nếu không tệp và nội dung thư mục không di dời gì cả.

Nếu bạn gõ lệnh **rm** (xoá) hoặc **cp** mà không cho thêm tuỳ chọn về thư mục, máy sẽ không thi hành lệnh và báo lỗi rằng bạn đang xử lý một thư mục. Muốn xoá hoặc chép thư mục, bạn phải dùng cờ -r với **rm** và **cp.** Tuy nhiên lệnh **mv** lại di dời thư mục không gặp vấn đề gì.

17.5 Xoá tệp hoặc thư mục

Như bạn đã biết, muốn xoá bỏ tệp hoặc thư mục, bạn dùng lệnh **rm**. Muốn xoá tệp không thuộc quyền sở hữu của mình, bạn phải có quyền hạn ghi trên thư mục chứa tệp cần xoá.

Nếu đang làm chủ tệp, tất nhiên là bạn có thể tuỳ ý xoá bỏ, nhưng với điều kiện là quyền hạn của thư mục chứa tệp đó cho phép bạn ghi.

Lệnh **rm** * sẽ xoá toàn nhóm tệp mà bạn có quyền ghi trong thư mục hiện hành, nhưng thư mục cấp dưới không bị ảnh hưởng. Muốn xoá thư mục cấp dưới, bạn phải dùng tuỳ chọn -r (recursive, đệ quy).

Ở một vài phiên bản, **rm** sẽ tạm dừng để hỏi bạn có thực sự muốn xoá tệp mà bạn có quyền sở hữu nhưng lại không có quyền hạn ghi. Một vài phiên bản khác lại thắc mắc khi bạn gõ lệnh **rm** kèm với wildcard. Đương nhiên bạn có thể viết riêng cho mình một macro hoặc một shell script để tạo cơ hội suy nghĩ trước khi khẳng định xoá tệp.

Nếu phiên bản **rm** đang sử dụng đắn đo khi bạn ra lệnh xoá những tệp mà mình sở hữu nhưng lại không có quyền hạn ghi, bạn có thể phòng ngừa việc lỡ tay xoá mọi thứ trong thư mục bằng lệnh sau đây:

```
# touch "0 0"
```

Lệnh trên tạo ra tệp mang tên "0 0". Trong chuỗi ASCII, ký tự số “0” được sắp xếp trước mọi ký tự chữ, do đó khi gõ lệnh **rm** * thì **rm** sẽ thử xoá tệp "0 0" đầu tiên và dừng lại hỏi. Nếu quả thật không muốn xoá mọi thứ trong thư mục, bạn còn kịp bấm <Del> hoặc <Ctrl-c> để huỷ (kill) tiến trình **rm**. Muốn thử, bạn hãy xoá tệp "0 0" xem. Đừng gõ **rm** * nữa nếu phiên bản bạn đang dùng lại không dừng lại hỏi.

Một cách hay hơn nữa để phòng trường hợp lỡ tay xoá tất là bạn dùng cờ -i với **rm**. Ở đây -i viết thay chữ interactive (tương tác). Nếu gõ **rm** -i tên_tệp, máy sẽ yêu cầu bạn khẳng định. Và bạn phải trả lời "yes" trước khi tệp được xoá. Nếu bạn gõ **rm** -i *, máy sẽ buộc bạn trả lời cho từng tệp trong thư mục.

Khi phải thường xuyên sử dụng **rm** -i, bạn có thể cài lệnh này vào trong một shell script hoặc tạo ra một chức năng shell. Nếu thảo ra shell script, bạn nên nhớ là shell sẽ duyệt qua các lệnh trong thư mục liệt kê ở biến PATH theo đúng thứ tự liệt kê. Nếu thư mục $HOME/bin của bạn ở dưới chót, shell script mang tên **rm** sẽ không bao giờ được tìm thấy. Bạn có thể đặt thư mục $HOME/bin ở đầu danh sách của biến PATH, hoặc tạo ra một lệnh mới, chẳng hạn như **del**. Nếu tạo ra shell script mang tên “**del**”, bạn phải đánh dấu "khả thi" bằng lệnh **chmod** để shell có thể nhận ra nó. Khi tạo ra lệnh **del**, bạn chỉ phải ghi một lệnh duy nhất là **rm** -i $*. Bởi vì khi gõ lệnh **del** *, shell sẽ diễn dịch ra là **rm** -i *.

Xem "Chỉnh sửa và đặt bí danh cho các lệnh shell", Chương 18

Còn một cách khác để xoá tệp là dùng alias (bí danh). Bạn có thể hiểu alias như là một lệnh shell nội bộ (giống các lệnh doskey như của DOS kể từ phiên bản 5.0).

Trong khi đang dùng shell C, nếu muốn thêm alias bạn phải chỉnh sửa lại tệp mang tên .cshrc. Bạn có thể sử dụng bất kỳ trình soạn thảo văn bản nào, chẳng hạn như **vi** (xem

"Sử dụng trình soạn thảo **vi**") để chỉnh sửa tệp này. Với shell C, bạn thêm những dòng sau đây vào đầu tệp .cshrc:

```
rm()
{
/bin/rm -i $*
}
```

Muốn thêm **alias** vào shell Korn, bạn thêm những dòng sau vào tệp $HOME/.kshrc

```
alias rm 'rm -i $*'
```

Khi định xoá một thư mục bằng lệnh **rm**, máy sẽ báo đấy là thư mục và không được xoá. Nếu muốn xoá các thư mục rỗng, bạn dùng lệnh rmdir như với DOS.

Linux còn một cách khác để xoá thư mục cùng với nội dung, nhưng cách này rất nguy hiểm: lệnh **rm** -r sẽ xoá bất kỳ thư mục và tệp nào được phát hiện. Thí dụ bạn có thư mục ./foo chứa tệp và thư mục cấp dưới, lệnh **rm** -r foo sẽ xoá sạch nội dung thư mục ./foo, kể cả các thư mục cấp dưới.

Với lệnh **rm** -i -r, máy sẽ yêu cầu bạn khẳng định trước khi thực hiện thao tác xoá. Trong trường hợp thư mục chứa rỗng, **rm** sẽ không tự ý xoá, giống như trường hợp bạn ra lệnh **rm** mà không kèm tuỳ chọn nào.

Ghi chú: Bạn không nhất thiết phải đặt từng cờ cho mỗi lệnh Linux. Nếu cờ nào không cần đối số kèm theo, bạn có thể phối hợp chúng với nhau. Thí dụ, **rm** -i -r có thể được viết là:

```
rm -ir
```

17.6 Xem nội dung của tệp

Hầu hết các lệnh Linux đều đưa kết quả ra thiết bị xuất chuẩn, tức màn hình. Những lệnh này thường hiển thị ngay lên màn hình trong hoặc sau khi thao tác tệp hoặc nhập liệu trực tiếp từ bàn phím. Muốn tệp được hiển thị theo ý mình, bạn hãy chọn một lệnh Linux thích hợp, trong đó có ba lệnh tiêu chuẩn sẽ được bàn kỹ hơn ở dưới đây, đó là: **cat**, **more** và **less.**

17.6.1 Các thiết bị xuất nhập chuẩn

Linux, cũng như mọi hệ thống UNIX, khi khởi động sẽ mở ra bốn (tệp) thiết bị xuất nhập chuẩn đó là: nhập tiêu chuẩn (stdin), xuất tiêu chuẩn (stdout), lỗi tiêu chuẩn (stderr) và cổng phụ (AUX). Những thiết bị này được coi như mặc định, tức là không cần nêu tên trong dòng lệnh.

Đầu vào/ra	*Tiếng Anh*	*Tên*	*Thiết bị mặc định*
Nhập tiêu chuẩn	standard input	stdin	Bàn phím
Xuất tiêu chuẩn	standard output	stdout	Màn hình
Lỗi tiêu chuẩn	standard error	stderr	Màn hình
Cổng phụ	auxiliary	AUX	Cổng COM

Ghi chú: Nếu không muốn dùng những giá trị mặc định, bạn có thể dùng các dấu > và < với tên (tệp) để chỉnh hướng các đầu vào/ra theo ý mình.

Xem "Chỉnh hướng xuất nhập"

17.6.2 Xem tệp bằng lệnh cat

cat là một lệnh dùng để xem những tệp văn bản ngắn dạng ký tự ASCII. **cat** tên_tệp mở một tệp, sau đó hiển thị hết nội dung lên màn hình. Nếu đối số là nhiều tệp thì **cat** móc nối chúng lại với nhau trước khi đưa lên màn hình, tệp này tiếp nối tệp kia, thí dụ:

```
cat tệp1 tệp2 tệp3
```

Khi dùng **cat** để hiển thị một tệp dài hơn 1 trang, bạn sẽ thấy nội dung của tệp khó đọc vì các dòng trôi nhanh như chạy. Muốn đọc, bạn phải luân phiên bấm <Ctrl-s> và <Ctrl-q> để máy biết khi nào thì cho trôi hay dừng lại. Nếu không, bạn có thể xem từng trang màn hình bằng các lệnh **more** hoặc **less.**

17.6.3 Xem tệp bằng lệnh more

Cả lệnh **more** và **less** đều hiển thị mỗi lần một trang màn hình. Căn cứ vào biến môi trường TERM và cơ sở dữ liệu về các loại terminal có trên máy, lệnh **more** và **less** sẽ xác định mỗi trang đó gồm bao nhiêu dòng.

Lệnh **more** ra đời trước **less**, xuất xứ từ phiên bản Berkeley UNIX. Lệnh này ngày càng chứng tỏ hiệu quả của mình nên đã trở thành một tiêu chuẩn, cũng như trường hợp của trình soạn thảo **vi.**

Cũng như **cat**, hình thức đơn giản nhất của lệnh **more** là:

```
more tên_tệp
```

Máy sẽ hiển thị trang màn hình đầu tiên của tệp. Muốn xem tiếp trang sau, bạn bấm thanh phím Spacebar, còn nếu bấm phím <Enter> thì màn hình chỉ dịch lên một dòng.

Nếu đang xem một chuỗi tệp (bằng lệnh **more** tệp1 tệp2 ...) và muốn dừng lại để chỉnh sửa tệp nào đó, bạn bấm phím <e> hoặc <v>.

Bấm phím <e> với **more** nghĩa là bạn gọi một trình soạn thảo đã được xác định trong biến môi trường shell EDIT liên quan đến tệp hiện hành.

Nếu bấm <v>, máy sẽ gọi trình soạn thảo nào được xác định trong biến môi trường VISUAL. Nếu bạn chưa định nghĩa các biến này trong môi trường, **more** sẽ mặc định dùng trình soạn thảo **ed** khi bạn bấm <e> và trình soạn thảo **vi** khi bấm <v>.

Xem "Thiết lập môi trường shell"

Với lệnh **more**, nếu muốn xem lại trang màn hình đã lật qua, bạn dùng phím <b> Ngoài ra lệnh **less** cũng có thể giúp bạn việc này.

17.6.4 Xem tệp bằng lệnh less

Điểm bất tiện của lệnh **less** là bạn không thể sử dụng phần mềm soạn thảo với tệp đang được hiển thị. Tuy nhiên để bù lại, lệnh **less** giúp bạn di chuyển tới lui khi xem tệp.

Công dụng của lệnh **less** gần giống lệnh **more**. Để xem từng trang của tệp, bạn gõ:

```
less tên_tệp
```

Máy sẽ hiển thị đầy một trang màn hình. Cũng như với lệnh **more**, bạn bấm phím Spacebar để xem trang tiếp theo.

Muốn xem màn hình vừa trôi qua, bạn bấm <b>. Muốn di chuyển đến vị trí nào đó trong tệp (vị trí này được biểu thị bằng số phần trăm), bạn bấm <p> và khi dấu nhắc <:> hiện ra, bạn gõ số phần trăm vào.

17.6.5 Duyệt tìm xuyên tệp và thoát khỏi shell

Hai lệnh **less** và **more** giúp bạn duyệt suốt tệp đang hiển thị để tìm ra một chuỗi ký tự. Lệnh **less** còn cho phép bạn tìm ngược về phía đầu tệp bằng cú pháp sau:

```
less / chuỗi_ký_tự
```

Sau khi tìm thấy chuỗi_ký_tự, lệnh **less** và **more** sẽ hiển thị chuỗi ấy ở đầu một trang mới mở. Với **less**, khi bạn bấm <n> máy sẽ lặp lại thao tác tìm vừa rồi.

Khi dùng hai lệnh **less** và **more**, nếu bấm <!> bạn sẽ thoát khỏi shell. Thực ra khi thoát khỏi bằng lệnh <!>, bạn đang ở trong một shell thứ cấp. Từ đây bạn phải thoát khỏi giống như bạn đăng xuất một phiên làm việc bình thường.

Tuỳ vào shell đang sử dụng, bạn có thể bấm <Ctrl-d> hoặc gõ **exit** để trở về màn hình **more** hoặc **less** mà bạn vừa thoát khỏi.

Nếu bấm <Ctrl-d> mà máy yêu cầu bạn dùng lệnh logout thay vì <Ctrl-d>, bạn hãy làm theo máy.

17.6.6 Xem tệp bằng những cách khác

Có thể hiển thị nội dung tệp bằng nhiều cách khác. Thí dụ, nếu muốn xem nội dung của một tệp nhị phân, bạn dùng lệnh **od** (từ chữ **o**ctal **d**ump). Lệnh **od** hiển thị tệp dưới dạng mặc định là bát phân (hệ đếm cơ sở 8). Dùng kèm với những cờ khác, **od** có thể hiển thị tệp nhị phân dưới dạng thập phân, ASCII, hoặc thập lục phân (cơ sở 16).

Sở dĩ phải hiển thị tệp theo những gián tiếp như dạng bát phân, thập phân, hoặc thập lục phân là vì hiển thị dữ liệu ở dạng nhị phân vẫn còn là một vấn đề nan giải. Nếu dữ liệu được thể hiện ở dạng ASCII thì rất dễ đọc, vì các ký tự ASCII là dạng mà mọi người nhìn thấy khi xem hầu hết các tệp văn bản. Song nếu tệp là một chương trình nhị phân thì dữ liệu không thể được trình bày ở dạng ASCII. Trong trường hợp này bạn phải hiển thị bằng hình thức chuyển đổi hệ đếm số.

UNIX ra đời trên những máy minicomputer sử dụng các ký tự 12 bit. Chúng ta ngày nay thường lấy byte (8 bit) làm một chuẩn độ dài bộ nhớ. Mặc dù có thể hiển thị theo hệ thập phân quen thuộc, song vấn đề là hiển thị cái gì - một byte, một chữ, hay 32 bit? Thao tác hiển thị một số lượng bit nào đó, đòi hỏi phải chuyển đổi hệ nhị phân căn cứ trên số lượng bit ấy. Với hệ 12 bit bạn có thể hiển thị toàn bộ 12 bit với bốn con số (hiển thị bằng 2^3, dưới dạng bát phân). Các hệ UNIX trước đây đều chạy trên những máy tính 12 bit như thế, do đó phần lớn các phép ghi của UNIX (và do đó của Linux) đều ở dạng bát phân. Bất cứ byte nào cũng có thể hiển thị bằng một mã bát phân với ba chữ số có dạng như sau (thí dụ hiển thị giá trị thập phân của 8):

```
\010
```

Vì thế giới ngày nay đã chuyển sang dạng byte 8 bit nên bát phân không còn hiệu quả khi trình bày dữ liệu. Hệ thập lục phân (cơ sở 16 hoặc 2^4) là cách tốt hơn. Một byte 8 bit có thể trình bày bằng hai chữ số thập lục phân. Thí dụ byte có giá trị thập phân là 10 sẽ được hiển thị thành 0A theo cơ sở 16.

Lệnh **od** giúp bạn chọn lựa cách hiển thị dữ liệu. Hình thức tổng quát như sau:

```
od [tuỳ_chọn] [tệp]
```

hoặc:

```
od - traditional [tệp] [[+]offset[[+]nhãn]]
```

*Bảng 17.3: Các cờ dùng với lệnh **od***

Cờ ngắn	**Cờ dài**	**Mô tả**
-A	--address-radix=cơ sở	Xác định cách in những khoảng trống offset trong tệp
-N	--read-bytes=byte	Giới hạn phần kết xuất bằng lượng byte nhập liệu cho mỗi tệp
-j	--skip-bytes=byte	Bỏ qua lượng byte nhập liệu đầu tiên ở mỗi tệp
-s	--strings[=byte]	Xuất chuỗi với ít nhất là bao nhiêu byte ký tự đồ hoạ
-t	--format=loại	Chọn một hoặc nhiều dạng xuất liệu
-v	--output-duplicates	Không cho phép dùng * để đánh dấu việc bỏ dòng
-w	--width[=byte]	Xuất bao nhiêu byte ở mỗi dòng xuất liệu
	--traditional	Chấp nhận đối số ở dạng pre-POSIX
	--help	Hiển thị phần trợ giúp này và thoát khỏi
	--version	Xuất thông tin phiên bản và thoát khỏi

Bạn có thể dùng chung dạng thức pre-POSIX ở bảng sau đây với các lệnh ở bảng trên và nhận được kết quả tổng hợp.

*Bảng 17.4: Các đặc tả dạng pre-POSIX dùng với lệnh **od**.*

Cờ ngắn	*Tương đương POSIX*	**Mô tả**
-a	- t a	Chọn các ký tự có tên
-b	-t cC	Chọn các byte bát phân
-c	-t c	Chọn các ký tự ASCII hoặc backlash escape
-d	-t u2	Chọn các thập phân ngắn không đánh dấu
-f	-t fF	Chọn các float
-h	-tx2	Chọn các thập lục phân ngắn
-I	-t d2	Chọn thập phân ngắn

-l	-t d4	Chọn thập phân dài
-o	-t o2	Chọn bát phân ngắn
-x	-t x2	Chọn thập lục phân ngắn

Với cú pháp cũ (dạng second-call), offset có nghĩa là -j offset. nhãn là một pseudo-address (giả-địa chỉ) tại byte đầu tiên được in, pseudoaddress này tăng dần trong tiến trình kết xuất. Với offset và nhãn, tiền tố 0x hoặc 0X cho biết đó là dạng thập lục phân. Hậu tố có thể là dấu chấm đối với bát phân, hoặc có thể nhân với 512. Tham số loại được hình thành từ một hoặc nhiều khai báo liệt kê ở bảng dưới đây.

Bảng 17.5: Các tham số loại

Tham số	**Mô tả**
a	Ký tự có đặt tên
c	Ký tự ASCII hoặc backslash escape
d [kích cỡ]	Số thập phân có đánh dấu, kích cỡ byte cho mỗi số nguyên.
f [kích cỡ]	Dấu phẩy động, kích cỡ byte cho mỗi số nguyên.
o [kích cỡ]	Bát phân, kích cỡ byte cho mỗi số nguyên.
u [kích cỡ]	Thập phân không đánh dấu, kích cỡ byte cho mỗi số nguyên.
x [kích cỡ]	Thập lục phân, kích cỡ byte cho mỗi số nguyên.

Trong bảng trên, kích cỡ là một số và cũng có thể là C cho sizeof(char), S cho sizeof(short), I cho sizeof(int), hoặc L cho sizeof(long). Nếu là loại f, kích cỡ có thể là F cho sizeof(float), D cho sizeof(double), hoặc L cho sizeof(Long double).

Ghi chú: Trong ngôn ngữ C, sizeof là một chức năng phản hồi số byte trong cấu trúc dữ liệu được dùng làm tham số. Thí dụ bạn sẽ dùng chức năng gọi sau đây để xác định số byte trong một số nguyên trên hệ thống máy, bởi vì số byte trong một số nguyên tuỳ thuộc vào hệ thống:

```
sizeof(int);
```

cơ số trong bảng 17.3 đại diện cho số hệ thống và sẽ là d cho thập phân deca, O cho bát phân (octa), X cho thập lục (hexa), hoặc n cho none (không có). Byte là thập lục phân với tiền tố 0x hoặc 0X; byte được nhân với 512 nếu có hậu tố b (bit), nhân với 1024 nếu có k (kilo) và với 1.048.476 nếu có m (mega). Nếu không có con số nào, -s sẽ mặc định 3, trong trường hợp tương tự, -w sẽ mặc định 32. Theo mặc định, **od** dùng -A0 -td2 -w16.

17.7 Duyệt tìm tệp

Nếu dùng lệnh **ls** tìm vẫn không ra một tệp, bạn thử dùng lệnh **find**. **find** là công cụ cực kỳ mạnh và cũng chính vì thế mà **find** lại trở thành một trong những lệnh khó dùng nhất. **find** có ba phần, mỗi phần lại có thể gồm nhiều phần nhỏ:

-Tìm ở đâu

-Tìm cái gì

-Tìm được rồi thì thực hiện thao tác nào.

Nếu biết tên tệp nhưng không biết vị trí tệp ấy, bạn gõ lệnh như sau:

```
find /-name tên_tệp -print
```

Ghi chú: Hãy cẩn thận khi duyệt tìm từ thư mục gốc. Trên những hệ thống lớn, nếu tìm từ thư mục gốc, máy sẽ lục lọi mọi thư mục bên dưới rồi đến phiên các thư mục cấp dưới nữa, chưa kể những ổ đĩa khác tại chỗ và ổ đĩa từ xa, do đó sẽ tốn rất nhiều thì giờ.

Do đó có lẽ bạn nên giới hạn việc duyệt tìm ở một hai thư mục thôi. Thí dụ, khi biết có tệp ở thư mục /usr hoặc /usr2, bạn gõ lệnh:

```
find /usr /usr2 -name tên_tệp -print
```

Bạn có thể dùng nhiều tuỳ chọn với **find**. Bảng sau đây chỉ liệt kê một phần nhỏ. Muốn tham khảo mọi tuỳ chọn, bạn gõ lệnh:

```
man find
```

*Bảng 17.6: Một mẫu các cờ dùng với lệnh **find**.*

Lệnh	**Mô tả**
- name tệp	Biến tệp có thể là tên của một tệp hoặc tên tệp kèm ký tự wildcard. Nếu có kèm wildcard, những tệp nào khớp với wildcard sẽ được chọn để xử lý.
-links n	Bất kỳ tệp nào có số kết nối bằng hoặc lớn hơn n sẽ được chọn. Bạn thay thế n bằng con số cần thiết.
-size n [c]	Bất kỳ tệp nào có số block 512 byte bằng hoặc lớn hơn n sẽ được chọn. Nếu có c gắn theo n, máy sẽ chọn những tệp có số ký tự bằng hoặc lớn hơn n.
-atime n	Tệp nào được truy cập trong vòng n ngày sẽ được chọn. Bạn lưu ý là thao tác duyệt tìm tệp bằng lệnh **find** sẽ thay đổi nhãn ngày tháng của tệp.
-exec cmd	Sau khi chọn xong danh sách tệp, bạn có thể dùng danh sách này làm đối số để chạy một lệnh Linux. Có hai quy định đơn giản với -exec: tên của tệp được đại diện bởi {} và lệnh phải tận cùng bằng dấu chấm phẩy thoát, nghĩa là \;. Giả sử khi đã đăng nhập với quyền hạn root, bạn tạo ra một thư mục user. Kết quả là mọi tệp đều thuộc về root, nhưng lẽ ra chúng phải thuộc quyền sở hữu của user đó. Bạn phải ra lệnh sau đây để đổi chủ sở hữu của mọi tệp trong /usr/pla và mọi thư mục cấp dưới từ root sang pla: **find** /home/pla -exec chown pla {}\;
-print	In tên và vị trí các tệp được chọn.

Lệnh **find** giúp bạn thực hiện nhiều thử nghiệm logic đối với tệp. Thí dụ bạn muốn tìm một loạt những tên tệp nhưng không thể dùng wildcard để đại diện, vì không khớp hoàn toàn. Trường hợp này bạn dùng tuỳ chọn or (-o) như sau:

```
find /home (-name tệp1 -o -name tệp2) -print
```

Với **find**, bạn có thể phối hợp bao nhiêu tiêu chí chọn cũng được. Nếu không dùng tuỳ chọn -o, máy sẽ ngầm hiểu là bạn dùng and.

Thí dụ lệnh:

```
find / size 100 -atime 2
```

Có nghĩa là bạn muốn tìm một tệp có kích cỡ ít nhất 100 block và được truy cập trong thời gian là ít nhất hai ngày vừa qua trên toàn bộ hệ thống. Bạn có thể dùng dấu ngoặc đơn, như thí dụ trên để phòng việc máy hiểu sai ý của mình, nhất là trong trường hợp bạn phối hợp hai tiêu chí và/hoặc.

17.8 Thay đổi nhãn ngày giờ

Mỗi tệp Linux duy trì ba nhãn ngày giờ:

-Ngày giờ tạo ra tệp

-Ngày giờ lần thay đổi gần nhất và

-Ngày giờ lần truy cập gần nhất

Bạn có thể thay đổi nhãn thời gian tạo ra tệp, trừ khi bạn thực hiện thao tác sao chép và đặt lại tên tệp. Mỗi khi tệp được mở hoặc được đọc, nhãn thời gian truy cập sẽ thay đổi. Và như đã nói ở đoạn trên, dùng lệnh **find** duyệt tìm tệp cũng khiến nhãn ngày tháng truy cập thay đổi.

Nhãn thời gian của tệp có ích khi bạn cần sao lưu có chọn lọc những tệp nào đã thay đổi ở thời điểm nhất định. Ở đây bạn cũng dùng lệnh **find.**

Nếu muốn thay đổi nhãn thời gian trong khi vẫn giữ nguyên tệp, bạn dùng lệnh **touch** (sửa lại). Theo mặc định, **touch** cập nhật nhãn thời gian truy cập và chỉnh sửa theo đồng hồ hệ thống hiện hành. Cũng theo mặc định, nếu bạn định thao tác một tệp hiện vẫn chưa có thực, **touch** sẽ tạo ra tệp ấy.

Bạn có thể dùng **touch** để "đánh lừa" lệnh nào có chức năng kiểm tra ngày tháng. Thí dụ nếu hệ thống máy chạy sao lưu một số tệp đã thay đổi vào thời gian nào đó, bạn có thể **touch** một tệp để tệp ấy được hay không được tính đến.

Lệnh **touch** có ba cờ sau đây:

-a	Chỉ cập nhận nhãn thời gian truy cập tệp.
-m	Chỉ cập nhật nhãn thời gian chỉnh sửa tệp.
-c	Không cho **touch** tạo ra một tệp mới.

Cú pháp mặc định của **touch** là:

```
touch  -am danh_sách_tệp
```

17.9 Nén và nới tệp

Nếu sức chứa của hệ thống có hạn, hoặc nếu có những tệp ASCII quá lớn mà ít khi dùng đến, bạn có thể nén chúng lại để tiết kiệm khoảng trống trên đĩa. Tiện ích tiêu chuẩn của Linux để nén tệp là **gzip**. Lệnh này có thể nén tệp ASCII còn chừng 20% kích thước ban đầu.

```
gzip tên_tệp
```

Hầu hết các hệ UNIX đều có lệnh **compress**, thường dùng với tar để nén từng nhóm tệp dùng cho tồn trữ. Tệp nén bằng lệnh **compress** có đuôi .z, thí dụ archive1.tar.z. Bản phát hành Red Hat còn có các chương trình **zip** và **unzip** để nén và tồn trữ một loạt các tệp.

Sau khi nén thành công một tệp bằng lệnh **gzip** tên_tệp, tệp nén sẽ mang tên dạng tên_tệp.gz, sau đó tệp nguyên thuỷ bị huỷ. Muốn phục hồi tệp nén về trạng thái cũ, bạn dùng lệnh sau:

```
gunzip tên_tệp
```

Ghi chú: Khi bung một tệp, bạn không cần phải gõ đủ cái đuôi .gz phía sau tên tệp. Lệnh **gunzip** sẽ tự động hiểu.

Trong trường hợp muốn giữ tệp ở thể nén, những lại muốn đặt ống dẫn dữ liệu sang một lệnh khác, bạn dùng lệnh **zcat**. Lệnh này hoạt động như lệnh **cat**, song đòi hỏi phía đầu vào phải là tệp nén. **zcat** bung tệp rồi hiển thị lên thiết bị kết xuất tiêu chuẩn.

Xem "Kết nối các tiến trình bằng những ống dẫn"

Thí dụ. bạn đã nén một loạt nhiều tên và địa chỉ trong tệp mang tên namelist và tệp nén xong mang tên namelist.gz. Giờ đây nếu muốn dùng nội dung tệp nén để làm nhập liệu cho một chương trình, bạn dùng lệnh **zcat** để bắt đầu một ống dẫn:

```
zcat namelist | chuongtrình1 | chuongtrình2
```

zcat cũng có cùng những hạn chế như **cat**: không thể trở lui về sau trong một tệp. Linux có chương trình **zless** hoạt động như **less**, chỉ khác là **zless** làm việc với tệp nén. Những lệnh nào làm việc được với **less** sẽ làm việc được với **zless.**

Những bạn nào đã quen sử dụng pkzip của pkware giờ có thể dùng **zip** và **unzip** trong bản phát hành Red Hat, vì chúng hoạt động tương tự nhau. Mời bạn đọc các trang **man** của **zip** và **unzip** để biết thêm chi tiết.

17.10 Quản lý tệp và thư mục trong môi trường X

17.10.1 Giao diện GNOME

Trong môi trường X với giao diện GNOME có trình duyệt Nautilus hoạt động như Explorer trong Windows. Bạn khởi động Nautilus theo một trong hai cách sau:

Mở Main Menu Button -> Programs->Applications và bấm vào Nautilus
Trên Desktop, bấm chuột trái vào biểu tượng ngôi nhà của Home Directory

Minh hoạ 17.1

Trong cửa sổ Nautilus, phần bên trái biểu diễn thư mục đang làm việc, phần bên phải cho thấy nội dung của thư mục đó. Bạn có thể thay thế các biểu tượng (View as Icon) bằng cách bấm trên menu sổ xuống ở góc phải và chọn "View as List".

17.10.2 Di chuyển trong Nautilus

Dùng các nút dịch chuyển trên thanh Định vị (Location bar), bạn có thể lướt duyệt trên các tệp và thư mục.

```
Up: di chuyển về thư mục cha.
Back / Forward: di chuyển qua lại theo quá trình duyệt thư mục.
Refresh: làm tươi lại cửa sổ.
Home: di chuyển về thư mục mặc định của user.
Web search: mở trình tìm kiếm Web.
Stop: ngưng việc nạp trang.
Chép và dời tệp (Copy /Move file)
```

Việc bấm chuột lên một tệp muốn sao chép hay di dời, kéo và thả từ thư mục cũ sang thư mục mới... đều tương tự như trong Windows. Song nếu chỉ sao chép, bạn cần ấn và giữ phím Shift chứ không phải phím Ctrl như trong Windows.

Bạn cũng có thể bấm chuột phải lên biểu tượng và chọn Copy (khi sao chép) hay Cut (khi dời).

Khi bạn không chắc chắn sẽ chỉ sao chép hay là dời hẳn một tệp sang vị trí mới, hãy ấn và giữ phím Alt khi bạn kéo rê tệp đó sang vị trí mới. Lúc này, khi bạn buông tay nhấn, một trình chọn hiện ra cho phép bạn định lại việc mình sẽ làm.

17.10.3 Thuộc tính của tệp

Hai minh hoạ 17.2 và 17.3 cho thấy các thuộc tính của một tệp cho bạn biết các thông tin về tệp đó và chỉ định lại chúng.

17.10.4 Định lại cấu hình cho Nautilus

Bấm vào Preferences ở đầu màn hình của Nautilus, bạn có thể xác định lại cấu hình của Nautilus để nó hiện ra theo yêu cầu sử dụng.

Minh hoạ 17.4: Cửa sổ cấu hình Nautilus.

Bạn bấm vào bảng bên trái và chỉnh lại các thông số tương ứng ở bảng bên phải.

17.10.5 Giao diện KDE

Trình duyệt trong môi trường KDE được gọi là Konqueror (hay Konquerer). Bạn khởi động Konqueror bằng một trong hai cách sau:

Từ Main Menu K => bấm vào Home Directory
Từ Panel=> bấm vào Home Directory.

Các thao tác trên Konqueror hoàn toàn tương tự như trong Windows hay trên Nautilus. Bạn có thể thay đổi cấu hình và biểu hiện của Konqueror thông qua các điều chỉnh phần Option=>Configure File Manager.

Minh hoạ 17.5: Màn hình Konqueror

Chương 18. Các shell của Linux

Mặc dù những năm gần đây đã có vài giao diện đồ hoạ, song phần lớn những công cụ sử dụng và quản trị hệ Linux (cũng như các hệ kiểu UNIX) vẫn chạy bằng cách điều khiển từ dòng lệnh. Chương trình điều khiển những lệnh như thế được gọi là shell (tương tự COMMAND.COM của DOS). Dưới đây sẽ giải thích cách dùng các shell khác nhau để làm việc với những tiện ích và hệ thống tệp của Linux.

Các chủ đề chính sẽ được đề cập đến trong chương này gồm có:

-Đăng nhập
-Tìm hiểu các shell
-Tìm hiểu về việc phân tách lệnh trong shell
-Thực hiện công việc xử lý hậu trường
-Tìm hiểu việc phản hồi lệnh
-Chỉnh sửa và đặt bí danh cho các lệnh shell
-Làm việc với shell script
-Căn chỉnh các shell Linux

18.1 Đăng nhập

Với trương khoản của một user, đồng thời là một quản trị viên (root) trên hệ Linux, bạn có mật khẩu và ID để đăng nhập. Là một hệ thống multiuser, Linux biết phân biệt user và các cấp user. Linux sử dụng ID đăng nhập của bạn để thiết lập một phiên làm việc với tên của bạn và xác định những ưu tiên mà bạn được cấp. Linux sử dụng mật khẩu để kiểm tra xem bạn là ai.

Bởi vì trên lý thuyết mỗi user có thể đăng nhập ở bất kỳ terminal nào mà họ thích (thực ra vẫn có ngoại lệ), hệ điều hành UNIX bắt đầu phiên làm việc bằng cách hiển thị dấu nhắc đăng nhập ở mỗi terminal. Vẫn trên lý thuyết, bởi vì bạn mới bắt đầu dùng hệ thống Linux vừa khởi động, cho nên ít có khả năng bạn kết nối nhiều terminal thực vào hệ thống, mà thường sử dụng loại terminal thay thế, còn gọi là terminal ảo.

Để chuyển sang các terminal ảo, bạn bấm phím <Alt> cùng với một trong sáu phím chức năng (F1-F6). Thí dụ muốn đăng nhập terminal ảo số 1 với tư cách là root, bạn bấm <Alt-F1> và máy sẽ hiển thị mấy dòng đại để như sau:

```
Red Hat Linux release 7.3 (Valhala)
Kernel 2.4.18-3 on an i686
login:
```

Ghi chú: mấy dòng trên cho biết phiên làm việc này sẽ chạy lõi Linux phiên bản 2.4.18-3 trên chip Intel Pentium. Càng phát triển các kernel mới hơn thì dãy số phiên bản sẽ lớn hơn. Các kernel có chung nguồn gốc ổn định được đại diện bởi con số chẵn ở giữa, trong khi số lẻ cuối cùng cho biết đó là bản mới nhất hoặc bản beta.

Sau dấu nhắc "login:" bạn có thể gõ vào tên user (root) cùng với mật khẩu của mình. Kể từ lúc bạn đăng nhập một terminal, phiên làm việc sẽ thuộc quyền ID của bạn cho đến khi đăng xuất.

Sau khi bạn đăng xuất, Linux lại hiển thị dấu nhắc để bạn có thể đăng nhập phiên tiếp theo. Giữa hai thời điểm đăng nhập và đăng xuất, Linux đảm bảo bạn sở hữu tất cả những chương trình do bạn chạy và các tệp do bạn tạo ra. Ngược lại, Linux không cho phép bạn đọc hoặc thay đổi một tệp của user khác, trừ khi quản trị viên hoặc chính user ấy cho phép bạn. Tóm lại, ID và mật khẩu đăng nhập giúp Linux duy trì an ninh cho tệp của bạn cũng như của những user khác.

Nếu bạn là quản trị viên, bạn có thể cấp cho mỗi user các thông tin sau: ID cá nhân, mật khẩu tạm thời, ID nhóm, home directory và shell. Những thông tin vừa kể được lưu trong một tệp mang tên /etc/passwd, thuộc quyền sở hữu và kiểm soát của quản trị viên, còn gọi là superuser hay root. Sau khi đăng nhập thành công, user có thể thay đổi mật khẩu và mật khẩu này được máy mã hoá để cho người khác không đọc được. Nếu một user quên mất mật khẩu, quản trị viên (hoặc người nào đó có quyền) sẽ đăng nhập hệ thống và tạo cho user ấy mật khẩu mới. Lệnh passwd giúp bạn tự thay đổi mật khẩu nếu bạn còn nhớ mật khẩu cũ, còn superuser thì không cần.

18.2 Tìm hiểu các shell

Sau khi đăng nhập, Linux đưa bạn về home directory của bạn và chạy một chương trình gọi là shell (vỏ ốc). Shell là một loại chương trình điều khiển, nó nhận (các) lệnh từ bàn phím hoặc từ một tệp, rồi sau đó thi hành. Nhiều chương trình có thể được dùng dưới dạng shell, song mỗi phiên bản Linux đều có sẵn một vài shell tiêu chuẩn.

Ghi chú: Các shell của Linux hoạt động tương tự như COMMAND.COM của DOS, tức là đều có thể nhận và thi hành lệnh, chạy tệp **batch** và các chương trình khả thi.

18.2.1 Điểm qua các shell tiêu chuẩn

Bản Red Hat Linux có các shell sau đây: sh, bash (viết tắt từ Bourne Again SHell), tcsh, csh, pdksh (Public Domain Korn SHell), zsh, ash và mc. Chương này tập trung xem xét hai shell sh và bash, bởi vì hầu hết các bản phát hành Linux đều cài đặt bash làm shell tiêu chuẩn và hầu như hệ UNIX nào cũng có sh, ngoài ra bạn sẽ thấy rằng nhiều shell script được viết bằng những lệnh sh.

Vì shell giữ vai trò giao diện đầu tiên giữa hệ điều hành và user, cho nên nhiều người xem shell chính là Linux, song thực chất shell không phải là thành phần của kernel. Nếu biết lập trình hệ thống và hiểu biết về hệ điều hành Linux, bạn có thể tự viết một chương trình shell.

Mặc dù có rất nhiều shell đã được tạo ra, song nổi bật nhất vẫn là các shell sau đây: sh, Korn, C và T.

Shell sh hay Bourne là shell xuất hiện trước nhất và không có một số đặc trưng của các shell ra đời sau. Linux sử dụng một biến thể của shell Bourne gọi là shell bash và xem đó như shell mặc định của chính mình.

Đối với người mới sử dụng Linux, hai shell Bourne và Korn trông giống nhau y hệt; thực ra Korn được phát triển từ Bourne.

Shell C được phát triển tại Đại học California (Berkeley) và thích hợp cho giới lập trình hơn là shell Bourne.

Ở hình thức đơn giản nhất của mình, hai shell Bourne và Korn dùng ký tự dollar ($) làm dấu nhắc mặc định. Shell C dùng dấu phần trăm (%). Tuy nhiên bạn có thể thay đổi tuỳ ý các dấu nhắc.

18.2.1.1 Shell Bourne

Shell Bourne là shell UNIX nguyên thuỷ, do Steve Bourne viết ra với sự trợ giúp của John Mashey, cả hai hồi đó làm việc cho Bell Laboratories. Chương trình thi hành của shell này nằm trong tệp /bin/sh. Vì có sẵn trên mọi hệ Linux, với những tính năng đã nói ở trên và sức mạnh lập trình, Bourne đã trở thành một shell rất phổ biến.

Ghi chú: Nhiều thí dụ về shell script trong chương này được viết riêng để chạy với shell Bourne. Shell script là tệp những dòng lệnh shell, thường được viết bằng một trình soạn thảo văn bản, chẳng hạn như **vi**. Bạn có thể hiểu shell script là tương đương với loại tệp **batch** của DOS.

18.2.1.2 Shell C

Shell C, còn gọi là csh, được phát triển bởi Bill Joyat thuộc Đại học Berkeley. Trường Berkeley có ảnh hưởng lớn lao đến UNIX và do đó đến Linux. Trường là nơi ra đời của shell C và trình soạn thảo văn bản **vi**. Tuy Bourne có ưu thế về lập trình shell, song shell C lại có tính tương tác cao hơn. Chương trình thi hành của shell C nằm trong tệp /bin/csh.

Cú pháp shell C rất giống ngôn ngữ lập trình C. Đó là một trong những lý do tại sao các shell script viết cho shell C thường không thể chạy với các shell Bourne và Korn (tất nhiên những tệp khả thi soạn thảo bằng shell C thường chạy không có vấn đề). Nhưng shell C có một vài đặc trưng mà shell Bourne không có: biên soạn lệnh, theo dõi lịch trình sử dụng lệnh và đặt bí danh cho lệnh.

18.2.1.3 Shell bash

Shell mặc định là bash, nằm tại /bin/bash. Mọi phiên bản Linux đều có shell này.

Trong tiến trình cài đặt, có thể bạn đã cài thêm một số shell khác. Muốn biết mình đang dùng shell nào, bạn gõ:

```
echo $SHELL
```

Lệnh **echo** sẽ hiển thị lên màn hình terminal tất cả những gì theo sau chữ **echo.** SHELL là một biến do shell duy trì và mang tên của shell hiện hành. $SHELL là giá trị của biến ấy.

Muốn biết shell C có sẵn không, bạn gõ lệnh:

```
csh
```

Nếu ký tự % xuất hiện tại dấu nhắc, có nghĩa là shell C có sẵn và đang chạy. Lúc này nếu cần trở về shell trước, bạn gõ **exit**. Nếu đã đăng nhập như là root, dấu nhắc cho shell C là #. Còn nếu máy báo lỗi thì nghĩa là shell C không sẵn dùng đối với bạn.

Shell mà bạn dùng làm shell đăng nhập được khai báo trong tệp mật khẩu. Mỗi ID đăng nhập được đại diện bởi một mục ghi hoặc một dòng trong tệp mật khẩu.

Trường cuối cùng của mục ghi là shell đăng nhập của bạn. Muốn thay đổi shell đăng nhập, bạn chỉ thay đổi trường này. Tuy nhiên trước khi thay đổi shell, bạn nên suy nghĩ về việc phải học cú pháp mới và thực hành phương pháp mới. Muốn tham khảo cú pháp của shell, bạn xem các trang **man** tương ứng.

Cẩn thận: Đừng bao giờ trực tiếp chỉnh sửa tệp mật khẩu (/etc/passwd) trong Linux. Bởi vì mỗi bản phát hành mới ra đời đều bổ sung những đặc tính về an ninh, bạn chỉ nên thao tác tệp mật khẩu bằng những lệnh tương thích. Thí dụ với Red Hat Linux, nếu bạn muốn dùng **usermod** để đổi sang shell C, bạn gõ:

```
usermod –s /bin/csh user_id
```

với user_id là số định danh của user mà bạn đang dùng để thay đổi shell, lời nhắc nhở này đặc biệt quan trọng khi bạn dùng bộ tiện ích mật khẩu shadow.

Nhiều shell khác có thể dùng với Linux, xuất phát từ nhiều nguồn đa dạng, hoặc là bản miễn phí trên Internet, hoặc là bản thương phẩm. Trước khi quyết định dùng shell nào, bạn nên đọc các trang **man** về nó và dùng thử. Vì shell là chương trình, bạn chạy chúng như những ứng dụng bình thường.

18.2.2 Lập cấu hình môi trường đăng nhập

Trước khi bạn nhìn thấy dấu nhắc shell, Linux thiết lập môi trường mặc định cho bạn. Môi trường này chứa các giá trị và dữ liệu có chức năng kiểm tra phiên làm việc của bạn trong suốt thời gian đăng nhập. Đương nhiên, bạn có thể thay đổi những tham số vừa kể theo nhu cầu của riêng mình. Môi trường phiên làm việc gồm hai thành phần:

Thành phần thứ nhất gọi là môi trường terminal, liên quan đến thiết bị terminal của bạn (nói đúng hơn là cổng ra vào máy tính, nơi mà bạn nối cáp với terminal).

Thành phần thứ hai gọi là môi trường shell, liên quan đến nhiều khía cạnh khác nhau của shell, cùng với các chương trình do bạn chạy.

Ghi chú: Trong trường hợp phổ biến nhất, khi bạn làm việc với Linux cài trên PC, "terminal" chính là màn hình và bàn phím của PC. Hệ thống của bạn có thể có hoặc không kết nối với những terminal khác. Đương nhiên, bạn có đến 6 terminal ảo và đăng nhập từ terminal nào cũng được.

18.2.2.1 Thiết lập môi trường terminal

Thực ra phiên đăng nhập của bạn bao gồm hai chương trình riêng biệt nhưng chạy cùng lúc với nhau, tạo cho bạn cảm giác rằng máy đang phục vụ cho riêng mình. Mặc dù shell là chương trình nhận lệnh và thi hành, song trước khi shell nhận được lệnh, tất cả những gì bạn gõ vào phải đi qua một trình điều khiển thiết bị, gọi tắt là driver.

Driver kiểm soát terminal, nhận những ký tự bạn gõ vào rồi sau đó quyết định xem xử lý như thế nào trước khi giao cho shell diễn dịch. Tương tự như thế, mỗi ký tự phát sinh từ shell phải đi qua driver trước khi đến terminal. Mục này đề cập trước tiên đến cách kiểm tra hoạt động của driver.

Mọi thiết bị kết nối với hệ thống đều được coi như nhau và được Linux xử lý giống như tệp ; driver giúp cho Linux có cái nhìn nhất quán về thiết bị.

Thí dụ đĩa cứng được phân thành các block, sector và cylinder ; khi đọc hoặc ghi dữ liệu, chúng phải được trỏ đúng địa chỉ.

Khi dữ liệu được gửi về terminal, đó là một chuỗi liên tiếp các ký tự được giao nhận một cách từ từ và có thứ tự. Chính driver sắp xếp thứ tự dữ liệu và gửi đến terminal với tốc độ 1200, 2400, 9600 bps (bit/giây) hoặc nhanh hơn. Đồng thời, driver cũng chèn vào dòng dữ liệu các bit start, stop và phụ trợ.

Vì terminal lúc nào cũng kết nối với hệ thống, driver giúp bạn xác định các ký tự đặc biệt, gọi là ký tự điều khiển, để đánh dấu đầu tệp và cuối tệp cho shell biết. Ký tự điều khiển cũng gửi tín hiệu đến một tiến trình đang hoạt động (chẳng hạn như tín hiệu ngắt có khả năng dừng hẳn một tiến trình đang chạy và đưa bạn trở về shell).

Bạn có thể lập nhiều tham số cho thư mục, song hầu hết các tham số này đều được xử lý tự động. Dù sao, bạn cũng nên biết qua vài tham số và chế độ.

Driver hoạt động theo hai chế độ, gọi là chế độ nguội và chế độ nóng.

Khi driver hoạt động ở chế độ nóng, mọi ký tự bạn gõ vào sẽ đi thẳng đến shell hoặc đến một chương trình do shell chạy. Những chương trình như soạn thảo văn bản và bảng tính điện tử yêu cầu chế độ nóng và tự động thiết lập chế độ nóng. Khi chấm dứt hoạt động, những chương trình loại này thường đưa terminal về chế độ nguội, song không phải lúc nào chúng cũng làm như thế. Lúc đang trong chế độ nóng, terminal không phản ứng với các phím điều khiển, chẳng hạn như phím ngắt (Interrupt).

Khi terminal ở chế độ nguội, driver sẽ diễn dịch mọi ký tự gõ vào, chúng thường được lưu trong vùng đệm cho đến khi bạn bấm phím hết dòng. Trong hầu hết các trường hợp, phím hết dòng chính là <Enter> hoặc <Return>, tuy nhiên phím này có thể thay đổi. Khi nhận được ký tự hết dòng, driver diễn dịch cả dòng trước khi chuyển giao dòng được diễn dịch hoặc được phân tích (parsed) ấy cho shell hoặc cho chương trình.

Bảng 18.1: Các phím điều khiển

Phím	**Mô tả**
Interrupt	Ngắt việc thực hiện chương trình. Khi ra một lệnh cho Linux và bấm <Enter>, chương trình sẽ chạy cho đến khi hoàn tất. Nếu bấm phím ngắt nghĩa là phát tín hiệu cho chương trình dừng lại. Một vài chương trình không nhận ra tín hiệu này khi terminal ở chế độ nóng. UNIX quy ước dùng <Del> làm phím ngắt, nhưng Linux đổi phím này sang <Ctrl-c> để tiện cho những người quen dùng DOS.
Erase	Xoá ký tự cuối cùng trong vùng đệm. Phím xoá hoạt động như phím Backspace trên máy chữ. Một vài terminal và hệ điều hành có lẫn lộn giữa hai phím <Del> và <Backspace>.
Kill	Thường bấm phím @ sẽ xoá toàn bộ vùng đệm trước khi chuyển sang shell hoặc một ứng dụng. Khác khi bấm phím dừng, sẽ không hiện ra dấu nhắc shell, bởi vì driver chờ bạn gõ tiếp vào.
End-of-line	Linux sử dụng phím <Enter> hoặc <Return> làm ký tự «cuối dòng», báo cho driver biết bạn đã gõ xong các ký tự và muốn chúng được diễn dịch rồi chuyển sang shell hoặc chương trình.

End-of-file	Ký tự «cuối tệp» <Ctrl-d>, bắt shell thoát ra và hiển thị dấu nhắc đăng nhập. Linux coi mọi thiết bị như là tệp. Vì terminal là nguồn ký tự vô tận, Linux thường dùng phím <Enter> để báo rằng phiên đăng nhập đến đấy chấm dứt.

Muốn thiết lập và hiển thị các tham số phím điều khiển, bạn gõ lệnh stty, viết tắt của set teletype. Thí dụ terminal của bạn được xác định như là thiết bị tty mang tên tty14. Muốn hiển thị tất cả mọi thiết lập, bạn gõ stty –a. Máy sẽ hiển thị đại loại như sau:

```
speed 38400 baud; rows 25; columns 80; line=0;
intr=^C; quit=^\; erase=^?; kill=^U; eof=^D;
eol=<undef>;
eol2=<undef>; start=^Q; stop=^S; susp=^Z; rprnt=^R; werase=^W;
lnext=^V; flush=^O; min=1; time=0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -tgncr i crnlixon
ixoff
-iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onocr -onlret -ofill -ofdel n10 cr0
tab0bs0 vt0
ff0
isig icanon iexten echo echoe echok -chonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

Bạn để ý thấy trên hệ này, phím interrupt (intr) được định nghĩa là <Ctrl-c> (hiển thị như ^C) và phím kill là <Ctrl-u>. Mặc dù bạn có thể thiết lập lại mọi thứ, song các user thường chỉ khai báo lại hai phím interrupt và kill. Thí dụ muốn chuyển phím kill từ ^U sang ^C bạn gõ:

```
stty kill '^C'
```

Ghi chú: Trong trường hợp terminal có vẻ lỗi, bạn nên chọn cách thiết lập chắc chắn hơn bằng cách gõ lệnh stty sane.

Cần chú ý cả đến dấu nháy khi quy định lại các tổ hợp phím.

Lưu ý: Khi chạy shell bash, nếu muốn máy thực hiện một thiết lập nào đó, bạn hãy ghi lệnh ấy vào tệp .bash_profile (nằm trong home directory của bạn). Đối với shell C, hãy ghi lệnh ấy vào tệp .login. Đối với shell Brourne và Korn thì hãy ghi vào .profile.

18.2.2.2 Thiết lập môi trường shell

Mở phiên làm việc với Linux được gọi là tiến trình đăng nhập và một phần của tiến trình ấy là việc tạo ra môi trường. Những chương trình chạy trên hệ thống Linux được gọi là tiến trình và mỗi tiến trình như thế có một môi trường riêng cho mình, tách bạch và phân biệt hẳn với bản thân chương trình. Nói cách khác, chương trình phải chạy trong một môi trường nhất định. Môi trường Linux được gọi là môi trường shell, bao gồm một số các biến và giá trị của chúng. Các biến và giá trị biến cho phép một chương trình đang chạy, chẳng hạn như shell, xác định rõ được môi trường.

Mỗi môi trường liên quan đến những biến như shell, home directory và loại terminal đang sử dụng. Một số biến như thế đã được xác định trong quá trình đăng nhập nên không thể hoặc không cần thay đổi. Bạn tuỳ ý thêm bớt thay đổi bao nhiêu biến khác cũng được, miễn là chúng không bị đánh dấu "read-only" (chỉ đọc mà thôi).

Các biến được lập trong môi trường dưới dạng BIẾN=giá trị. Ý nghĩa của một BIẾN như thế nào là tuỳ bạn. Tuy nhiên một số biến đã được xác định trước cho các chương trình Linux tiêu chuẩn. Thí dụ biến TERM là tên của loại terminal và đã được xác định trước tại một trong những cơ sở dữ liệu terminal chuẩn Linux. Suốt nhiều năm, hãng Digital Equipment Corporation đã sản xuất một terminal mang tên VT-100. Nhiều nhà sản xuất khác đã sao chép loại terminal rất phổ biến này, hoặc mô phỏng nó bằng phần mềm trên PC. Tên của những loại terminal đó được gọi chung là vt100 và được ghi trong môi trường là TERM=vt100.

Trong môi trường bạn đang sử dụng đã sẵn có nhiều biến được xác định trước. Nếu đang chạy shell C, bạn ra lệnh **printenv** để liệt kê chúng ra xem. Với hai shell Bourne và Korn, bạn ra lệnh **set**. Bảng sau liệt kê các biến môi trường thông dụng nhất. Cột "Biến" bao gồm những gì bạn cần gõ vào tại dòng lệnh.

Bảng 18.2: Các biến môi trường phổ biến của shell Bourne

Biến	**Mô tả**
HOME=/home/đăng_nhập	HOME lập home directory của bạn, đăng_nhập là ID đăng nhập. Thí dụ nếu ID đăng nhập của bạn là pla, thì HOME sẽ là /home/pla
LOGNAME=đăng_nhập	Máy sẽ tự lập LOGNAME bằng ID đăng nhập của bạn
PATH=đường_dẫn	Tuỳ chọn đường_dẫn trỏ đến danh sách các thư mục mà shell sẽ duyệt qua để tìm lệnh. Thí dụ bạn có thể lập đường dẫn: PATH=/usr:/bin:/usr/local/bin
PS1=dấu_nhắc	PS1 là dấu nhắc shell đầu tiên, yêu cầu bạn xác định hình dáng của dấu nhắc riêng theo ý mình. Nếu bạn không có thay đổi gì, dấu nhắc mặc định sẽ là dấu $. Nếu gõ PS1=Enter Command>, máy sẽ hiển thị dấu nhắc dòng lệnh của bạn thành Enter Command>
PWD=thư_mục	Máy tự động lập PWD, xác định vị trí của bạn trong hệ thống tệp. Thí dụ nếu bạn kiểm tra PWD (bằng cách gõ **echo** $PWD tại dòng lệnh) và máy báo /usr/bin, nghĩa là bạn đang ở tại thư mục ấy. Lệnh pwd cũng hiển thị thư mục hiện hành
SHELL=shell	SHELL xác định vị trí của chương trình dùng làm shell. Thí dụ có thể lập SHELL trong tệp .bash_profile hoặc .login là SHELL=/bin/ksh để lấy Korn làm shell.
TERM=loại_terminal	Lập tên của loại terminal sẽ dùng. Thí dụ bạn lập TERM trong tệp .bash-profile hoặc .login là TERM=vt100

Ghi chú: Khi muốn lập biến môi trường theo từng phiên đăng nhập, bạn hãy xác định chúng trong tệp .bash-profile (nếu chạy shell bash), trong tệp .login (nếu chạy shell C) và trong tệp .profile (nếu chạy shell Bourne).

Có lẽ PATH là biến quan trọng nhất của môi trường. PATH chứa các đường dẫn được cách nhau bằng những hai dấu hai chấm và chúng trỏ về mọi thư mục chứa những chương trình bạn đang sử dụng. Máy sẽ căn cứ vào thứ tự của những đường dẫn đó để tìm kiếm. Thứ tự này là quan trọng đối với các hệ thống chấp nhận nhiều hình thức của cùng một câu lệnh. Hệ thống của bạn có thể gồm nhiều lệnh tại chỗ (local). Thí dụ biến PATH có thể chứa những giá trị như sau:

```
/usr/ucb:/bin:/usr/bin:/usr/local/bin
```

Căn cứ vào đấy, trước tiên máy sẽ tìm kiếm trong thư mục /usr/ucb. Nếu tìm thấy câu lệnh cần thiết, shell sẽ ngưng tìm kiếm và bắt đầu thi hành lệnh. Hai thư mục /bin và /usr/bin chứa mọi lệnh Linux tiêu chuẩn. Thư mục /usr/local/bin chứa các lệnh tại chỗ do các user tạo riêng hoặc lấy từ nơi khác về, thông thường chính quản trị viên là người bổ sung những lệnh tại chỗ vào hệ thống.

Nếu là quản trị viên, hoặc nếu muốn truy cập những lệnh có quan hệ đến hệ thống hơn, bạn sẽ bổ sung /usr/sbin hoặc /usr/local/sbin vào, hoặc cả hai, để khỏi phải gõ /usr/sbin/traceroute.

Nếu định tạo ra những lệnh riêng cho mình, bạn có thể chỉnh sửa biến PATH để ghi thêm vào các đường dẫn trỏ đến các thư mục chứa những lệnh riêng ấy. Tuỳ vào shell đang sử dụng, bạn có nhiều cách làm khác nhau. Thí dụ nếu đang chạy shell Bourne hoặc Korn, bạn thêm một đường dẫn mới vào biến PATH bằng cách gõ như sau tại dấu nhắc lệnh:

```
$ PATH=$PATH:đường_dẫn_mới:
```

Khi gõ dấu $ trước tên một biến, giá trị hiện hành của biến này sẽ được gán vào, do đó trong thí dụ trên, giá trị **$PATH** được gán chính là đường dẫn hiện hành. Dấu hai chấm và tham số đường_dẫn_mới được thêm vào sau đường dẫn hiện hành để tạo ra một giá trị mới cho biến PATH. Nếu ngay lúc này bạn tham khảo đến giá trị PATH (dùng lệnh **echo** $PATH) thì vẫn nhận được giá trị cũ. Bạn phải dùng lệnh **export** PATH để xác lập giá trị mới cho lệnh PATH.

Mục tiếp theo sẽ mô tả những cách thức khác dùng để thao tác các biến môi trường. Đến đây, bạn có thể kết luận sơ bộ rằng môi trường shell chứa các biến và hàm mà bạn thao tác được bằng cách dùng shell và các chương trình ứng dụng.

18.2.2.3 Sử dụng các biến shell đặc biệt

Bạn có thể dùng lệnh env để liệt kê những biến shell có sẵn trong môi trường làm việc. Sau đây là danh sách một số các biến shell đặc biệt:

```
HOME=/usr/wrev
SHELL=/bin/sh
MAIL=/usr/mail/wrev
LOGNAME=wrev
PATH=/bin:/usr/bin:.
```

```
TZ=PST8PDT
PS1="[\u@\h\W]\$"
TERM=linux
```

Việc thao tác các biến đặc biệt cũng giống như đối với các biến shell khác. Bảng sau sẽ mô tả ý nghĩa của những biến đặc biệt này.

Bảng 18.3: Các biến môi trường đặc biệt

Tên biến	**Ý nghĩa**
HOME	Tên đường dẫn đầy đủ về home directory của bạn.
SHELL	Tên của shell hiện hành.
MAIL	Tên đường dẫn đầy đủ của hộp thư.
LOGNAME	Tên đăng nhập của bạn.
PATH	Những thư mục mà shell sẽ duyệt tìm lệnh.
TZ	Múi giờ cho lệnh date.
SECONDS	Số giây kể từ khi gọi shell.
PS1	Dấu nhắc hệ thống.
TERM	Loại terminal đang sử dụng.

18.2.2.3.1 Biến HOME

Biến HOME xác định home directory của bạn. Khi đăng nhập xong, bạn được vào ngay trong home directory. Sau đó bạn có thể dùng lệnh cd để chuyển sang những thư mục khác. Thí dụ để chuyển sang thư mục /usr/local/games, bạn gõ:

```
cd /usr/local/games
```

Muốn trở về home directory, bạn chỉ cần gõ:

```
cd
```

Bạn có thể dùng biến HOME để xác định những tệp trong home directory khi biên soạn shell script.

Thay vì viết lệnh như:

```
grep $number /usr/wrev/office/data.01
```

tốt hơn bạn nên viết:

```
grep $number $HOME/office/data.01
```

bởi vì dòng lệnh dễ đọc hơn và nếu di dời home directory thì lệnh vẫn có giá trị.

$HOME luôn đại diện cho home directory của bất kỳ ai đang sử dụng lệnh. Nếu bạn gõ lệnh bằng $HOME thì những người khác cũng có thể dùng chung lệnh.

18.2.2.3.2 Biến PATH

Biến PATH liệt kê các thư mục nơi shell sẽ đến tìm những câu lệnh trong đó. Shell duyệt tìm các thư mục theo thứ tự đã liệt kê.

Nếu PATH=/bin:/usr/bin/:., mỗi khi diễn dịch một câu lệnh, shell sẽ tìm nó trước tiên trong thư mục /bin. Nếu chưa phát hiện ra lệnh cần tìm, shell tiếp tục duyệt sang thư mục /usr/bin. Nếu vẫn chưa có kết quả, shell lại dò sang thư mục hiện hành (.). Giả dụ bạn gõ lệnh **cal** để in lịch tháng này, shell sẽ tìm trước tiên trong /bin. Vì lệnh **cal** không có ở đây, shell tìm tiếp tại /usr/bin và thấy **cal** ở đó.

Lưu ý: Nếu bạn đã biên soạn một lệnh riêng và đặt tên là **cal** thì shell sẽ không bao giờ tìm ra lệnh này. Bởi vì cho dù bất kỳ lúc nào bạn gõ lệnh **cal** thì shell cũng đi tìm trong /usr/bin. Do đó khi biên soạn lệnh riêng cho mình, bạn đừng đặt trùng tên với những lệnh của hệ thống.

Bạn nên xếp mọi shell script của mình vào một thư mục và ghi đường dẫn đến đó vào biến PATH. Như thế sau này cho dù bạn đang ở thư mục nào thì cũng thực thi được những shell script ấy. Công việc sắp xếp shell script gồm các bước như sau:

Tạo một thư mục thứ cấp bin trong home directory của bạn để chứa script, thí dụ:

```
mkdir $HOME/bin
```

Chuyển từng shell script sang thư mục thứ cấp mới tạo. Thí dụ để chuyển script có tên là stamp sang thư mục thứ cấp bin, bạn gõ lệnh:

```
mv stamp $HOME/bin
```

Thêm thư mục này vào biến PATH bằng lệnh PATH=$PATH:$HOME/bin trong tệp .bash_profile để mỗi lần đăng nhập hệ thống thì thay đổi vừa rồi có hiệu quả ngay.

Bạn chỉ tạo thư mục thứ cấp mới này và chỉnh sửa biến PATH có một lần thôi. Với Linux, thư mục /usr/local/bin được tạo ra để lưu các lệnh "tại chỗ" (local), cùng mọi script nào không thuộc gói phần mềm tiêu chuẩn Linux nhưng được quản trị viên thêm vào để cho các user khác dùng chung. Trong trường hợp kể trên, bạn hiểu rằng /usr/local/bin cũng là một phần của PATH.

18.2.2.3.3 Biến MAIL

Biến MAIL chứa tên thư mục lưu trữ e-mail của bạn mỗi khi bạn nhận thư. Nếu có chương trình thông báo việc thư đến, chương trình này sẽ liên hệ tới thư mục đó.

18.2.2.3.4 Biến PS1

Biến PS1 chứa những chuỗi ký tự nhìn thấy tại dấu nhắc sơ khởi. Dấu nhắc là chuỗi ký tự mà shell hiển thị khi sẵn sàng nhận lệnh. Bạn sẽ biết cách thay đổi biến này và những biến khác vào cuối chương này, tại mục "Căn chỉnh các shell Linux".

18.2.2.3.5 Biến TERM

Biến TERM dùng để nhận dạng loại terminal đang sử dụng. Những chương trình nào chạy ở chế độ toàn trang, thí dụ như **vi,** sẽ cần tham khảo biến TERM.

18.2.2.3.6 Biến TZ

Biến TZ chứa thông tin để nhận dạng múi giờ. Một số chương trình như date cần loại thông tin này.

Hệ thống máy tính của bạn giữ nhịp thời gian theo múi giờ quốc tế GMT. Nếu được lập theo PST&PDT, biến TZ sẽ cho múi giờ của khu vực Thái Bình Dương (PST) cách GMT tám múi về phía Tây và hỗ trợ cách đổi giờ PDT (giờ Thái Bình Dương tự động đổi theo mùa để tiết kiệm điện). Máy của bạn sẽ tự động hoán đổi giữa giờ tiết kiệm điện và giờ tiêu chuẩn.

18.2.2.3.7 Biến LOGNAME

Biến LOGNAME chứa chuỗi ký tự mà hệ thống dùng để nhận dạng user đăng nhập. Biến này còn giúp hệ thống biết bạn là chủ sở hữu các tệp, là người ra lệnh chạy một số chương trình và là tác giả của e-mail gửi bằng lệnh **write.**

Thí dụ sau đây là phần mở rộng của **safrm**, một shell script dùng để hỗ trợ xoá tệp một cách an toàn. Biến LOGNAME dùng để xoá mọi tệp thuộc quyền sở hữu của bạn đang nằm trong /tmp. Để làm việc này, shell script dùng lệnh **find** như sau:

```
find /tmp –user $LOGNAME –exec rm { } \;
```

Tham số đầu tiên /tmp có nghĩa là tìm trong thư mục /tmp.

Tuỳ chọn –user cho máy biết phải tìm những tệp thuộc user nào.

Trước khi thi hành lệnh, shell sẽ thay thế $LOGNAME bằng tên đăng nhập của user hiện hành.

Tuỳ chọn –exec cho máy biết rằng lệnh này được áp dụng cho mọi tệp mà lệnh **find** tìm ra. Trong thí dụ này chương trình **rm** dùng để xoá chúng. Hai ngoặc móc { } chỉ vị trí của mỗi tên tệp chuyển sang cho lệnh **rm.**

Hai ký tự \ và ; là do lệnh **find** đòi hỏi. (Dùng dấu sổ ngược \ để chuyển trực tiếp ký tự sang cho chương trình mà không cần đưa cho shell diễn dịch). Khi thêm dòng lệnh kể trên vào shell script trong danh sách 18.1, bạn sẽ được một chương trình xoá tệp an toàn, đồng thời dọn sạch những gì thuộc quyền sở hữu của một user trong thư mục /tmp, nếu chúng đã cũ hơn 10 ngày.

Danh sách 18.1 Shell script safrm

```
# Name: safrm
# Purpose: copy files to directory /tmp, remove them
# from the current directory, clean up /tmp,
# and finally send mail to user
# first copy all parameters to /tmp
cp $* /tmp
# remove the files
rm $*
# create a file to hold the mail message
# The file's name is set to msg
# followed by process ID number of this process
# For example, msg1208
msdfile=/tmp/msg$$
```

```
# construc mail message
date>$msgfile
echo These files were deleted from /tmp>>$msgfile
# get list of files to be deleted from tmp
# -mtime +10 gets all files that haven't been
# modified in 10 or more days, -print displays the names.
find /tmp –user $LOGNAME –mtime +10 –print>>$msgfile
# remove the appropriate files from /tmp
find /tmp –user $LOGNAME –mtime +10 –exec rm {} \;
# mail off the message
mail $LOGNAME<$msgfile
# clean up
rm $msgfile
```

18.2.3 Tìm hiểu các tiến trình

Mỗi chương trình đang chạy trong Linux được gọi là một tiến trình. Vì Linux là hệ đa nhiệm multitasking nên nhiều tiến trình có thể chạy cùng lúc. Để phân biệt, Linux cấp cho mỗi tiến trình một ID riêng biệt gọi là ID tiến trình (PID – Process Identification)

ID tiến trình chỉ là một con số định danh. Muốn biết mỗi ID được kết hợp với tiến trình nào, bạn gõ lệnh ps. Muốn xem phần lớn các ID đang hiện diện trên hệ thống, bạn thêm cờ –guax và máy sẽ hiển thị đại loại như sau:

```
USER PID %CPU %MEM SIZE RSS TTY StaT Start TIME COMMAND
pla  53  3.2  7.0  352  468 p 1  S   02:01 0:01 –bash
pla  65  0.0  3.5   80  240 p 1  R   02:01 0:00 ps –guax
root 1   0.8  3.1   44  208 con  S   02:00 0:00 init
root 6   0.0  1.8   24  124 con  S   02:00 0:00 bdflush (daemon)
root 7   0.0  1.9   24  128 con  S   02:01 0:00 update(bdflush)
root 40  10.  3.5   65  240 con  S   02:01 0:00 /usr/sbin/syslogd
root 42  0.2  2.9   36  200 con  S   02:01 0:00 /usr/sbin/klogd
root 44  0.5  3.2   68  216 con  S   02:01 0:00 /usr/sbin/inetd
root 46  0.2  3.0   64  204 con  S   02:01 0:00 /usr/sbin/lpd
root 52  0.1  2.0   32  140 con  S   02:01 0:00 selection –tms
root 58  0.2  2.4   37  164 p 6  S   02:01 0:00 /sbin/agetty38400 tt
```

ID tiến trình nằm ở cột PID. Dòng in đậm cho biết đây là tiến trình đầu tiên khi hệ thống khởi động (init). Tiến trình init sẽ được mô tả ở một mục sau.

Khi bạn bắt Linux chạy một chương trình nào đó (có nghĩa là tạo ra một tiến trình nào đó) Linux sẽ chép một bản sao chính xác của chương trình có tín hiệu yêu cầu.

Trong trường hợp đơn giản nhất, bạn ra lệnh cho shell chạy một chương trình, từ đó shell mới gửi yêu cầu fork đến lõi Linux.

18.2.3.1 Fork và các tiến trình init, exec

Fork sẽ nhân bản một tiến trình đã có sẵn. Linux tạo các tiến trình mới bằng cơ chế forking (phân nhánh). Nhân bản một tiến trình có nghĩa là tạo ra bản sao chính xác của tiến trình ấy (bao gồm cả môi trường và các tệp đã mở có liên quan). Ngoài ra, còn có một cờ giúp tiến trình đã được nhân bản phân biệt được đâu là chương trình mẹ (hay tiến trình mẹ, parent process), đâu là chương trình đã được sao chép (tiến trình con, child process).

Vì mọi tiến trình đều được tạo cùng một cách thức như thế cho nên mỗi tiến trình đều có tiến trình mẹ và một ID tiến trình mẹ. Bất kỳ tiến trình nào chạy trên Linux đều có thể truy ngược dòng họ của mình về đến init, vốn là mẹ của mọi tiến trình. Bản thân init, với ID tiến trình 1, là tiến trình duy nhất chạy trực tiếp bằng lõi Linux mà một user có thể tiếp xúc được. Mỗi tiến trình mà bạn tạo ra trong một phiên làm việc đều lấy shell đăng nhập làm tổ tiên và shell đăng nhập lại lấy init là bậc trên của mình. Sau khi khâu nhân bản hoàn tất, tiến trình con sẽ gọi chương trình exec để tự biến mình thành tiến trình mà bạn vừa yêu cầu. Điều duy nhất thay đổi sau khi exec làm việc là ID của tiến trình đang chạy. Môi trường của tiến trình mới là bản sao trung thực môi trường của tiến trình mẹ.

18.2.3.2 Xuất nhập tiêu chuẩn

Mỗi tiến trình mới đều được tạo ra bằng ba "tệp" mở. Vì Linux xử lý tệp và các thiết bị như nhau, do đó một "tệp" mở có thể là một tệp dữ liệu hoặc một (cổng) thiết bị chẳng hạn như terminal. Ba tệp mở ấy được gọi là nhập tiêu chuẩn (stdin, standard input), xuất tiêu chuẩn (stdout, standard out) và báo lỗi tiêu chuẩn (stderr, standard error output).

Mọi lệnh Linux, cũng như những chương trình ứng dụng, đều nhận dữ liệu nhập từ stdin và đưa dữ liệu xuất ra stdout. Và máy sẽ tự động gửi thông báo lỗi ra stderr.

Khi bạn đăng nhập, các tệp stdin, stdout, stderr được kết hợp với terminal bạn đang dùng. Bất kỳ chương trình nào bạn chạy (nghĩa là bất kỳ tiến trình nào bạn tạo ra) đều thừa hưởng terminal của bạn cũng như ba tệp mở nói trên.

18.3 Phân tích lệnh trong shell

Parsing (phân tích) là thao tác chia tách dòng lệnh (hoặc những gì bạn gõ vào) ra thành từng thành tố riêng rẽ để xử lý.

Với Linux, parsing còn tiến hành nhiều thao tác khác nữa, chứ không đơn thuần chỉ là tách dòng lệnh ra. Thoạt tiên, Linux tách chuỗi lệnh ra từng phần với những việc như: xem xét tên tệp có dùng ký tự đại diện (wildcard) hay không, mở rộng các biến shell, thiết lập chuyển hướng vào/ra (I/O), lập shell thứ cấp (shell con) hoặc nhóm lệnh và thay thế lệnh. Chỉ sau khi thực hiện xong một loạt những thao tác vừa kể thì Linux mới thi hành dòng lệnh bạn vừa gõ vào.

Nếu những cụm từ như ký tự đại diện và chuyển hướng I/O có vẻ mới đối với bạn, phần sau của chương này sẽ giúp bạn hiểu. Tạm thời, mời bạn hãy làm quen với cú pháp căn bản của các lệnh.

18.3.1 Sử dụng lệnh, cờ và tham số

Muốn thi hành một lệnh Linux, bạn chỉ cần gõ tên của tệp. Muốn liệt kê các tệp, bạn gõ lệnh **ls** nằm trong thư mục /bin. Nếu /bin được liệt kê trong biến PATH (và nên được liệt kê ở đấy), shell sẽ tìm ra /bin/**ls** và thi hành lệnh này.

Một vài lệnh trong Linux không phải là tệp độc lập, chúng được gắn liền trong shell. Thí dụ lệnh cd (chuyển thư mục) được gắn liền trong hầu hết các shell và shell sẽ thi hành nó ngay được vì không cần tìm kiếm bên ngoài. Muốn biết lệnh nào được thi hành trong shell hoặc ngoài shell, bạn tìm đọc các trang **man** cho shell ấy. Một vài shell có một tệp lệnh riêng để chứa những lệnh thi hành trực tiếp.

18.3.1.1 Cờ

Muốn một lệnh được thi hành đúng đắn, bạn phải gửi lệnh ấy đến shell một cách đúng đắn. Bản thân tên của lệnh phải đứng đầu dòng lệnh, theo sau là các cờ hoặc tham số. Cờ (còn gọi là tuỳ chọn) là những chữ cái đứng sau dấu gạch ngang. Cờ sẽ điều chỉnh hành động của một lệnh.

Thí dụ lệnh **ls** đơn giản chỉ liệt kê các tên tệp trong thư mục hiện hành theo thứ tự abc. Nếu thêm vào vài cờ, **ls** sẽ liệt kê nội dung thư mục bằng nhiều cách khác nhau.

Thí dụ **ls** có thể liệt kê mọi tệp với tất cả thuộc tính của chúng bằng cờ "dài" là –l. Do đó bây giờ cú pháp lệnh liệt kê sẽ trở thành:

```
ls -l
```

-l chính là cờ. Nếu muốn dùng nhiều cờ trong cùng dòng lệnh, bạn chỉ việc viết chúng nối đuôi nhau, chẳng hạn như **ls** –lF. Cờ –F sẽ hiển thị một dấu sao nếu đây là tệp khả thi, hiển dấu a vòng (@) nếu tệp là kết nối tượng trưng và dấu sổ ngược / nếu đó là thư mục thứ cấp.

Những trang **man** cho từng lệnh thường liệt kê mọi cờ điều chỉnh cùng với ý nghĩa của chúng trước khi mô tả các tham số. Bạn cũng có thể gõ từng cờ riêng ra, shell sẽ phân tích chúng trước khi chuyển cho chương trình. Thí dụ bạn có thể viết lệnh **ls** –lF thành **ls** –l –F.

Ghi chú: Linux có một tính năng được ưa thích, đó là thể hiện bằng màu. Khi bạn gõ lệnh **ls**, Red Hat Linux sẽ hiển thị tệp bằng nhiều màu khác nhau, tuỳ theo loại của tệp, giúp bạn biết ngay đó là tệp loại thi hành, thư mục, hoặc kết nối. Ngoài ra nếu chuyển hướng dòng đầu ra của **ls** sang một tệp, tệp này sẽ chứa các mã điều khiển dùng để chỉ màu. Mã màu có thể xung đột với các chương trình khác, chẳng hạn như **less**, nếu tệp này sử dụng **less.**

Một vài loại cờ báo trước rằng tham số tiếp theo sẽ có ý nghĩa đặc biệt. Thí dụ cờ –t trong lệnh **sort** cho biết ký tự tiếp theo sẽ là ký tự phân cách trường. Nếu muốn sắp xếp tệp /etc/passwd (nơi các trường được phân cách bằng dấu hai chấm), bạn gõ:

```
sort -t: /etc/passwd
```

Với lệnh **sort**, cờ –t chỉ cần thiết trong trường hợp tệp sử dụng một dấu phân cách trường khác với dấu mặc định. Dấu mặc định để phân cách trường được xác định trong biến môi trường IFS (Inter Field Separator).

Shell dùng biến IFS tách dòng lệnh ra để biết rằng mình nên dùng dấu tiêu chuẩn phân cách trường, hay phải dùng dấu khác theo điều chỉnh của –t.

18.3.1.2 Tham số

Trên dòng lệnh, các cờ phải đứng trước mọi tham số khác. Tham số là những chuỗi ký tự được phân cách bởi các ký tự đặc biệt xác định trong biến môi trường IFS. Những chuỗi ký tự mặc định giữa các IFS là dấu trắng, dấu tab và ký tự "dòng mới" (newline hay CR). Bạn có thể đặt bao nhiêu ký tự phân cách trường giữa các tham số cũng được.

Khi tách dòng lệnh, shell sẽ giảm các ký tự này xuống thành một ký tự duy nhất trước khi tiến hành xử lý. Thí dụ nếu một lệnh được tiếp theo sau bởi ba dấu trống (SP), một dấu tab và sau đó là tham số đầu tiên, shell sẽ tự động giảm ba dấu trống và dấu tab thành một dấu tab duy nhất. Do đó dòng:

```
lệnh<dấu trống><dấu trống><dấu trống><Tab>tham_số
```

sẽ trở thành:

```
lệnh<Tab>tham_số
```

với tham_số thường là tên tệp hoặc chuỗi ký tự báo cho lệnh biết cần thi hành gì. Nếu tham số có kèm theo một dấu trống, cả chuỗi phải được đặt trong ngoặc kép để shell không triển khai chuỗi ra. Thí dụ dòng lệnh sau đây được hiểu là có hai tham số; shell sẽ phải đi tìm chữ New trong tệp mang tên York:

```
grep New York
```

Nhưng có lẽ bạn muốn tìm cụm từ "New York" trong stdin, vậy phải gõ lệnh thành:

```
grep "New York"
```

Trong trường hợp này, chuỗi "New York" được chuyển sang lệnh **grep** như là một tham số duy nhất.

18.3.2 So sánh tên tệp

Hầu hết các hệ điều hành hiện đại (bao gồm mọi phiên bản Linux) đều chấp nhận việc sử dụng ký tự đại diện (wildcard) khi tìm tên tệp hoặc chuỗi ký tự. Bảng sau tóm lược cách dùng các ký tự đó.

Bảng 18.4: Các ký tự đại diện

Ký tự	Ý nghĩa
*	Đại diện bất kỳ nhóm ký tự nào, ngoại trừ dấu chấm khi dấu chấm ấy là ký tự đầu tiên của tên tệp. Thí dụ lệnh **cat** office*>alloffice sẽ ghép mọi tệp có tên bắt đầu bằng office vào một tệp mang tên alloffice
?	Đại diện một ký tự đơn lẻ. Thí dụ lệnh **lp** office.9? sẽ hiển thị danh sách các tệp có tên dạng office.yy, với yy là một năm trong thập niên 90, chẳng hạn như office.97, office.98, v.v.
[]	Đại diện cho một ký tự có trong khoảng đó, chẳng hạn lệnh **rm** office.9[7-8] sẽ xoá những tệp nào có tên là office. 97 và office.98

Ghi chú: Nếu bạn đặt một ký tự hoặc biểu thức wildcard bên trong ngoặc kép, phần đuôi của tên tệp sẽ không được tính đến khi máy phân tích dòng lệnh. Thí dụ nếu bạn gõ **ls*** (viết liền), bạn sẽ có mọi tệp trong thư mục hiện hành. Nhưng nếu gõ **ls** * (có dấu trống), máy sẽ báo lỗi file not found bởi vì cho rằng phải tìm tệp mang tên *.

18.3.2.1 wildcard hình sao

Ký tự wildcard hình sao (*) là dạng được sử dụng rộng rãi nhất. Dấu * có nghĩa là tất cả và bất kỳ ký tự nào.

Thí dụ chuỗi a* có nghĩa là tất cả những tệp nào bắt đầu bằng chữ a. Trong một biểu thức bạn có thể dùng nhiều sao để xác định bao nhiêu nhóm tệp cũng được.

Thí dụ biểu thức *xx*.gif có nghĩa là tất cả tên tệp nào có đuôi là .gif, đồng thời có xx bất kỳ nơi nào ở phần còn lại. Do đó những tên tệp như abxx.gif, xxyyzz.gif và xx.gif đều phù hợp.

Bạn dùng dấu sao để đại diện cho bất kỳ chuỗi ký tự nào. Thí dụ muốn in mọi tệp trong thư mục hiện hành có đuôi là .txt, bạn gõ:

```
lp *.txt
```

Hãy cẩn thận khi sử dụng wildcard, bởi vì nếu bạn gõ lệnh sau đây, tất cả những tệp nào có tên tận cùng bằng txt đều sẽ được in:

```
lp *txt
```

Tệp mang tên reportxt sẽ phù hợp thí dụ thứ hai, nhưng sẽ không được tính đến với thí dụ thứ nhất. Và trong thí dụ thứ ba sau đây, shell sẽ chuyển tên những tệp trong thư mục, cũng như bản thân tệp mang tên txt đến lệnh **lp** (tệp mang tên txt trong thư mục của bạn sẽ được chuyển hai lần đến **lp**):

```
lp * txt
```

Ở thí dụ này, giữa * và txt có một dấu trống, do đó trước tiên lệnh **lp** sẽ in những tệp nào được đại diện bởi ký tự * (có nghĩa là mọi tệp trong thư mục hiện hành). Tiếp theo **lp** sẽ "quan tâm" đến đối tượng thứ hai trong danh sách phải hiển thị lên (Linux hiểu ký tự dấu trống giữa * và txt như là một dấu phân cách). Do đó **lp** sẽ xử lý txt như là tên của tệp tiếp theo phải in.

Ký tự * có thể được dùng bất kỳ chỗ nào trong chuỗi ký tự. Thí dụ nếu muốn dùng lệnh **ls** để liệt kê tất cả tệp trong thư mục hiện hành có tên chứa cụm ký tự rep, bạn gõ như sau:

```
lp *rep*
```

Linux sẽ liệt kê các tệp mang tên chẳng hạn như frep.data, report và janrep. Một ngoại lệ duy nhất là tên tệp bắt đầu bằng dấu chấm sẽ không được liệt kê. Muốn liệt kê tên loại tệp này (thường được gọi là tệp ẩn), bạn phải xác định dấu chấm đứng đầu. Thí dụ nếu muốn liệt kê tệp mang tên .reportrc, bạn gõ:

```
ls .*rep*
```

Cẩn thận: Hãy hết sức cẩn thận khi dùng dấu sao để xoá (deleting) hoặc gỡ bỏ (removing) tệp. Lệnh **rm** * sẽ gỡ bỏ mọi tệp trong thư mục hiện hành. Lỗi rất thường gặp là lỡ xoá toàn bộ các tệp trong khi chỉ muốn xoá một nhóm tệp nào đó mà thôi.

Thí dụ thay vì gõ **rm** *txt (gỡ bỏ những tệp có tên tận cùng bằng txt), lệnh bị gõ nhầm thành **rm** * txt. Linux sẽ xoá toàn nhóm tệp, xong rồi lại cố gắng xoá tệp mang tên txt dù rằng đến lúc này dĩ nhiên trong thư mục không còn tệp nào để xoá cả.

Để đề phòng, bạn nên dùng tuỳ chọn –i với **rm** khi sử dụng dấu *. Lệnh **rm** –i *txt sẽ hiện dấu nhắc chờ khẳng định của bạn trước khi tiến hành xoá bỏ.

18.3.2.2 wildcard hình chấm hỏi

Dấu wildcard chấm hỏi đại diện cho một ký tự đơn lẻ. Giả sử thư mục hiện hành của bạn có các tệp như report1, reportb, report10, reportb3, report.dft và report.fin. Vì biết lệnh **lp** rep* sẽ in mọi tệp đó, nhưng bạn chỉ muốn in hai tệp đầu (report1 và reportb) nên gõ:

```
lp report?
```

Để liệt kê các tệp có tên dài ba ký tự và tận cùng bằng ký tự x, bạn gõ:

```
ls ??x
```

Lệnh này sẽ liệt kê tệp tax, nhưng không liệt kê tệp trax. Vì dấu chấm hỏi chỉ đại diện cho một lần xuất hiện của một ký tự nào đó, cho nên chuỗi ??? sẽ tượng trưng cho những tệp nào có tên dài ba ký tự. Bạn có thể tạo ra danh sách những tệp có đuôi (phần mở rộng) gồm ba ký tự bằng chuỗi *.???. Thí dụ muốn duyệt tìm trong thư mục xem có tệp đồ hoạ nào có đuôi là .gif, .tif và .jpg, cùng với những tệp khác có đuôi ba ký tự, bạn gõ:

```
ls *.???
```

Ghi chú: Xin nhắc bạn rằng Linux khác DOS và tên tệp Linux không chỉ hạn chế trong vòng tám ký tự. Và tên tệp của Linux phân biệt chữ hoa với chữ thường (**case-**sensitive)

18.3.2.3 Biểu thức []

Đôi lúc bạn cần có những lựa chọn chính xác hơn là những gì mà wildcard cho phép. Giả sử bạn muốn chọn các tệp job1, job2, job3, nhưng không chọn jobx. Bạn không thể dùng wildcard kiểu dấu hỏi vì nó đại diện cho một lần xuất hiện duy nhất của một ký tự bất kỳ. Tuy nhiên bạn có thể gõ job[123] là được.

Bạn cũng có thể mô tả một ký tự đơn lẻ bằng cách gộp dãy ký tự chứa nó vào trong ngoặc vuông. Muốn liệt kê các tệp có tên bắt đầu bằng chữ hoa, bạn gõ lệnh:

```
ls [A-Z]*
```

Giả sử bạn có các tệp office.90, office.91, office.92 và office.93 và muốn chép ba tệp đầu vào một thư mục thứ cấp mang tên oldstuff. Nếu thư mục này đang sẵn có, bạn gõ:

```
cp office.9[0-2] oldstuff
```

Giống như trường hợp wildcard "chấm hỏi", những gì bên trong dấu ngoặc vuông chỉ đại diện cho một ký tự duy nhất. Bạn có thể mô tả một dãy giá trị như [123], để cho phép thay một trong các ký tự 1, 2, hoặc 3. Bạn cũng có thể mô tả một dãy ký tự chẳng hạn như trường hợp [A-Z], dãy này đại diện cho mỗi ký tự nằm giữa A hoa và Z hoa, tính luôn cả A và Z.

Bạn cũng có thể xác định một số dãy khác nhau. Thí dụ muốn chọn các ký tự chữ, bạn viết [A-Z,a-z]. Nên nhớ là bộ mã ASCII chứa nhiều ký tự đặc biệt nằm giữa ASCII Z và ASCII a, do đó nếu bạn viết [A-Z], máy sẽ hiểu luôn cả những ký tự đặc biệt ấy.

18.3.3 Kết nối các tiến trình bằng ống dẫn

Thường khi bạn dùng đầu ra của một chương trình này làm đầu vào cho chương trình khác. Thay vì phải ra từng câu lệnh riêng lẻ rồi sau đó lại phải lưu kết quả vào tệp trung gian, bạn có thể kết nối một dãy lệnh bằng ống dẫn (|).

Thí dụ để sắp xếp một tệp mang tên alloffice, rồi sau đó in ra, bạn gõ:

```
sort alloffice | lp
```

Đầu ra của chương trình nằm phía trước ống dẫn được gửi qua ống dẫn ấy rồi trở thành đầu vào của chương trình nằm phía sau. Bạn có thể kết nối nhiều tiến trình bằng |. Thí dụ muốn sắp xếp, rồi sau đó in một danh sách các dữ liệu trong mọi tệp có tên bắt đầu là office, bạn gõ:

```
cat office* | sort | lp
```

18.3.4 Chuyển hướng xuất và nhập

Có những chương trình sử dụng nhập dữ liệu từ bàn phím, trong khi nhiều chương trình khác lại xuất dữ liệu ra màn hình. Linux kết hợp đầu vào từ bàn phím với tệp mang tên stdin; kết hợp đầu ra màn hình với tệp stdout. Bạn có thể chuyển hướng xuất và nhập để dữ liệu đi từ một tệp này đến một tệp khác, thay vì từ bàn phím đến màn hình.

Muốn chuyển hướng đầu vào cho lệnh hay chương trình như thể đầu vào đến từ tệp thay vì đến từ bàn phím, bạn dùng ký hiệu < (nhỏ hơn). Giả sử bạn muốn **mail** tệp info đến địa chỉ lan_anh. Thay vì phải gõ lại nội dung của tệp cho lệnh **mail**, bạn bắt Linux phải dùng tệp info làm đầu vào (stdin) cho lệnh **mail**:

```
mail lan_anh<info
```

Để chuyển hướng đầu ra của một chương trình đến một tệp, bạn dùng ký hiệu > (lớn hơn). Như thế, thay vì chạy đến màn hình, đầu ra sẽ chạy thẳng đến tệp. Như bạn biết, lệnh **date** hiển thị ngày giờ hiện hành ra màn hình. Nếu muốn chuyển ngày giờ này vào tệp mang tên now, bạn gõ:

```
date>now
```

Cẩn thận: Nếu tên tệp phía sau ký hiệu > là của một tệp có sẵn thì đầu ra từ **date** sẽ ghi đè tệp này. Do đó bạn phải cẩn thận để không làm mất thông tin hữu ích.

Nếu muốn ghép thêm hoặc gộp thêm thông tin vào một tệp có sẵn, bạn dùng hai ký tự >>. Thí dụ muốn ghép thêm ngày giờ hiện hành vào tệp report, bạn gõ:

```
date>>report
```

Bạn tham khảo một thí dụ khác dài hơn như sau: Giả sử tệp office gồm dữ liệu quản trị hành chính; trường thứ nhất của mỗi dòng chứa mã ID client. Dòng lệnh đầu tiên đưa đầu ra của lệnh **date** vào tệp office_report. Dòng lệnh thứ hai dùng tệp office làm đầu

vào cho lệnh **sort**, rồi ghép đầu ra của lệnh **sort** vào tệp office_report. Dòng lệnh cuối cùng gửi tệp office_report đến các user tên lan_anh và thu_oanh bằng đường e-mail:

```
date>office_report
sort<office>>office_report
mail lan_anh thu_oanh< office_report
```

Cẩn thận: Đừng chuyển cùng một tệp đến một lệnh nào đó vừa làm đầu vào vừa làm đầu ra, vì sẽ có nguy cơ phá hỏng nội dung của tệp.

Bảng sau tóm tắt các ký hiệu chuyển hướng được dùng trong Linux.

Bảng 18.5: Các ký hiệu chuyển hướng dùng trong Linux

Ký hiệu	**Ý nghĩa**	**Thí dụ**
<	Lấy đầu vào từ một tệp	**mail** lan_anh<report
>	Gửi đầu ra đến một tệp	**date**>now
>>	Ghép vào một tệp có sẵn	**date**>>report

18.3.5 Thay thế các biến shell

Phần đầu chương này đã giới thiệu với bạn về việc mở rộng các biến shell khi chuyển biến PATH thành PATH=$PATH:đường_dẫn_mới. Shell thay thế $PATH bằng giá trị hiện hành của biến PATH. Shell là một ngôn ngữ diễn dịch thực sự, cũng giống như BASIC. Những biến của shell là đối tượng chính được xử lý. Và bởi vì thường xuyên được xử lý, mỗi shell có phương pháp riêng để thử và xác định các biến.

Biến shell được lưu dưới dạng chuỗi. Khi hai biến được đặt sát nhau, các chuỗi của chúng được gộp chung. Thí dụ bạn có hai biến X=hello và Y=world, biểu thức XY sẽ cho ra chuỗi helloworld. Với lệnh dưới đây, shell sẽ phân tích hai tham số và giá trị của X và Y (hai chuỗi hello và world) sẽ được thay thế trước khi chuyển cho lệnh **echo**:

```
echo $X$Y
```

Sau đó lệnh **echo** sẽ hiển thị lên hello world.

Ghi chú: Dù bạn gõ một loạt ký tự tab giữa $X và $Y, kết quả đầu ra cũng không đổi.

Nếu việc thay thế có vẻ mơ hồ, shell sẽ chọn khả năng thay thế nào hiển nhiên nhất và thường cho ra những kết quả bất ngờ. Thí dụ nếu bạn gõ **echo** $XY, shell sẽ thay bằng giá trị rỗng hay một giá trị có thực của biến XY. Trong trường hợp này nếu bạn sẵn có một biến XY, shell sẽ thay thế bằng giá trị của biến ấy. Hiểu được cách làm của shell đối với một trường hợp "mờ" như thế, bạn sẽ biết cách xác định chính xác những gì mình muốn. Nếu bạn gõ ${X}Y, shell sẽ thay thế giá trị của X trước khi ghép ký tự Y vào chuỗi.

Hai shell Bourne và Korn có nhiều kỹ thuật mở rộng biến shell rất phong phú. Những shell này sẽ tiến hành một loạt thử nghiệm đối với biến trước khi thay thế..

18.3.6 Thay thế kết quả của lệnh

Sau khi thay thế xong các biến và trước khi bảo đảm rằng dòng lệnh đã sẵn sàng chạy, shell sẽ soát lại dòng ký tự được gõ vào xem còn lệnh nào phải thi hành không.

Thay thế lệnh có nghĩa là Linux thay thế kết quả của một lệnh vào chỗ một tham số vị trí. Mời bạn xem thí dụ sau:

```
lệnh-1 tham_số ´lệnh-2´
```

Bạn cẩn thận đừng nhầm lẫn giữa dấu ´ (apostrophe, giống như dấu sắc) và dấu ` (dấu huyền). Bảng sau liệt kê ý nghĩa của mỗi dấu.

Bảng 18.6: Phân biệt các dấu

Dấu	Ý nghĩa
´	Dấu sắc sẽ vô hiệu hoá mọi thao tác phân tích. Những gì nằm bên trong hai dấu sắc đều được coi như là một tham số đơn lẻ.
`	Dấu huyền hàm ý thao tác thay thế lệnh. Những gì nằm giữa hai dấu huyền được thi hành giống như bản thân dòng lệnh được thi hành. Đầu ra nào được đưa ra stdout sẽ thay thế lệnh. Sau đó Linux sẽ phân tích dòng lệnh lần nữa để xem xét các tham số.
″	Dấu nháy kép sẽ vô hiệu hoá việc tạo ra tên tệp và việc mở rộng tham số. Tuy nhiên việc thay thế các biến shell và lệnh vẫn được tiến hành.

Dòng lệnh sau đây:

```
echo Today\ ´s date and time are ` date`
```

sẽ cho đầu ra:

```
Today's date and time are Mon May 18 14:35:09 EST 1999
```

Để cho lệnh **echo** làm việc đúng, cụm 's của từ Today's trong dòng lệnh phải được dấu sổ ngược (\) đi trước(Today\ ´s). Dấu này còn được gọi là ký tự thoát. Bạn nên nhớ rằng mỗi ký tự không phải là chữ đều có ý nghĩa đặc biệt đối với shell. Muốn dùng những ký tự ấy trong một chuỗi và không cho shell diễn giải chúng, bạn phải "thoát" khỏi ký tự ấy – nghĩa là bạn phải ghi dấu sổ ngược vào phía trước ký tự đặc biệt ấy. Nếu muốn dùng chính dấu sổ ngược, bạn gõ ký tự này hai lần (\\). Muốn chuyển một dấu dollar cho lệnh, bạn gõ \$.

18.3.7 Biểu thức thường

Biểu thức thường là một loạt các ký tự tiêu chuẩn đi cùng với các toán tử đặc biệt. Bạn dùng chúng để duyệt tìm một chuỗi ký tự trong tệp. Biểu thức thường có thể dùng với các công cụ thuộc họ **grep**, chẳng hạn như **grep, egrep** và **fgrep**, song vẫn được dùng với các lệnh UNIX khác.

Hình thức đơn giản nhất của một biểu thức thường là một chuỗi. Chuỗi là một cụm ký tự, chẳng hạn như hand. Cú pháp cho lệnh **grep** như sau:

```
grep chuỗi tên_tệp
```

Thí dụ để tìm ra những dòng có chữ hand trong tệp mang tên thu_oanh, bạn gõ lệnh:

```
grep hand thu_oanh.txt
```

Và sẽ nhận được kết quả:

```
on the other hand, thu_oanh has been working hard this week
```

nếu dòng kể trên là dòng duy nhất của tệp có chứa chữ hand, **grep** sẽ hiển thị tất cả những dòng nào của tệp khớp với chuỗi được xác định.

Các biểu thức thường sử dụng những ký tự đặc biệt như các dấu chấm, sao, ngoặc vuông, sổ ngược, mũ và dollar. Bảng sau tóm tắt ý nghĩa của các ký tự đặc biệt ấy trong biểu thức thường.

Bảng 18.7: Các ký tự đặc biệt trong biểu thức thường

Ký tự	**Mô tả**
.	So sánh với một ký tự đơn, trừ khi ký tự đơn đó lại là một dấu xuống dòng. Thí dụ b.d sẽ khớp với bad và bod.
*	Khớp với zero hoặc nhiều hơn nội dung của biểu thức thường đứng trước đó. Vì vậy 4* sẽ khớp với 4s, 1 4, 2 4s, v.v..
[]	Dùng để tập hợp một bộ các bội số để so sánh. Xin nhắc bạn rằng Linux và UNIX phân biệt chữ thường với chữ hoa. Vì thế để duyệt tìm mọi trường hợp của chữ Thu_oanh, bạn có thể dùng [Tt]hu_oanh để tìm ra thu_oanh và Thu_oanh, nhưng sẽ không được kết quả THU_OANH. Nếu muốn tìm ra mọi ký tự], bạn có thể gõ []] hoặc dùng dấu sổ ngược làm ký tự thoát để xử lý ngoặc vuông bên phải như là một ký tự văn bản (\]). Dấu gạch nối bên trong ngoặc vuông sẽ được coi như tượng trưng cho một dãy, do đó [a-j] sẽ được coi như [abcdefghij].
\	Dùng để đánh dấu thoát khỏi ý nghĩa của các ký tự đặc biệt và xử lý chúng như là ký tự văn bản bình thường khi duyệt tìm một chuỗi. Do đó * sẽ khớp với mọi thứ, trong khi * sẽ khớp với một dòng nào đó có dấu *. Và đương nhiên \\ sẽ dùng để tìm ra dấu sổ ngược \.
^	Nếu nằm ở đầu chuỗi, dấu mũ ^ chỉ so sánh với một dòng khi chuỗi nằm ở đầu dòng. Do đó nếu bạn có tệp văn bản gồm các số điện thoại được sắp xếp theo mã vùng, biểu thức thường ^704 sẽ khớp mọi số có mã vùng 704, nhưng sẽ không khớp với số điện thoại 407-555-7043.
$	Nếu là ký tự cuối cùng trong biểu thức thường, dấu $ sẽ so sánh với một dòng với điều kiện là chuỗi nằm ở cuối dòng.

Muốn thể hiện số lượng ký tự được biểu thức xác định, bạn dùng cặp móc nhọn {}. Thí dụ lệnh:

```
g\{3,4}
```

sẽ khớp những dòng nào có ggg hoặc gggg.

Nếu bạn có một tệp lớn chứa các e-mail cũ, lệnh:

```
grep 'whatever' ~/mail/*
```

sẽ duyệt tìm chuỗi whatever trong thư mục **mail**. Việc này rất có ích khi bạn sực nhớ ra rằng cô Lan Anh chẳng hạn, có ghi số điện thoại trong tệp office1 (nhóm văn thư), song bạn lại không biết mình đã để tệp ấy trong thư mục nào. Lệnh:

```
grep 'lan_anh' ~/mail/*
```

Sẽ so sánh tất cả mọi trường hợp có mang tên lan_anh. Song còn có cách tốt hơn để so tìm số điện thoại. Thí dụ số điện thoại ấy mang mã TP Hồ Chí Minh, lệnh:

```
grep '848.[0-9]\{3\}.[0-9]\{4\}' ~/mail/*
```

Sẽ tìm ra mọi số điện thoại bắt đầu bằng 848. Bạn lưu ý các dấu chấm phân cách số 848 với [0-9]\{3\} và [0-9]\{4\}. Dấu chấm này sẽ khớp với từng ký tự đơn, giúp bạn so sánh số điện thoại 848-555-1212 hoặc 848.555.1212, vì một số người sử dụng dấu chấm để phân cách số điện thoại, thay vì dùng gạch nối.

18.3.8 Tìm hiểu nhóm lệnh, shell thứ cấp và các lệnh khác

Bạn kết thúc lệnh thường bằng một dấu xuống dòng. Nếu muốn đặt thêm lệnh ở dòng lệnh trước khi bấm <Enter>, bạn có thể tạo ra một nhóm lệnh bằng cách dùng dấu chấm phẩy phân cách những lệnh riêng rẽ. Khi phân tích dòng lệnh, shell sẽ xem dấu chấm phẩy là dấu hết dòng. Ở thí dụ sau đây, shell sẽ thi hành từng lệnh một, mặc dù bạn gõ các lệnh vào chung một dòng:

```
lệnh-1;lệnh-2;lệnh-3
```

Chẳng hạn bạn gõ **clear**;**ls** để xoá màn hình, sau đó mới hiển thị thư mục.

18.3.8.1 Nhóm lệnh

Muốn chuyển hướng đầu ra hoặc đầu vào cho một nhóm lệnh chung, bạn biến dòng lệnh thành nhóm lệnh. Nhóm lệnh là một số lệnh bao hàm giữa hai dấu móc nhọn {}. Thí dụ chuỗi lệnh sau đây sẽ chuyển đầu ra của mọi lệnh về tệp mang tên output-file:

```
(lệnh-1;lệnh-2)>output-file
```

Bạn có thể dùng bất kỳ hình thức chuyển hướng nào, chẳng hạn như đưa đầu ra một nhóm lệnh vào ống dẫn như sau:

```
(lệnh-1;lệnh-2)|lệnh-3
```

Ở trường hợp này, đầu ra của lệnh-1 được đưa vào ống dẫn, sau đó đầu ra của lệnh-2 cũng được đưa vào cùng ống dẫn ấy và lệnh-3 xem đó là một dòng dữ liệu liên tục.

Ghi chú: Những lệnh được thi hành trong một nhóm lệnh sẽ chạy trong shell hiện hành. Nghĩa là chúng có khả năng thay đổi môi trường và chuyển thư mục.

18.3.8.2 Shell thứ cấp

Khi bạn chạy một loạt lệnh trong nhóm lệnh, bạn sẽ hoạt động trong shell hiện hành. Nếu một trong những lệnh này thay đổi môi trường hoặc chuyển thư mục, thì những thay đổi này chỉ có hiệu quả khi nào cả nhóm lệnh đã xong nhiệm vụ. Nếu muốn tránh trường hợp này, bạn có thể chạy nhóm lệnh trong một shell thứ cấp.

Shell thứ cấp là bản sao nhân bản của shell hiện hành, nhưng bởi vì các tiến trình con không thể thay đổi môi trường của tiến trình mẹ cho nên mọi lệnh chạy trong shell thứ cấp sẽ không ảnh hưởng đến môi trường khi nhóm lệnh xong việc. Muốn chạy nhóm lệnh trong shell thứ cấp, bạn thay thế cặp móc nhọn bằng cặp ngoặc đơn. Nhóm lệnh của thí dụ ở mục trước sẽ được viết lại như sau:

```
(lệnh-1;lệnh-2)|lệnh-3
```

Do đó chỉ có lệnh-3 là chạy trong shell hiện hành, nhưng đầu ra của shell thứ cấp lại được ống dẫn đưa vào stdin của lệnh-3.

18.4 Thực hiện xử lý hậu trường

Vì Linux là hệ điều hành multitasking cho nên bạn có thể chạy các lệnh ở hậu trường bằng nhiều cách. Hình thức đơn giản nhất của xử lý hậu trường sẽ giúp bạn chạy một lệnh khác song song với dòng lệnh đang chạy ở mặt trước. Một số phương pháp khác đặt các lệnh càng sâu càng xa hơn nữa trong hậu trường.

18.4.1 Sắp xếp tiến trình chạy trong hậu trường

Shell giúp các bạn khởi hành một tiến trình và, trước khi tiến trình đầu tiên xong việc, tiến trình tiếp theo đã bắt đầu. Khi việc này diễn ra thì tiến trình đầu tiên đã được đẩy vào hậu trường. Bạn đưa một tiến trình nào đó vào hậu trường bằng cách đặt dấu "và" (&) ở cuối dòng lệnh như ở thí dụ sau đây:

```
sort office>office.sorted&
```

Khi nhập lệnh này xong, màn hình hiển thị một con số. Đây là ID của tiến trình (PID) vừa được bạn đưa vào hậu trường. Hệ điều hành dùng PID để nhận dạng tiến trình ấy.

Bình thường khi bạn chạy một lệnh, shell sẽ ngưng hoạt động cho đến khi lệnh hoàn tất. Nếu ghi dấu & sau chuỗi lệnh, shell sẽ bắt đầu hoạt động trở lại ngay sau khi lệnh ở hậu trường được kích hoạt.

Chỉ trừ khi bạn chuyển hướng I/O bằng lệnh hậu trường, còn thì lệnh hậu trường và shell hiện hành sẽ cùng chờ nhập liệu từ bàn phím và sẽ gửi đầu ra ra màn hình, khi đó rất có thể terminal nhận đầu ra liên tục từ lệnh hậu trường và không thể ngắt. Thí dụ:

```
$du/&
```

sẽ liên tục xuất kết quả ra terminal. Bạn không thể ngắt lệnh, nếu không chuyển lệnh trở lại mặt trước. Thật ra bạn vẫn có thể nhập lệnh cho chạy ở mặt trước nhưng dòng đầu ra liên tục của lệnh hậu trường không cho bạn theo dõi được các lệnh một cách tường minh.

Chỉ trừ trường hợp lệnh hậu trường quản lý luôn I/O, còn lại cú pháp thích hợp để chạy công việc hậu trường sẽ như sau:

```
lệnh-chuỗi [tệp_đầu_vào] tệp_đầu_ra &
```

Chẳng hạn bạn muốn chép một loạt những tệp có đuôi .txt vào thư mục thứ cấp mang tên oldstuff và, trong khi máy chưa chép xong hẳn, bạn lại muốn in một danh sách sắp xếp các dữ liệu của tất cả những tệp có tên bắt đầu là office, bạn gõ như sau:

```
cp *.txt oldstuff &
```

```
cat office* |sort| lp
```

Vì tiến trình hậu trường là con của shell hiện hành, nó sẽ bị buộc chấm dứt khi bạn đăng xuất. Tiến trình con luôn tự động chấm dứt khi tiến trình mẹ chấm dứt.

Lưu ý: Khi muốn khởi đầu chương trình này mà không muốn chờ chương trình kia kết thúc, bạn hãy đưa nó vào hậu trường. Ngoài ra bạn cũng có thể chạy ở hậu trường khi phải thực hiện một loạt các chương trình trong đó có ít nhất một chương trình có khả năng tự chạy. Bạn kích hoạt chương trình này và cho nó vào hậu trường.

Bạn cũng có thể sử dụng các terminal ảo của Linux để thi hành một lệnh, sau đó đăng nhập vào terminal khác.

18.4.2 Sử dụng lệnh nohup

Để đặt lệnh sâu hơn nữa vào hậu trường, sâu hơn giới hạn của dấu &, bạn dùng lệnh **nohup** (viết tắt của no hang up). Lệnh **nohup** dùng chuỗi lệnh làm đối số. Tuy nhiên nếu muốn lệnh thực sự chạy ở hậu trường, bạn phải dùng **nohup** với toán tử &. Nếu **nohup** chạy cùng với một lệnh nào ở mặt trước thì lệnh đó sẽ không tự động chấm dứt khi bạn ngắt kết nối terminal hoặc modem. Cú pháp **nohup** như sau:

```
nohup lệnh-chuỗi [tệp_đầu_vào] tệp_đầu_ra &
```

18.4.3 Sử dụng daemon cron

Nếu chạy một lệnh với **nohup**, lệnh sẽ được thi hành tức khắc. Nếu muốn máy chạy lệnh sau này vào thời gian thích hợp hơn, bạn phải gọi daemon mang tên **cron.**

Daemon **cron** là một lệnh được Linux, nói chính xác hơn đó là do chương trình chủ **init**, chạy ở hậu trường. Nhiệm vụ chính của **cron** là cung cấp dịch cụ thời gian cho mọi tiến trình Linux. Bạn có thể bắt **cron** chạy chương trình vào thời điểm nào đó được xác định trước.

*Xem "Lập thời biểu chạy lệnh bằng **cron** và **crontab**"*

18.4.3.1 Lệnh at

Lệnh **at** dùng đối số là thời gian hoặc ngày tháng và nhận chuỗi lệnh từ stdin. Khi phát hiện dấu cuối tệp, **at** tạo ra một shell script Bourne để thi hành lệnh vào thời gian được định sẵn. Lệnh **at** khá uyển chuyển với các loại đối số thời gian và ngày tháng. Thí dụ khi bạn gõ **at** now + 1 day, những lệnh đi sau đó từ stdin vào sẽ được **at** thi hành vào đúng giờ này ngày hôm sau. Bạn có thể sử dụng **at** bên trong một shell script.

Shell script đơn giản chỉ là một tệp chứa mọi lệnh cần thiết để thực hiện một loạt lệnh. Lúc ấy tên của tệp sẽ trở thành một phần của ngôn ngữ lệnh Linux. Sau đây là một thí dụ sử dụng lệnh **at:**

```
at now + 1 day
lệnh-1
lệnh-2
```

Khi được đặt vào một shell script, những dòng trên sẽ giúp bạn thoải mái chạy một vài lệnh vào ngày hôm sau. Muốn chạy bao nhiêu lệnh, bạn chỉ việc gõ chúng vào phía sau dòng lệnh **at.** Shell script này cho phép bạn chạy bao nhiêu lệnh cũng được.

18.4.3.2 Lệnh batch

Về mặt logic, lệnh **batch** tương đương với **at** now. Nếu bạn nhập lệnh **at** now, máy sẽ báo lỗi đại loại như now has passed. Nhưng lệnh **batch** chỉ khác một điều với **at** now ở chỗ **cron** xếp hàng những lệnh chờ thi hành của **at**, **batch** và **cron** thành từng hàng riêng biệt, chứ không lẫn lộn với nhau.

Giả sử bạn nhập những lệnh sau đây vào tệp sao lưu mang tên bkup:

```
tar –cvf lan_anh bkup /usr/home/lan_anh
```

Sau đó bạn bắt máy hãy sao lưu thư mục /usr/home/lan_anh bằng lệnh **batch** bkup.

18.4.3.3 Lệnh crontab

Một trong những điểm tiện lợi nhất của **cron** là nó tự động hoá việc duy trì một hệ thống. Thí dụ bạn có thể bắt **cron** tự động sao lưu hệ thống mỗi ngày vào lúc 4 giờ, từ Thứ hai đến Thứ bảy. Song song, bạn có thể cài đặt, xoá bỏ và liệt kê những lệnh nào muốn chạy tự động như thế bằng lệnh **crontab**.

Muốn chạy những lệnh nào theo thời biểu định sẵn, bạn phải tạo tệp theo dạng **crontab**. Tệp dạng **crontab** bao gồm sáu trường được phân cách bởi dấu trống hoặc tab. Năm trường đầu tiên là những số nguyên chỉ rõ số phút (00-59), giờ (00-23), ngày của tháng (01-31), tháng của năm (01-12) và ngày trong tuần (0-6, với 0 là chủ nhật). Trường thứ sáu là một chuỗi lệnh.

Mỗi trường số có thể chứa một dãy số (chẳng hạn như 1-5 để cho máy biết đó là từ Thứ hai đến Thứ sáu), hoặc những bộ số cụ thể (như 1,20,40 để cho máy biết cứ mỗi 20 phút phải chạy một bộ lệnh nào đó). Trường cũng có thể chứa ký tự sao *, đại diện cho mọi giá trị hợp pháp.

Thí dụ sau đây chạy lệnh **calendar** mỗi 20 phút một lần, bắt đầu từ giữa đêm Thứ hai và kết thúc vào lúc 11:40 chiều Thứ sau:

```
0,20,40***1-5calendar-
```

Nếu đặt tệp này tên là **cron**file, bạn có thể cài đặt nó vào hệ thống **cron** bằng cách gõ lệnh **crontab cron**file.

cron dùng đơn vị mặc định là một phút, có nghĩa là thời lượng ngắn nhất để cho bạn làm việc là một phút. Với tư cách là quản trị viên hệ thống, bạn có thể giới hạn số lượng lệnh mà một user được phép chạy cùng lúc ở một thời điểm nào đó. Bởi vì khi bạn bắt **cron** chạy tệp **at**, **batch**, hoặc **crontab**, điều đó không có nghĩa là tệp sẽ được thi hành đúng vào lúc mà bạn đã xác định.

18.5 Tìm hiểu việc phản hồi lệnh

Đối với những lệnh vì lý do này khác mà không thi hành được, Linux sẽ phản hồi ngay. Hầu hết đều rơi vào các trường hợp gõ sai chính tả hoặc tên tệp không hợp lệ. Khi bạn định chạy một lệnh không có thực, Linux sẽ phản hồi như sau:

```
command: command not found
```

Nếu bạn định sử dụng một tệp không có thực, Linux sẽ báo lỗi:

```
command: file: No such file or directory
```

Ngoài ra nếu bị lỗi do nguyên nhân nào khác, bản thân lệnh sẽ báo cáo, mặc dù không phải lúc nào cũng bằng hình thức dễ hiểu.

Nếu bạn định chạy một lệnh với **nohup** và không chuyển hướng stderr, Linux tự động đặt các thông báo lỗi vào tệp nohup.out. Tệp này nằm ở thư mục nào ra lệnh chạy **nohup**.

Bởi vì những lệnh do **cron** chạy không mang tính khẩn trương, mọi thông báo lỗi (bất cứ đầu ra nào đặt ở stdout và không chuyển hướng) sẽ được gửi đến bạn qua e-mail.

18.6 Chỉnh sửa và đặt bí danh cho các lệnh shell

Nhiều shell có cách đi tắt để thi hành lệnh. Việc chỉnh sửa lệnh (command editing) giúp bạn thay đổi những lệnh đã được gõ vào. Lịch trình lệnh (command history) giúp bạn xem lại lệnh nào đã nhập vào. Thao tác đặt bí danh (aliasing) cho phép bạn tạo một lệnh đại diện cho những lệnh khác. Khi bạn mới gõ vào một phần của câu lệnh, máy sẽ tiên đoán và điền đủ câu lệnh (command completion) bằng cách tự động hiển thị phần còn lại của tên tệp.

18.6.1 Chỉnh sửa lệnh

Chỉnh sửa lệnh có nghĩa là khi gõ xong lệnh nhưng chưa bấm phím <Enter>, bạn có thể thay đổi chỉnh sửa mà không phải gõ lại phần lớn những gì đã gõ. Cách chỉnh sửa tuỳ theo terminal bạn sử dụng. Với terminal thông thường, bạn chỉ cần dịch chuyển con chạy đến chữ cần sửa, bấm <**del**> và điền ký tự đúng vào là xong.

18.6.2 Xem lại lịch trình lệnh

Đặc điểm này giúp bạn xem lại lịch trình những dòng lệnh đã nhập vào và gọi một số dòng lệnh đó lại lần nữa. Như thế bạn đỡ tốn công sức gõ lại cả dòng lệnh. Nếu phối hợp lịch sử lệnh với chỉnh sửa lệnh, bạn dễ dàng chữa lỗi hoặc thay đối số trong các lệnh phức tạp.

Ở cả hai shell, lệnh **history** đều liệt kê danh sách các dòng lệnh mà shell đã lưu lại. Mỗi dòng lệnh đều được đánh số. Thí dụ để thi hành dòng lệnh thứ 10, bạn chỉ cần gõ ! 10. Shell bash sử dụng phím mũi tên trên bàn phím để di chuyển trong danh sách liệt kê.

18.6.3 Đặt bí danh cho lệnh

Bạn có thể gõ bí danh thay vì gõ chính bản thân lệnh đó. Thí dụ lệnh **man** sẽ liệt kê tư liệu về Linux, còn gọi là các trang **man**. Chẳng hạn muốn đặt chữ help thành một bí danh, hoặc là thành một chữ thay thế cho **man**, bạn gõ:

```
alias help=man
```

Lúc này bạn có thể gõ **help cp** hoặc **man cp**, máy sẽ hiển thị các trang **man** về lệnh **cp**.

Bạn có thể sử dụng bí danh cho các lệnh thường đi kèm tuỳ chọn hoặc đối số. Thí dụ nếu muốn liệt kê tên những tệp trong thư mục hiện hành, sắp xếp theo thứ tự thời gian giảm dần (thời gian thay đổi lần gần nhất), bạn gõ:

```
ls -art
```

Lệnh **ls** là để liệt kê tệp. Tuỳ chọn –a chọn tất cả tệp; tuỳ chọn –r xếp tệp theo thứ tự đảo, giảm dần và tuỳ chọn –t xếp theo thời gian thay đổi. Phải nhớ nhiều thế thì gay quá! Bạn có thể gán bí danh timedir cho lệnh vừa kể:

```
alias timedir="ls -art"
```

Bạn phải gõ đủ cặp dấu nháy kép bởi vì shell biết rằng bí danh timedir phải tận cùng bằng dấu trống hoặc <Enter>. Từ nay trở đi khi gõ timedir, bạn sẽ có danh sách cần thiết.

Ghi chú: Đặt bí danh từ dòng lệnh chỉ làm cho bí danh ấy có hiệu lực trong phiên làm việc hiện hành mà thôi. Muốn bí danh luôn hiệu lực mỗi lần bạn đăng nhập, hãy ghi xác định bí danh vào tệp .bash_profile của shell bash, hoặc vào tệp .login của shell C.

18.6.4 Điền đủ câu lệnh

Với đặc điểm này, bạn chỉ cần gõ một phần tên tệp rồi bấm <Tab>, máy sẽ điền đủ cho bạn. Như thế bạn đỡ phải tốn thời gian và tránh gõ sai tên tệp. Nếu nhiều tên tệp có phần mở đầu giống nhau, Linux điền cho đến ký tự chung, sau đó ngừng lại và phát liên tiếp tiếng bip. Sau đó bạn phải tự gõ phần khác nhau giữa tên các tệp.

18.6.5 Bổ sung văn bản bằng cách cắt dán

Cả hai bản phát hành Linux Red Hat và Slackware đều có chương trình tự động kích hoạt ngay lúc máy khởi động, cho phép bạn dùng chuột để chọn văn bản ở bất cứ nơi nào trên màn hình, sau đó dán vào dòng lệnh.

Trong chế độ văn bản (text), muốn hiển thị con chạy, bạn bấm phím chuột. Sau đó bạn chọn bằng cách nhấn và giữ phím trái chuột ở đầu đoạn văn bản, tiếp theo kéo rê cho đến điểm cuối của đoạn văn muốn chép, sau đó di chuyển con chuột đến vị trí cần thiết rồi bấm phím phải chuột để dán chép vào dòng lệnh.

Trong chế độ đồ hoạ, việc dán chép sẽ cần bấm phím giữa của chuột 3 phím hay bấm đồng thời 2 phím của chuột 2 phím để phỏng tạo phím giữa.

18.7 Làm việc với shell script

Shell nhận lệnh, diễn dịch lệnh rồi sau đó sắp xếp sao cho hệ điều hành thi hành lệnh theo ý người sử dụng. Những mục vừa rồi giúp bạn hiểu cách thức shell diễn dịch các ký tự đặc biệt để điền đủ tên tệp, để chuyển hướng các đầu vào/ra, dùng ống dẫn kết nối các tiến trình và đưa chương trình vào thực hiện ở hậu trường. Máy có thể nhận lệnh từ bàn phím terminal hoặc từ một tệp. Shell script là một chuỗi các lệnh shell ở dạng tệp văn bản. Muốn thi hành shell script, bạn gõ tên tệp ấy. Việc này có ba lợi điểm vì bạn:

-Không phải gõ lại cả chuỗi lệnh.

-Chỉ xác định một lần những bước phải thực hiện.

-Đơn giản hoá các thao tác cho mình và cho người khác.

Bạn dùng các biến và từ khoá để viết những chương trình mà shell có thể diễn dịch. Điều này rất có ích bởi vì bạn có thể tạo ra các shell script tổng quát để sử dụng trong nhiều trường hợp khác nhau.

Giả sử mỗi khi đăng nhập bạn có thói quen xem thử bạn bè nào đang có mặt trong hệ thống rồi chạy chương trình **calendar** để hiển thị những cuộc hẹn hôm nay và ngày mai, sau đó lại in ngày giờ hiện hành lên màn hình. Để làm tất cả những việc đó, bạn gõ các lệnh sau:

```
who
calendar
date
```

Nếu đặt tất cả những lệnh này vào một tệp mang tên whatsup và gán cho nó quyền thi hành, bạn sẽ có một shell script. Tệp whatsup phải là tệp văn bản. Bạn dùng trình soạn thảo **vi** hoặc **emacs** để ghi lệnh vào tệp, sau đó ra lệnh sau đây để gán quyền thi hành:

```
chmod +x whatsup
```

Lệnh **chmod** sẽ thay đổi hoặc gán quyền hạn cho tệp. Tuỳ chọn +x gán quyền thi hành cho tệp, nghĩa là làm cho tệp trở thành một lệnh chuẩn của Linux. Ghi lệnh vào tệp và gán quyền thi hành là thao tác chỉ thực hiện một lần. Từ giờ phút này trở đi, bạn chỉ cần gõ whatsup là máy sẽ chạy shell script ấy. Bạn dùng shell script như dùng lệnh. Thí dụ để in kết quả lệnh whatsup, bạn gõ:

```
whatsup | lp
```

Muốn đưa kết quả này vào tệp info, bạn gõ:

```
whatsup > info
```

Để nắm được những gì vừa xem, bạn theo dõi những bước sau đây và thử tạo ra một shell script:

Gọi một trình soạn thảo văn bản, chẳng hạn như **vi** hoặc **emacs** để ghi lệnh shell vào tệp văn bản hoặc ASCII. Ở thí dụ trên, lệnh được ghi vào tệp whatsup.

Gán quyền hạn thi hành cho tệp thí dụ bằng cách gõ:

```
chmod +x whatsup
```

Thử nghiệm lệnh bằng cách gõ tên lệnh và bấm <Enter>.

Bạn thử nghiệm và xem các bước thực hiện shell script bằng lệnh:

```
sh –x tên_script
```

Trong cú pháp này, tên_script là tên của tệp chứa script mà bạn đang quan tâm. Lệnh **sh** –x hiển thị mọi bước mà script phải lần lượt tiến hành. Lệnh này đặc biệt có ích khi bạn muốn gỡ lỗi cho script.

18.7.1 Lập trình có sử dụng shell

Muốn viết ra những chương trình có dùng shell, bạn phải biết về các biến và cấu trúc điều khiển (còn gọi là cấu trúc kiểm tra). Biến là một đối tượng được gán giá trị này hay giá trị khác. Còn cấu trúc điều khiển xác định cách thức mà bạn sẽ điều khiển luồng thi hành một script. Có hai loại cấu trúc cơ bản:

-Cấu trúc quyết định (chẳng hạn như **if .. then .. else** hoặc **case**).

-Cấu trúc vòng lặp (chẳng hạn như các vòng lặp for hoặc **while**).

Cấu trúc quyết định giúp bạn chọn lựa tiến trình của một hành động từ một vài khả năng, thường là tuỳ vào giá trị của một biến hoặc tuỳ vào đầu ra của một lệnh. Cấu trúc vòng lặp giúp bạn lặp lại một phiên lệnh. Mục "Thiết lập môi trường shell" ở trên đã xem xét các biến shell, mục "Lập trình bằng các cấu trúc điều khiển" ở phần sau sẽ cung cấp thông tin về những cấu trúc ấy.

18.7.1.1 Sử dụng echo

Bạn dùng lệnh **echo** để hiển thị thông báo cho biết những gì diễn ra bên trong shell script. Lệnh **echo** hiển thị ra màn hình chính những đối số của bản thân lệnh ấy, nghĩa là những gì tiếp theo phía sau chữ **echo**. Những gì đặt bên trong ngoặc kép sẽ được hiển thị. Bạn có thể chuyển hướng các kết quả của **echo** vào tệp. Lệnh:

```
echo "Please stand by"
```

Sẽ hiển thị trên màn hình terminal như sau:

```
Please stand by
```

Lệnh sau đây đưa cụm Please stand by vào một tệp mang tên messg:

```
echo "Please stand by" >messg
```

Lưu ý: Nhiều người thích lệnh **echo** vì lệnh này giúp họ có cảm giác rằng sau khi gõ lệnh vào, quả là máy có làm việc thật, nhất là trong trường hợp đối với những lệnh cần thời gian mới cho ra kết quả.

echo cũng có ích khi bạn muốn truy vấn một shell script. Việc dùng **echo** ở những thời điểm **then** chốt sẽ cho bạn biết những gì đang xảy ra trong script. Sau đây là tệp whatsup có thêm vào vài lệnh:

```
echo:
echo Let's see who is on the system.
who
echo Any appointment?
calendar
date
echo All done
```

Khi chạy tệp whatsup, màn hình sẽ hiển thị:

```
$whatsup
Let's see who is on the system.
lan_anh tty01 Dec 20 08:51
thu_oanh tty03 Dec 20 08:12
ernie tty07 Dec 20 08:45
Any appointment?
12/20 Office meeting at 1:45
12/21 party after work!
```

```
Mon Dec 20 09:02 EST 1999
All done
$
```

18.7.1.2 Sử dụng chú thích

Có khi soạn xong chương trình shell script, bạn chưa dùng ngay hoặc dùng xong rồi cất đi một thời gian. Sau này bạn quên mất script này dùng để làm gì và làm như thế nào. Vì thế bạn nên ghi chú thích vào script.

Dấu thăng (#) báo cho shell biết điểm bắt đầu của phần chú thích. Những ký tự kể từ dấu thăng cho đến cuối dòng đều thuộc về phần chú thích. Sau đây là thí dụ chú thích cho whatsup:

```
# Ten tep: whatsup
# Muc dich: xem ai dang dang nhap va ngay thang cac cuoc hen
echo Let's see who is on the system.
who # Xem ai dang dang nhap
echo Any appointment
calendar # Kiem tra cac cuoc hen
date # Hien thi ngay thang
echo All done
```

Bạn thử chạy script này lần nữa và thấy kết quả trên màn hình vẫn không thay đổi, bởi vì các chú thích không tham gia vào hoạt động của shell script.

18.7.1.3 Sử dụng biến trong shell

Muốn sử dụng biến, bạn phải biết cách gán giá trị cho biến và làm cách nào để truy cập giá trị được chứa trong biến. Có bốn cách để gán giá trị cho biến:

1. Gán trực tiếp.
2. Dùng lệnh read.
3. Sử dụng các tham số dòng lệnh.
4. Thay thế đầu ra của một lệnh.

18.7.1.3.1 Gán trực tiếp

Cách trực tiếp nhất để gán một giá trị cho biến là viết một biểu thức như sau:

```
myemail=edsgar@crty.com
```

Biểu thức này gán giá trị edsgar@crty.com cho biến myemail. Bạn không được chừa dấu trống trước và sau dấu bằng. Phương pháp gán giá trị trực tiếp cho biến có dạng thức chung như sau:

```
tên_của_biến=giá trị_của_biến
```

Nếu giá trị_của_biến có chứa dấu trống, bạn phải đặt giá trị ấy bên trong nháy kép. Thí dụ muốn gán địa chỉ Phòng 21, Nhà C cho biến myoffice, bạn gõ lệnh:

```
myoffice="Phòng 21, Nhà C"
```

Shell sẽ lấy giá trị của biến bất kỳ khi nào trông thấy tên của biến di theo sau dấu dollar ($). Bạn sẽ thấy việc này diễn ra khi hai lệnh sau đây được thi hành:

```
echo "My e-mail address is $myemail"
echo "My office is $myoffice"
```

Giả sử bạn thường xuyên chép tệp vào thư mục có tên /corporate/info/publics/office. Muốn chép một tệp mang tên current vào thư mục ấy, bạn gõ lệnh:

```
cp current /corporate/info/publics/office
```

Để cho dễ dàng hơn, bạn gán cái tên đường dẫn khá dài ấy cho biến corpoffice bằng biểu thức:

```
corpoffice=/corporate/info/publics/office
```

Bây giờ khi muốn chép tệp hiện hành vào thư mục ấy, bạn gõ:

```
cp current $corpoffice
```

Shell thay thế $corpoffice bằng giá trị của biến corpoffice, sau đó thi hành lệnh **cp.**

18.7.1.3.2 Dùng lệnh read

Lệnh read lấy dòng nhập liệu thứ hai gán cho biến. Shell script sau đây mở rộng thí dụ corpoffice vừa rồi để yêu cầu một user khẳng định tên tệp sắp được chép:

```
# Tên: copycorp
# Mục đích: chép tệp được xác định vào
# /corporate/info/publics/office
corpoffice=/corporate/info/publics/office
echo Nhập tên tệp cần chép # Nhắc người sử dụng
read filename                 # Ghi nhận tên tệp
cp $filename $corpoffice            # Tiến hành sao chép
```

Ghi chú: Bạn cần đặt biến filename trong nháy kép để đảm bảo script này vẫn chạy khi trong filename có chứa dấu trống.

Lệnh **read** tạm dừng script và chờ nhập liệu từ bàn phím. Khi có tín hiệu <Enter>, script sẽ tiếp tục làm việc. Nếu trong khi lệnh **read** đang chờ nhập liệu mà bạn bấm <Ctrl-d> (đôi khi được viết là ^D), script sẽ kết thúc công việc.

18.7.1.3.3 Sử dụng các tham số dòng lệnh

Khi diễn dịch lệnh, shell gắn tên biến vào từng thành tố của dòng lệnh. Các thành tố này gồm những chuỗi ký tự được phân cách bởi ký tự trống SP hoặc tab (nếu chuỗi ký tự được phân cách bởi nhiều dấu SP nhưng tất cả nằm trong một cặp nháy kép thì nó vẫn chỉ được coi là một thành tố mà thôi).

Biến được gắn với các thành tố ở dòng lệnh là $0, $1, $2 cho đến $n. Những biến này tương ứng với vị trí của các thành tố trong dòng lệnh: tên lệnh là $0, đối số hoặc tham số đầu tiên là $1, v.v. Để chứng minh điều này, mời bạn chạy shell script mang tên shovars sau đây:

```
# Tên: shovars
```

```
# Mục đích: chứng minh các biến dòng lệnh
echo $0
echo $2 $4!
echo $3
```

Đến đây giả sử bạn gõ lệnh:

```
shovars –s hello "look at me" bart
```

Đầu ra của shell script sẽ có dạng như sau:

```
/home/sss/shovars
hello bart!
look at me
```

Dòng đầu tiên là tên lệnh (biến $0), dòng thứ hai là các đối số thứ hai và thứ tư (các biến $2 và $4) và dòng cuối cùng là đối số thứ ba (biến $3).

Ở thí dụ sau đây, shell script sẽ xóa bỏ một tệp nhưng trước đó phải chép tệp này vào thư mục /tmp để phòng trường hợp cần xem lại sau này:

```
# Tên: safrm
# Mục đích: chép tệp vào thư mục /tmp sau đó xóa tệp
# khỏi thư mục hiện hành
# đầu tiên chép $1 vào /tmp
cp $1 /tmp
# bây giờ xóa tệp
rm $1
```

Đến đây nếu bạn gõ safrm abc def, chỉ mỗi tệp abc mới bị xóa khỏi thư mục hiện hành bởi vì shell script safrm được lệnh xóa mỗi biến $1. Tuy nhiên bạn có thể dùng $* để đại diện cho mọi tham số dòng lệnh. Bạn mở rộng phạm vi hoạt động của safrm bằng cách thay thế mỗi lần xuất hiện của $1 bằng $*. Sau đó nếu bạn gõ safrm abc def xx guio, tất cả bốn tệp vừa kể sẽ bị xóa khỏi thư mục hiện hành.

18.7.1.3.4 Thay thế đầu ra của một lệnh

Bạn có thể lấy kết quả của một lệnh vừa được thi hành xong mà gán cho biến. Ở thí dụ sau đây, bạn gán tên của thư mục hiện hành cho biến cwd:

```
cwd=`pwd`
```

Bạn ghi nhận rằng lệnh in thư mục đang hoạt động (pwd) được đặt trong dấu nháy ngược thay vì dấu ngoặc kép.

Shell script sau đây đổi tên của tệp bằng cách gán ngày tháng năm cho tên tệp:

```
# Tên: stamp
# Mục đích: đặt lại tên tệp và gán ngày tháng năm hôm nay vào tên đó
# gán cho td giá trị ngày tháng năm hôm nay theo dạng mmddyy :
td=`date+%m%d%y`
# đặt lại tên tệp :
mv $1 $1 .$td
```

Ở thí dụ trên, bạn gán ngày tháng năm hôm nay cho biến td. Ở dòng cuối, thông tin này được gán cho biến $1. Nếu hôm nay là ngày 24 tháng 2 năm 1999 và nếu bạn chạy script này cho tệp tên myfile, tệp này sẽ được đổi tên (di chuyển) thành myfile.022499.

18.7.1.4 Sử dụng những ký tự đặc biệt với shell

Bạn đã biết shell có cách xử lý riêng với những ký tự đặc biệt như >, *, ?, $, v.v. Trường hợp bạn muốn chúng được shell xử lý như ký tự bình thường thì phải làm sao?

A. Có thể dùng dấu nháy đơn ' (single quote) để bắt shell "lờ" tất cả các ký tự đặc biệt ấy đi. Thí dụ bạn gộp một chuỗi ký tự trong cặp nháy đơn như sau:

```
grep '^Huy Hùng' lái_xe
```

Kết quả của lệnh **grep** là những dòng nào ở tệp lái_xe mà bắt đầu bằng Huy Hùng sẽ được hiển thị. Dấu mũ (^) bắt **grep** tìm từ đầu dòng. Nếu không nằm trong cặp dấu nháy, chuỗi ký tự Huy Hùng sẽ bị hiểu nhầm (một vài hệ thống coi đây là ống dẫn). Bởi vì cùng nằm trong cặp dấu nháy, cả dấu trống giữa Huy và Hùng cũng sẽ không bị hiểu sai.

B. Có thể dùng dấu nháy kép " (double quote) để bắt shell "lờ" đi một số ký tự đặc biệt, trừ dấu sổ ngược và dấu dollar. Ở thí dụ dưới đây, các ký tự như dấu sao, dấu trống, dấu lớn hơn, được xử lý như là ký tự bình thường vì nằm giữa cặp nháy kép:

```
echo "**Hãy gõ tên bạn vào đây ->"
```

Nhờ dấu nháy kép mà dollar ($) vẫn được hiểu như ký tự đặc biệt; ở thí dụ sau $LOGNAME sẽ cho giá trị của biến LOGNAME:

```
echo ">>>$LOGNAME trả lệ phí $5"
```

Nhưng viết $5 thì sẽ bị hiểu sai, vậy phải dùng dấu sổ ngược \ (backslash) để bắt shell "lờ" đi ký tự $. Thí dụ muốn shell "lờ" dấu dollar trước số 5 bạn gõ lệnh:

```
echo ">>> Trả lệ phí \$5, $LOGNAME"
```

Và kết quả có thể sẽ là:

```
>>> Lan Anh trả lệ phí $5
```

18.7.2 Lập trình với cấu trúc điều khiển

Shell cho phép sử dụng hai cấu trúc điều khiển kiểu cấu trúc quyết định và cấu trúc vòng lặp. Với những cấu trúc quyết định như **if ... then ... else** và **case ... in**, bạn có thể bắt shell quyết định thi hành một mệnh đề (statement) căn cứ trên giá trị của một biểu thức (chẳng hạn như là biến, thuộc tính tệp, số lượng tham số trong một script, hoặc là kết quả thi hành một lệnh). Với những cấu trúc vòng lặp như **for** và **while**, bạn có thể thi hành một loạt các lệnh đối với một nhóm tệp nào đó hoặc chừng nào một điều kiện vẫn còn giá trị. Những mục tiếp theo đây sẽ minh hoạ các kỹ thuật chủ yếu của việc lập trình với cấu trúc điều khiển.

18.7.2.1. Sử dụng cấu trúc case

Cấu trúc **case ... in** là một cấu trúc quyết định cho phép bạn chọn một hoặc nhiều hành động căn cứ trên giá trị của biến. Danh sách 18.2 là shell script thể hiện một **menu.**

*Danh sách 18.2: Lập trình một **menu** bằng **case***

```
# Name: ShrtMenu
# Mục đích: Giúp user in, xoá một tệp, hoặc
# thoát khỏi chương trình
# Hiển thị menu :
echo Xin chon I, hoac X, hoac T de:
echo [I]n mot tep
echo [X]oa mot tep
echo [T]hoat
# Nhận phản hồi từ người sử dụng :
read response
# Dùng case để so sánh với phản hồi :
case $response in
I|i) echo Ten tep can in?
read filename
lp $filename;;
X|x) echo Ten tep can xoa?
read filename
rm $filename;;
*) echo Dang thoat ra;;
esac
```

Cú pháp của **case** là :

```
case word in
type ) statement(s) ;;
type ) statement(s) ;;
...
esac
```

Tham số word (chữ) được so sánh với từng tham số type (kiểu), bắt đầu bằng type đầu tiên tiếp theo **in**. Khi nào word so sánh thấy đúng với một type, thì một hoặc nhiều mệnh đề (statement) tiếp theo sẽ được thi hành. Những mệnh đề này được kết thúc bằng hai dấu chấm phẩy. Mỗi cấu trúc **case** kết thúc bằng một chữ **esac** (là chữ **case** viết ngược).

Trong danh sách 18.2 có sử dụng ống dẫn (ký tự vạch đứng) để cho phép chọn lựa khi so sánh. Thí dụ I|i nghĩa là gặp chữ I hoa hoặc i thường đều coi là đúng. Ký tự wildcard * đại diện cho mọi kiểu chữ khác mà không cần phải kể ra. Nếu bấm bất kỳ phím nào ngoài <I>, <i>, <X>, <x>, người sử dụng sẽ thoát khỏi **menu.**

Danh sách 18.3 dùng **case** để thực hiện một chọn lựa căn cứ vào giá trị của tham số $#.

*Danh sách 18.3: Phân tích dòng lệnh với **case***

```
# Tên: recent
```

```
# Mục đích: liệt kê những tệp mới nhất trong thư mục.
# Nếu user gõ recent <Enter> máy sẽ hiển thị tên của 10 tệp
# vừa được thay đổi gần đây nhất.
# Nếu user gõ recent n <Enter> máy sẽ hiển thị tên của n tệp
# vừa được thay đổi gần đây nhất.
# Trong các trường hợp còn lại, máy sẽ báo lỗi
#
# Case sẽ căn cứ vào giá trị của tham số
case $# in
ls –lt | head ;;
# ls –lt liệt kê tên các tệp theo thứ tự
# thay đổi gần đây nhất,
# head hiển thị 10 dòng đầu tiên của tệp
1) case $1 in
[0-9]*) ls –lt | head $1 ;;
*) echo Usage: recent number-of-files;;
esac;;
*) echo Usage: recent number-of-files;;
esac
```

18.7.2.2 Tìm trạng thái thoát khỏi

Khi một lệnh shell được thi hành thì trạng thái cuối cùng của nó sẽ là hoặc có kết quả, hoặc không có kết quả.

Giả sử bạn dùng lệnh **grep** French Terms customers để tìm xem chuỗi ký tự French Terms có nằm trong tệp customers hay không. Nếu tệp customers có thật và bạn có quyền hạn đối với nó, khi tìm được French Terms trong tệp thì coi như lệnh shell đã thực hiện có kết quả. Còn nếu một trong những điều kiện vừa kể không đúng thì coi như lệnh không có kết quả.

Shell luôn luôn phản hồi kết quả của một lệnh, một chương trình, hoặc một shell script. Giá trị được phản hồi gọi là "trạng thái thoát khỏi" (exit status) của một lệnh và được đại diện bởi biến $?. Thực hiện thí dụ sau, bạn sẽ thấy giá trị của biến này:

```
grep "French Terms" customers
echo $?
```

Ghi chú: Nếu $? có giá trị bằng 0 thì lệnh có kết quả, nếu giá trị $? khác đi là không có kết quả.

Ở thí dụ sau đây, trạng thái thoát khỏi của lệnh **who** | **grep** $1 sẽ được dùng trong một cấu trúc **case**:

```
# Tên: just.checking
# Mục đích: Xem qua ai đang ở trong hệ thống
# Cú pháp: just.checking login_name
```

```
#
case 'who | grep $1>/dev/null'in
0) echo "$1 co mat tren mang.";;
*) echo "$1 khong co tren mang. Hen lan sau.";;
esac
echo "Chuc mot ngay vui ve!"
```

Nếu bạn gõ just.checking lan_anh và nếu lan_anh có mặt trên mạng, máy sẽ hiển thị:

```
lan_anh co mat tren mang.
Chuc mot ngay vui ve!
```

Còn không, bạn sẽ thấy:

```
lan_anh khong co mat tren mang. Hen lan sau.
Chuc mot ngay vui ve!
```

18.7.2.3 Sử dụng cấu trúc if

Cấu trúc **if** ... **then** ... **else** là loại cấu trúc quyết định, cho phép bạn chọn một trong hai hành động căn cứ trên kết quả của một lệnh. Phần **else** của cấu trúc này không bắt buộc có biểu thức và khoảng ba chấm lửng là chỗ dành cho một hoặc nhiều lệnh. Nếu trạng thái thoát khỏi của lệnh cuối cùng theo sau **if** khác với 0 (nghĩa là nếu lệnh được thực hiện không có kết quả), thì các lệnh theo sau **then** và đi trước **else** (nếu có **else**) sẽ được thi hành.

Nếu lệnh cuối cùng theo sau **if** có kết quả, các lệnh theo sau **then** sẽ được thi hành và tiếp theo đó, đến lượt các lệnh theo sau **if** (điểm kết thúc của cấu trúc) sẽ được thi hành. Nếu những lệnh cuối cùng không kết quả, lệnh đi theo sau **else** sẽ được thi hành.

Sau đây bạn sử dụng lại thí dụ trước với cấu trúc **if** ... **then** ... **else**

```
# Tên: just.checking
# Mục đích: Xem qua ai đang ở trong hệ thống
# Dùng: just.checking login_name
#
if
who | grep $1>/dev/null
then
echo "$1 dang tren mang."
else
echo "$1 khong co tren mang. Hen lan sau."
fi
echo "Chuc mot ngay vui ve!"
```

18.7.2.4 Sử dụng lệnh test

Lệnh **test** dùng để kiểm tra xem các user có quyền hạn chép hay di dời tệp hay không, hoặc xem họ xử lý một tệp bình thường hay là một thư mục. Ngoài ra **test** còn thực

hiện một số thao tác khác, chẳng hạn **test** –f abc sẽ có kết quả nếu abc có thật và là một tệp bình thường.

Bạn có thể đảo ngược ý nghĩa của **test** khi dùng dấu chấm than trước tuỳ chọn. Thí dụ để kiểm tra xem có phải mình không có quyền hạn đối với tệp abc, bạn gõ **test** ! –r abc. Bảng 18.7 liệt kê các tuỳ chọn cho lệnh **test.**

*Bảng 18.7: Các tuỳ chọn dùng với lệnh **test***

Tuỳ chọn	**Ý nghĩa**
-f	Có kết quả nếu tệp có thật và là tệp bình thường.
-d	Có kết quả nếu tệp là thư mục.
-r	Có kết quả nếu tệp có thật và là tệp đọc được.
-s	Có kết quả nếu tệp có thật và là tệp không rỗng.
-w	Có kết quả nếu tệp có thật và là tệp có quyền ghi.
-x	Có kết quả nếu tệp có thật và là tệp khả thi.

*Danh sách 18.4: Một thí dụ dùng lệnh **test***

```
# Tên: safcopy
# Mục đích: chép file1 sang file2
# Kiểm tra xem bạn có quyền hạn đọc file1
# Nếu file2 có thật thì
# nếu file2 là một tệp bạn có thể ghi
# thì báo cho user biết và lấy quyền thi hành
# còn không sẽ thoát khỏi $# còn không
# chép tệp
#
# Kiểm tra số hợp lệ của đối số
case$# in
2) if test ! –r $1 # không thể đọc tệp đầu tiên;;
then
exit (1) thoát với tình trạng thoát khác với 0;;
if;
if test –f $2 # tệp thứ hai có thật hay không?;;
then
if test –w $2 # bạn có thể ghi vào tệp không?;;
then
echo "$2 có that, chep chong len? (Y/N)";;
read resp # nhận phản hồi từ user;;
case $resp in
Y|y) cp $1 $2;; # tiến tới;;
*) exit (1);; #goodbye!;;
```

```
esac
else
exit (1) # tệp thứ 2 có thạt nhưng không thể ghi
fi
else # tệp thứ hai không có thật; cứ việc chép!;
cp $1 $2;;
fi
*) echo "Usage: safcopy source destination"
exit (1);;
esac
```

Bạn cũng có thể dùng **test** để kiểm tra số. Thí dụ để xác định xem giá trị của biến hour có lớn hơn 12 không, bạn gõ **test** $hour –gt 12. Bảng 18.8 liệt kê vài tuỳ chọn bạn có thể dùng với **test** để so sánh những con số.

*Bảng 18.8: Tuỳ chọn dùng với **test** để so sánh các con số*

Tuỳ chọn	Ý nghĩa
-eq	Bằng nhau
-ne	Không bằng nhau
-ge	Lớn hơn hoặc bằng nhau
-gt	Lớn hơn
-le	Nhỏ hơn hoặc bằng nhau
-lt	Nhỏ hơn

*Danh sách 18.5: Hiển thị lời chào bằng lệnh **test***

```
# Tên: greeting
# Mục đích: Hiển thị Good Morning nếu hour nhỏ hơn 12
# Good Afternoon nếu hour nhỏ hơn 5PM
# Good Evening nếu hour lớn hơn 4PM
# Lấy hour
hour='date+%H'
# Kiểm tra giờ của ngày
if test $hour –lt 12
then
echo "Good Morning, $LOGNAME"
else
if test $hour –lt 17
then
echo "Good Afternoon, $LOGNAME"
else
echo "Good Evening, $LOGNAME"
```

```
fi
fi
```

18.7.2.5 Sử dụng các cấu trúc vòng lặp

Các cấu trúc vòng lặp giúp bạn viết shell script có chứa vòng lặp (loop). Hai loại vòng lặp cơ bản là **for** và **while.**

Với vòng lặp dạng **for**, bạn xác định một nhóm tệp hoặc giá trị để dùng với một số lệnh. Thí dụ để chép mọi tệp .txt vào thư mục textdir, bạn sử dụng vòng **for**:

```
for i in *.txt
do
cp $i textdir/$i
done
```

Shell diễn dịch biểu thức **for** i in *.txt và cho phép biến i lấy tên của những tệp trong thư mục hiện hành có đuôi là .txt. Bạn có thể sử dụng biến $i với bất kỳ biểu thức nào giữa hai từ khoá **do** và **done.**

Script ở danh sách 18.6 in một nhóm tệp, mỗi tệp có trang tit riêng. Script này cũng gửi thông báo đến user cho biết tình hình yêu cầu in ấn. Ký tự $* đại diện mọi tham số gửi đến cho lệnh shell.

*Danh sách 18.6: Xử lý tệp bằng lệnh **for***

```
# Tên: Prntel
# Mục đích: In một hoặc nhiều tệp
# mỗi tệp có trang tít riêng
# Lưu ý user những tệp nào đã gửi đến máy in,
# và những tệp nào chưa gửi.
# Thực hiện công việc này cho mọi tham số của lệnh
for i in $*
do
if lp -t $i -dlasers $i>/dev/null
then
echo $i>>printed
else
echo $i>>notprinted
fi
done
# cuối vòng lặp
if test -s printed
then
echo "Những tệp này đã gửi đến máy in">mes
cat printed>>mes
mail $LOGNAME<mes
```

```
rm mes printed
fi
if test –s notprinted
then
echo "Những tệp này chưa gửi đến máy in">mes
cat notprinted>>mes
mail $LOGNAME<mes
rm mes notprinted
fi
```

Vòng lặp **while** xem xét trạng thái thoát khỏi của một lệnh giống như cách làm của **if.** Script ở danh sách 18.7 lưu ý các user rằng họ có thư mới đến. Script giả định khi nội dung một hộp thư thay đổi thì có nghĩa là có thư mới đến. Script dùng lệnh diff để so sánh hai tệp, sau đó báo cáo những điểm khác nhau. Nếu hai tệp không khác nhau, trạng thái thoát khỏi là zero (lệnh có kết quả).

*Danh sách 18.7: Lặp lại các lệnh bằng **while***

```
# Tên: checkmail
# Mục đích: Thông báo user biết hộp thư có thay đổi
# Đề nghị: Chạy chương trình này ở hậu trường
# lấy kích thước hộp thư để so sánh
cp $MAIL omail
# MAIL là một biến "đặc biệt" cho biết tình trạng hộp thư của user
# trong khi omail và $MAIL giống nhau, tiếp tục vòng lặp
while diff omail $MAIL>/dev/null do
cp $MAIL omail
sleep 30 # sleep, tạm ngưng 30 giây
done
# Có lẽ các tệp đã thay đổi
echo "Có thư đến!!!" | write $LOGNAME
```

Bạn có thể thấy rằng một vài kỹ thuật được dùng với **if** ... **then** ... **else** cũng có thể chuyển sang dùng trong vòng lặp **while**. Điểm khác nhau là, với vòng **while**, bạn đang thao tác một tiến trình lặp đi lặp lại.

18.8. Căn chỉnh Shell Linux

Shell bắt đầu làm việc ngay khi bạn vừa đăng nhập xong. Qua hai bảng 18.2 và 18.3 bạn thấy rằng shell thiết lập giá trị cho các biến đặc biệt để xác định môi trường. Bạn có thể thay đổi những thiết lập ấy và gán giá trị khác cho biến bằng cách chỉnh sửa tệp .bash_profile, nếu bạn đang dùng shell bash. Nếu đang chạy shell C, bạn lập biến bằng cách chỉnh sửa tệp .login. Bạn cũng có thể đặt bí danh cho lệnh.

Shell bắt đầu làm việc mỗi khi bạn nhập lệnh. Lệnh thừa hưởng nhiều đặc điểm hoặc môi trường của shell hiện hành. Bạn lưu ý hai điểm sau đây về shell:

Shell mới sẽ chạy trong thư mục hiện hành của bạn. Lệnh pwd phản hồi cùng một giá trị bên trong shell, giống như trước khi shell khởi hành.

Shell mới sẽ nhận nhiều biến từ shell hiện hành. Có nhiều cách để kiểm tra xem các biến trong shell hiện hành có được xuất sang shell mới hay không.

18.8.1 Xuất các biến sang shell mới

Khi bạn tạo ra các biến shell hoặc gán giá trị cho các biến đang sẵn có, chúng hiện diện trong shell hiện hành. Một biến được lập sẵn trong shell đăng nhập sẽ khả dụng cho mọi đối số dòng lệnh. Biến được lập trong shell chỉ mang giá trị ấy trong chính shell đó mà thôi. Khi bạn thoát khỏi shell đó, giá trị biến mất hoặc được thiết lập lại.

Thí dụ bạn gõ hai lệnh sau từ dòng lệnh:

```
today=Thursday
echo $today
```

Giả sử lệnh **echo** hiển thị Thursday. Lại Giả sử rằng bạn chạy shell script mang tên whatday sau đây:

```
# Tên: whatday
# hiển thị giá trị hiện hành của biến today
echo "Hôm nay là $today."
# lập giá trị của today
today=Friday
# hiển thị giá trị hiện hành của biến today
echo "Hôm nay là $today."
```

Tiếp theo bạn nhập bốn lệnh như sau từ dòng lệnh:

```
chmod +x whatday
today=Thursday
whatday
echo $today
```

Bạn sẽ thấy trên màn hình:

```
Hôm nay là .
Hôm nay là Friday.
Thursday
```

Giá trị của biến today trong shell đăng nhập là Thursday. Khi chạy shell script whatday, bạn thấy thoạt tiên biến today chưa được định nghĩa (hiển thị "Hôm nay là .). sau đó biến today được gán giá trị Friday trong shell. Khi script whatday chạy xong, bạn trở về shell đăng nhập và today vẫn giữ y giá trị nguyên thuỷ, là Thursday.

Khi whatday khởi hành, nếu bạn muốn cho biến today có cùng giá trị như lúc ở shell đăng nhập, bạn hãy dùng lệnh export. Lệnh này sẽ xuất, nghĩa là chuyển các biến từ shell này sang shell khác:

```
 export today
```

Đến đây, bất kỳ shell nào khởi hành từ shell đăng nhập sẽ thừa hưởng giá trị của biến today. Bạn hãy viết thêm lệnh export vào chuỗi các lệnh phía trên:

```
today=Thursday
export today
whatday
echo $today
```

Bây giờ đầu ra sẽ là:

```
Hôm nay là Thursday.
Hôm nay là Friday.
Thursday
```

Bạn lưu ý giá trị mà biến nhận được trong shell do whatday khởi động đã không được đưa ngược về shell đăng nhập. Việc xuất khẩu hoặc thừa hưởng giá trị của biến chỉ đi theo một hướng – từ shell hiện hành xuống shell mới, không bao giờ chạy ngược lên trên. Do đó mỗi khi chuyển (thay đổi) thư mục từ bên trong shell, bạn sẽ trở về điểm khởi hành sau khi shell hoàn tất nhiệm vụ.

Bạn có thể xuất khẩu bất kỳ biến nào từ shell này xuống shell khác bằng cú pháp:

```
export tên_biến
```

Trong cú pháp này, tên_biến là tên của biến mà bạn muốn xuất. Thí dụ muốn thay đổi loại terminal từ cấu hình hiện hành sang vt100 chẳng hạn, bạn gõ các lệnh sau để cho các shell hoặc chương trình dưới có thể truy cập giá trị mới của TERM:

```
TERM=linux
export TERM
```

Khi bạn thiết lập hoặc thay đổi các biến shell bash trong tệp .bash_profile, hãy nhớ xuất khẩu chúng. Thí dụ bạn muốn biến PATH la PATH=/bin:/usr/bin:/usr/local/bin:, hãy ghi như thế vào tệp .bash_profile và tiếp sau đó là lệnh export:

```
export PATH
```

Muốn thay đổi dấu nhắc shell, bạn phải lập giá trị cho PS1 trong tệp .bash_profile. Muốn thay đổi dấu nhắc từ $ sang Ready $ chẳng hạn, bạn dùng trình soạn thảo để thêm những dòng sau vào tệp .bash_profile:

```
PS1="Ready $"
export PS1
```

Ghi chú: Muốn cho những thay đổi của những tệp .bash_profile hoặc .login có hiệu lực, bạn phải đăng xuất rồi đăng nhập trở lại.

18.8.2 Xác định các bí danh lệnh

Bí danh lệnh rất tiện dụng khi bạn muốn định nghĩa những lệnh bạn dùng thường xuyên nhưng lại không muốn nhớ từng chi tiết. Bí danh lệnh cũng tăng cường môi trường làm việc của bạn bằng nhiều công cụ có ích. Ở thí dụ dưới đây, lệnh sẽ gán bí danh recent cho một lệnh khác có nhiệm vụ liệt kê 10 tệp được thay đổi gần đây nhất trong thư mục hiện hành:

```
alias recent="ls -lat |head"
```

Để khỏi phải gõ các bí danh lệnh mỗi khi đăng nhập, bạn nhớ ghi chúng vào tệp .login hoặc .bash_profile. Từ thời điểm đó về sau, các bí danh lệnh sẽ sẵn sàng khi bạn đã vào trong shell của mình.

18.9 So sánh giữa các loại shell

Dưới đây là một bảng so sánh giữa các shell qua đánh giá nhiều chỉ tiêu khác nhau.

```
+++ rất tốt
++ tốt
+ có
- yếu
- - không có
```

Bảng 18.9 : So sánh giữa các loại shell

Thứ tự	Chỉ tiêu	sh	ksh	bash	zsh	csh	tcsh
1	Cấu hình	-	+	++	+++	+	++
2	Thi hành lệnh	+	+	+	++	+	++
3	Điền ký tự	--	+	++	+++	+	++
4	Soạn trên dòng	-	+	++	++	-	++
5	Thay thế tên	+	+	++	++	+	++
6	Lịch trình	--	+	++	++	+	++
7	Chỉnh hướng và ống dẫn	+	+	+	++	+	+
8	Sửa chính tả	--	--	--	+	--	+
9	Lập dấu nhắc	+	+	+	++	+	++
10	Điều khiển việc	--	+	+	+	+	+
11	Điều khiển thi hành	+	+	+	+	+	+
12	Quản lý báo hiệu	+	+	+	+	-	-

Chương 19. Quản lý đa tiến trình

Xin nhắc lại, Linux là một hệ điều hành multiuser (đa user, nhiều người sử dụng) và multitasking (đa nhiệm). Giống như Windows NT/2000, Linux có thể bắt đầu chạy một chương trình này, rồi chạy ngay một chương trình khác trước khi chương trình đầu tiên chấm dứt.

Hầu hết các máy vi tính đều chỉ có một CPU và một bộ nhớ chính (RAM) nhưng có thể có nhiều ổ đĩa hoặc ổ băng từ để làm bộ nhớ phụ, cùng với nhiều thiết bị xuất nhập dữ liệu khác. Tất cả những tài nguyên này được quản lý và chia sẻ cho nhiều user bởi Linux, làm cho mỗi user tưởng rằng máy được cấu hình theo ý riêng của mình.

Các chủ đề chính sẽ được đề cập đến trong chương này là:

-Tìm hiểu chế độ đa nhiệm

-Khởi hoạt đa tiến trình

-Sử dụng các lệnh hẹn giờ

-Theo dõi và báo cáo môi trường đa nhiệm

-Điều khiển đa tiến trình

19.1 Tìm hiểu chế độ đa nhiệm

Như vừa nói ở trên, Linux có nhiệm vụ tạo cho bạn cảm giác rằng mỗi khi vừa gõ câu lệnh là cả hệ thống sẽ phục tùng một mình bạn. Linux quả thật có thể xử lý hàng trăm ngàn lệnh như thế trong khoảnh khắc từ khi bạn bấm <Enter> đến lúc bạn thấy phản hồi đầu tiên.

Xin hãy tưởng tượng Linux đang theo dõi hàng tá công việc cùng diễn ra một lúc. Nó phải chia sẻ các khả năng xử lý, lưu trữ, xuất nhập dữ liệu cho nhiều user hoặc nhiều tiến trình do chỉ một user yêu cầu. Linux theo dõi một danh sách các công việc đang chờ xử lý, cho nên còn gọi là hàng đợi (queue). Những công việc này rất đa dạng, bao gồm công việc của user, của hệ điều hành, của dịch vụ e-mail và các công việc hậu trường khác như in ấn. Mỗi việc như thế được Linux cấp cho một "khoảnh" thời gian, mà nếu so với tốc độ của chúng ta thì khoảnh khắc ấy vô cùng ngắn - chỉ là một phần nhỏ của giây. Đối với máy tính, khoảnh khắc ấy đủ để thực hiện hàng trăm ngàn phép tính. Thời lượng của khoảnh khắc được cấp phát có thể tuỳ thuộc vào thứ tự "ưu tiên tác vụ".

Linux phải xử lý một công việc trong chốc lát, rồi lại tạm gác nó sang bên để bắt đầu (các) nhiệm vụ khác, sau đó trở lại với công việc ban đầu, v.v. Linux tiếp tục quay vòng như thế cho đến khi xong hẳn một công việc nào đó và xoá nó ra khỏi hàng đợi của "những việc cần làm ngay". Thao tác đó, đôi khi gọi là phân thời hay chia sẻ thời gian (time-sharing), giúp phân chia các tài nguyên khác của hệ thống cho những công việc đang xếp hàng chờ xử lý. Thao tác phân thời phải được tiến hành theo một phương thức ổn định và hiệu quả.

Như bạn đã biết khi đọc các chương trước, trong hệ điều hành đa nhiệm thì mỗi nhiệm vụ hoặc công việc được gọi là một "tiến trình". Bảng 19.1 trình bày các loại tiến trình khác nhau của Linux.

Bảng 19.1: Các loại tiến trình của Linux

Loại tiến trình	Mô tả
Interative	« Tương tác », do shell kích hoạt và chạy ở mặt trước hoặc hậu trường
Batch	« Bó », một loạt các tiến trình được hẹn giờ
Daemon	« Ma », được kích hoạt vào lúc khởi động máy để ngầm thực hiện các chức năng hệ điều hành khi có yêu cầu, chẳng hạn như LPD, NFS và DNS

Các chương trước đã giới thiệu những gì bạn có thể cho chạy ở hậu trường. Khi chương trình đã chạy ở hậu trường, bạn lại tiếp tục nhận lệnh mới, hoặc thực hiện một công việc khác. Đây là một đặc điểm của multitasking. Linux dùng phương pháp chia sẻ thời gian để "cân đong đo đếm" những lệnh mà bạn vừa ra với những lệnh đang chạy ở hậu trường. Chương này sẽ giới thiệu thêm với bạn nhiều cách khác nhau để hẹn giờ (thời biểu hoá) cho việc thực hiện các tiến trình, sao cho chúng âm thầm thực hiện mà bạn không phải chú ý đến (tiến trình **batch**).

Xem "Thực hiện tiến trình hậu trường"

Trách nhiệm hàng đầu của Linux là xử lý các chi tiết liên quan đến việc có nhiều user hoặc nhiều tiến trình làm việc cùng lúc. Một user có quyền xác định những chương trình nào mình muốn thực hiện. Một số lệnh của Linux sẽ giúp user xác định xem khi nào bắt đầu chạy chương trình. Bên cạnh đó user có thể theo dõi tiến trình của mình cũng như của nhiều người khác và trong một vài trường hợp, user có thể thay đổi thứ tự ưu tiên những tác vụ ấy. Nếu là quản trị viên hệ thống, bạn có toàn quyền và trách nhiệm để kích hoạt, theo dõi và quản lý các tiến trình thuộc về hệ điều hành hay thuộc về user.

Bảng 19.2 liệt kê các lệnh dùng để điều khiển những khả năng multiuser và multitasking của Linux.

Bảng 19.2: Các lệnh multiuser và multitasking

Lệnh	Hành động
at	Thi hành lệnh vào giờ hẹn
batch	Thi hành lệnh khi tải suất của hệ thống cho phép
cron	Thi hành những lệnh đã thời biểu hoá
crontab	Duy trì các tệp **crontab** cho user độc lập
kill	Dừng hẳn (xoá bỏ) các tiến trình
nice	Điều chỉnh mức độ ưu tiên của một tiến trình trước khi nó bắt đầu
nohup	Cho phép tiến trình chạy tiếp tục sau khi bạn đã đăng xuất

ps	Hiển thị thông tin về các tiến trình đang chạy
w	Cho biết ai đang ở trên hệ thống và họ đang làm gì
who	Cho biết ai đang ở trên hệ thống

Ghi chú: Muốn có thêm thông tin liên quan đến những lệnh trong bảng 19.2, mời bạn tham khảo các trang **man** theo cú pháp sau đây:

```
man lệnh
```

Ngoài ra bạn có thể xem trực tiếp phần trợ giúp vắn tắt của lệnh:

```
lệnh -help
```

19.2 Chạy đa tiến trình

Bạn có thể kích hoạt một chương trình bằng tên của chương trình ấy, hoặc từ các tệp có chứa lệnh shell. Trong khi thực hiện, chương trình có thể tương tác với nhiều phần khác của hệ thống. Chương trình có thể đọc và ghi vào tệp, quản lý thông tin trong RAM, hoặc gửi thông tin đến máy in, modem, hay những thiết bị khác. Ngoài ra hệ điều hành còn ghép kèm thông tin vào tiến trình để có thể theo dõi và quản lý tiến trình.

Tiến trình là một chương trình đang được thực hiện, song lại khác chương trình. Ở một khía cạnh nào đó, tiến trình hơn chương trình ở chỗ là biết sử dụng tài nguyên của một hệ thống đang chạy, trong khi chương trình chỉ tuần là một loạt các câu lệnh. Ở khía cạnh khác, một chương **menu** thuần của Linux lại biết kích hoạt nhiều tiến trình khác.

Để theo dõi công việc của tiến trình, Linux gán cho mỗi tiến trình một con số định danh gọi là PID (Process Indentifier - ID tiến trình).

19.2.1 Kích hoạt đa tiến trình

Bạn đã biết rằng shell đăng nhập của mình luôn luôn hoạt động. Mỗi khi nhập lệnh, bạn kích hoạt ít nhất một tiến trình mới trong shell đăng nhập vẫn tiếp tục chạy. Chẳng hạn nếu bạn gõ lệnh sau đây, tệp report.txt được gửi đến chương trình 1p:

```
1p report.txt
```

Xem "Tìm hiểu về shell"

Khi chương trình **lp** hoàn thành nhiệm vụ, dấu nhắc shell lại xuất hiện. Tuy nhiên, trước khi bạn thấy dấu nhắc này thì shell đăng nhập và lệnh **lp** vẫn tiếp tục chạy. Trong trường hợp này, bạn đã kích hoạt nhiều tiến trình cùng lúc, mà bạn sẽ quy ước gọi gọn là đa tiến trình. Shell chờ cho **lp** xong việc trước khi đưa dấu nhắc lên màn hình.

19.2.3 Kích hoạt tiến trình hậu trường

Bạn chạy tiến trình hậu trường bằng cách gõ câu lệnh và thêm ký tự & vào phía sau câu lệnh này, rồi bấm <Enter>. Thí dụ:

```
lp report.txt &
```

shell sẽ lập tức phản hồi một con số và đó là PID của tiến trình ấy. Dấu nhắc shell tái xuất hiện mà không cần chờ tiến trình hoàn thành. Sau đây là kết quả của thí dụ trên:

```
$ lp report.txt&
3146
$
```

Ở đây 3146 là PID của tiến trình do lệnh **lp** kích hoạt. Cho dù bạn có chạy lệnh **lp** ở hậu trường hay không, tiến trình **lp** là tiến trình con của shell hiện hành. Thí dụ này cho thấy quan hệ mẹ và con, một quan hệ rất phổ biến giữa các tiến trình. Shell hiện hành của bạn là tiến trình mẹ và **lp** đang chạy là tiến trình con. Thông thường thì mẹ tiếp tục hoạt động mà không phải đợi con, bạn chỉ việc gõ thêm dấu & vào phía sau lệnh nào sẽ kích hoạt đứa con. Như thế trong khi tiến trình con đang chạy, bạn vẫn có thể tiếp tục một công việc khác hoặc lệnh khác.

Ghi chú: nếu bạn đang làm việc trên một terminal ở chế độ dòng văn bản hoặc một phiên làm việc đăng nhập từ xa, thì shell hiện hành của bạn thường là shell đăng nhập. Nhưng nếu bạn đang sử dụng terminal ảo hoặc cửa sổ terminal từ một giao diện đồ hoạ GUI, mỗi phiên làm việc sẽ có shell riêng.

19.2.3 Sử dụng ống dẫn để kích hoạt đa tiến trình

Còn một cách khác nữa để kích hoạt đa tiến trình là sử dụng ống dẫn ngay tại dòng lệnh. Thí dụ muốn hiển thị lên danh sách 10 tệp có nội dung thay đổi gần đây :

```
ls-lt |head| lp
```

Lệnh này kích hoạt ba tiến trình cùng lúc và cả ba đều là con của shell hiện hành. Cách hoạt động của ống dẫn như sau: cả hai lệnh ở hai phía gạch thẳng đứng (|) đều khởi động cùng lúc và chúng không có quan hệ mẹ con gì với nhau mà đều là con của tiến trình đang chạy lúc chúng được tạo ra. Nếu nhìn ở góc độ này, bạn có thể hiểu rằng những lệnh ở hai bên ống dẫn đều là tiến trình con.

Một số chương trình được viết ra với mục đích phát sinh nhiều tiến trình khác.

Thí dụ như lệnh ispell sẽ liệt kê danh sách những chữ trong một văn bản mà Linux không thể tìm thấy khi tra trong từ điển của hệ thống. Lệnh ispell sinh ra vài tiến trình con. Giả sử bạn nhập lệnh:

```
ispell final.ret>final.errs&
```

và màn hình hiển thị kết quả

```
1286
$
```

Ở đây 1286 là PID của tiến trình ispell. Dấu $ cho biết shell sẵn sàng chờ bạn nhập lệnh khác để xử lý. Mặc dù trên ispell có thể sinh ra vài tiến trình con và nhờ chúng hoàn tất, song thực ra bạn không phải đợi gì cả. Trong thí dụ này, shell hiện hành là mẹ của ispell và con của ispell sẽ được xem là cháu của shell đăng nhập. Mặc dù mẹ phải chờ con hoàn tất tiến trình, nhưng bà thì không phải chờ đợi.

Tất cả những thí dụ trên cho bạn thấy là các user tuỳ ý kích hoạt đa tiến trình.

Bạn muốn chờ cho tiến trình con thực hiện xong rồi mới làm tiếp hay không chờ đều được cả. Nếu muốn tiếp tục mà không chờ, có nghĩa là bạn đưa tiến trình con về phía hậu trường. Mục tiếp theo đây sẽ giới thiệu bạn vài lệnh dùng để thời biểu hoá tiến trình.

19.3 Sử dụng các lệnh hẹn giờ

Môi trường Linux có nhiều cách để quản lý việc thi hành lệnh. Linux giúp bạn tạo ra danh sách các câu lệnh và hẹn đến khi nào mới thi hành. Chẳng hạn như lệnh **at** sẽ nhận danh sách lệnh từ bàn phím hoặc từ một tệp, sau đó bắt đầu kích hoạt lệnh đúng vào thời điểm đã hẹn trước. Lệnh **batch** tựa như **at**, chỉ khác **batch** sẽ chạy lệnh theo thời khoá biểu của hệ thống chứ không phải do user đặt ra. Lệnh **cron** kích hoạt lệnh theo chu kỳ và lệnh **crontab** cho phép các user chỉnh sửa tệp cho **cron** sử dụng.

Tất cả những lệnh dạng thời khoá biểu rất có ích để thực hiện công việc vào những thời điểm máy không quá bận. Những lệnh thời khoá biểu cũng giúp thực hiện các script hướng ngoại - chẳng hạn như tác vụ truy tìm cơ sở dữ liệu – vào giờ thấp điểm để giảm chi phí truy cập.

19.3.1 Chạy lệnh hẹn giờ bằng at

Lệnh **at** dùng để thời biểu hoá một vài lệnh Linux. Bạn dùng **at** để xác định ngày, giờ, hoặc cả hai. Lệnh này cần vài đối số và tối thiểu bạn phải cung cấp giờ giấc để chạy những lệnh bạn muốn thực hiện và những lệnh bạn muốn được thực hiện.

Thí dụ sau đây sẽ thi hành công việc vào lúc 1 giờ 23 am. Nếu bạn là người làm việc từ nửa đêm cho đến 12 giờ 23 sáng, lệnh sẽ được thực thi ngay ngày hôm sau. Nếu không lệnh sẽ được chạy vào lúc 1 giờ 23 ngày mai. Nhiệm vụ của máy là in mọi tệp trong thư mục /user/office/reports/ và gửi user tên **lan_anh** một thông báo cho biết là công việc đã được thực hiện vào lúc 1 giờ 23 am.

Bạn hãy gõ lệnh sau đây vào terminal, bấm <Enter> ở mỗi cuối dòng. Xong xuôi, bạn bấm <Ctrl-d> để chấm dứt câu lệnh.

```
at 1:23
lp/usr/office/reports/*
echo "đã in xong các tệp !"| mail – s "xong việc" lan_anh
```

Ghi chú: Những công việc dạng **cron** sẽ được chương này giới thiệu chi tiết. Chúng là các cấu trúc phổ biến nhất để chạy những công việc tự động quản lý hệ thống. Tuy nhiên, bạn phải là root user mới có quyền tạo và chỉnh sửa các mục ghi công việc **cron**. Lệnh **at** cho phép tất cả mọi người chạy lệnh, ngay khi người đó không có quyền hạn root.

Xem "Thiết lập môi trường terminal" chương 18

Những lệnh nhờ **at** hẹn giờ giúp sẽ được lập thành danh sách và ghi ngay phía sau lệnh **at**.

Sau khi lệnh **at** chạy xong, máy sẽ hiển thị:

```
job 756603300.a at Tues Jan 21 01: 23:00 1999
```

Câu này có nghĩa rằng công việc sẽ tiến hành vào lúc 1:23 như đã hẹn. Công việc mang số hiệu 756603300.a

Nếu có nhiều lệnh muốn dùng với **at**, bạn nên ghi chúng vào một tệp. Thí dụ bạn muốn tệp là getdone được ghi vào lúc 10 giờ sáng, hãy gõ:

```
at 10:00<getdone
```

hoặc

```
at 10:00 – f getdone
```

trong đó ký tự "nhỏ hơn" (<) bắt dùng nội dung tệp getdone làm dữ liệu vào cho lệnh **at** và tuỳ chọn – f cho phép bạn xác định tệp lệnh mà không phải chuyển hướng.

Bạn cũng có thể hẹn giờ theo một danh sách để hệ điều hành sẽ đều đặn kiểm tra. Sau đó, hệ sẽ tự động thi hành các công việc này mà bạn khỏi phải đăng nhập. Lệnh **at** luôn chạy ở hậu trường để tối ưu hoá việc sử dụng tài nguyên hệ thống. Mỗi khi có lệnh nào đó chạy xong, kết quả sẽ tự động gửi e-mail về trương khoản của bạn.

Muốn rà soát lại xem mình đã hẹn với **at** từ những công việc nào, bạn gõ **at** –1. Vẫn lấy những thí dụ kể trên, máy sẽ hiển thị:

```
job 756603300.a at Sat Dec 21 01:23:00 1998
job 756604200.a at Fri Jan 24 17:00:00 1999
```

Chỉ những công việc liên quan đến **at** của bạn mới được hiển thị. Muốn gỡ bỏ một công việc đã hẹn với **at**, bạn gõ **at** – d, tiếp theo sau là số hiệu công việc. Chẳng hạn để gỡ bỏ công việc thứ hai ở thí dụ trên, bạn gõ:

```
at – d 756604200.a
```

Bảng 19.3 tóm tắt những cách dùng lệnh **at**

*Bảng 19.3: Tóm tắt các lệnh **at***

Dạng thức	Hành động
at hh:mm	Thời biểu hoá công việc bằng giờ (hh) và phút (mm) theo cách tính 24 giờ/ngày
at hh:mm mm:dd:yy	Thời biểu hoá công việc bằng giờ (hh) và phút (mm), tháng (mm), ngày (dd) và năm (yy)
at – 1	Liệt kê những công việc đã được hẹn
at now + số đơn vị thời gian	Thời biểu hoá sau một số đơn vị thời gian (đơn vị có thể là giờ, phút, ngày, hoặc tuần)
at-d ID công việc	Huỷ công việc có ID tương ứng (bí danh của lệnh atrm)

Nếu là root bạn sẽ có toàn quyền sử dụng những lệnh kể trên. Quyền hạn đối với các user thường được ghi trong các tệp /etc/at.allow và /etc/at.deny. Nếu tệp /etc/at.allow có thực, thì chỉ những user nào có tên trong danh sách mới được sử dụng lệnh **at**. Nếu tệp /etc/at.allow không có thực, hệ thống sẽ rà soát tệp /etc/at.deny và chỉ user nào KHÔNG có tên mới có quyền đó. Nếu cả hai tệp đều không có thực, thì chỉ superuser mới được dùng **at**. Nếu tệp /etc/at.deny có thực nhưng lại rỗng (không có danh sách user) thì bất kỳ ai cũng có quyền dùng **at.**

19.3.2 Thực hiện công việc với batch

Lệnh **at** cho phép bạn quyết định khi nào thi hành một công việc đã hẹn trước. Song không phải lúc nào cũng sử dụng **at** bởi vì có thể máy còn bận nhiều nhiệm vụ khác.

Lệnh **batch** giao cho hệ điều hành quyền quyết định thời gian thích hợp để thực thi một tiến trình. Nói cách khác, nếu bạn thời biểu hoá công việc bằng **batch**, Linux sẽ kích hoạt tiến trình khi nào xét thấy tải của hệ thống không quá nặng. Và cũng như **at,** công việc với **batch** chạy ở hậu trường. Thực ra **batch** chính là bí danh của **at**-b trong phiên bản Red Hat Linux.

Lưu ý: Bạn nên ghi tất cả những lệnh thường chạy với **at** hoặc **batch** vào một tệp, để khi cần sẽ không phải gõ lại những lệnh ấy. Muốn dùng **batch** để thời biểu hoá những lệnh có trong tệp getdone, bạn hãy gõ **batch** <getdone.

Muốn kết hợp lệnh với **batch,** bạn cứ gõ danh sách các lệnh tiếp theo sau **batch.** Khi ghi xong, bạn kết thúc dòng lệnh bằng <ctrl-d>. Bạn có thể ghi hết lệnh vào tệp, sau đó chuyển hướng sang **batch**. Chẳng hạn muốn sắp xếp thứ tự một số tệp, sau đó in kết quả, rồi báo cho user tên **lan_anh** biết rằng công việc đã xong, bạn gõ:

```
batch
sort/usr/office/report/*|lp
echo "tệp đã in xong, Lan Anh ơi!"|mailx – s "xong công việc" lan_anh
```

Máy sẽ hiển thị kết quả:

```
job 7789001234. b at Fri Feb 21 11:43:09 1999
```

Ngày giờ ghi ở dòng kết quả là ngày giờ khi bạn bấm <Ctrl-d> để kết thúc ghi dòng lệnh **batch**. Khi xong công việc, bạn thử kiểm soát hộp thư đến, vì những hiển thị ở màn hình sẽ được máy gửi tới cho bạn.

19.3.3 Thời biểu hoá bằng cron và crontab

at và **batch** đều thi hành những lệnh đã hẹn trước chỉ có một lần rồi thôi. Muốn thời biểu hoá những câu lệnh tiến trình theo chu kỳ, bạn dùng chương trình **cron.** Trong những tệp **crontab**, bạn xác định ngày giờ chạy lệnh theo nhiều hình thức khác nhau như giờ, phút, ngày của tháng, tháng của năm, ngày trong tuần.

Chương trình **cron** chỉ kích hoạt một lần khi hệ thống khởi động. Cá nhân user không được quyền chạy nó trực tiếp. Kể cả bạn là quản trị viên hệ thống, bạn cũng không nên kích hoạt **cron** bằng cách trực tiếp gõ lệnh. Bạn hãy ghi **cron** vào shell script như là trong các lệnh phải kích hoạt trong tiến trình khởi động máy.

Khi được kích hoạt, **cron** (viết tắt chữ chronograph, nghĩa là thời kế) sẽ kiểm tra danh sách các công việc đang xếp hàng ở lệnh **at** để tiến hành. Bên cạnh đó **cron** cũng kiểm tra xem cả root lẫn user khác có thời biểu hoá công việc bằng tệp **crontab** hay không. Nếu không có việc, **cron** "ngủ" và không hoạt động. Tuy nhiên, cứ 60 giây thì **cron** lại "thức giấc" để xem có việc nào đang chờ không. Qua đây bạn thấy **cron** rất quan trọng và hữu dụng, chưa kể việc **cron** chiếm rất ít tài nguyên hệ thống.

crontab dùng để cài danh sách những lệnh hẹn thi hành theo chu kỳ. Lệnh được hẹn chạy vào một thời gian nhất định nào đó, chẳng hạn như mỗi tháng một lần, mỗi ngày

hoặc mỗi giờ một lần, v.v... Danh sách các câu lệnh phải được ghi vào tệp **crontab.** Tệp này được cài đặt cùng với lệnh **crontab.** Sau khi cài xong tệp **crontab**, lệnh **cron** sẽ đọc và thi hành. Ngoài ra **crontab** cũng giúp bạn xem và chỉnh sửa danh sách lệnh trong tệp.

Trước khi cài đặt tệp **crontab**, bạn tạo ra tệp chứa danh sách các câu lệnh, bằng cách dùng những trình soạn thảo văn bản như **emacs**. Sau khi cài xong tệp **crontab**, lệnh **crontab** sẽ quản vị giá trị tệp này. Mỗi user chỉ có một tệp **crontab**, tệp được tạo lúc bạn chạy lệnh **crontab.**

Linux lấy tên user đặt cho tệp **crontab** của user ấy và lưu trong thư mục /var/spool/cron/crontab. Nếu username của bạn là mcn và bạn dùng trình soạn thảo văn bản để tạo ra tệp mycron, sau đó cài đặt nó bằng lệnh **crontab** mycron, Linux sẽ tạo ra tệp /var/spool/cron/crontabs/mcn. Ở thí dụ, Linux tạo ra tệp mcn – nói đúng hơn là Linux ghi nội dung của tệp mycron chồng lên tệp mycron chồng lên tệp mcn.

Ghi chú: Chỉ những user có tên trong tệp /etc/cron.d/cron.allow mới được dùng lệnh **crontab**. Nếu bạn dùng lệnh useradd để thêm user vào hệ thống, tên user này sẽ không được tự động ghi vào tệp vừa kể. Với tư cách là root, bạn phải ghi tên user mới vào tệp ấy bằng trình soạn thảo văn bản.

Mặc dù lúc đầu bạn tạo ra tệp **crontab** bằng trình soạn thảo văn bản, nhưng khi tệp đã tạo ra rồi, bạn chỉ nên thay đổi nó bằng lệnh **crontab** mà thôi. Ngoài lệnh **crontab,** bạn không nên dùng cách nào khác để thay thế hoặc thay đổi tệp dưới sự giám sát của lệnh **cron** (nói cách khác, đó là tệp /var/spool/cron/crontabs/user).

Mỗi dòng trong tệp **crontab** chứa thời biểu của lệnh. Lệnh sẽ được thi hành theo thời biểu. Thời biểu gồm năm trường, được phân cách bằng khoản trắng hoặc tab. Những xuất liệu nào không chuyển cho stdout hoặc stderr được mail cho user. Sau đây là cú pháp những lệnh mà bạn ghi vào tệp để **crontab** sử dụng:

```
phút giờ ngày_của_tháng tháng_của_năm ngày_trong_tuần lệnh
```

Năm trường đầu tiên là tuỳ chọn thời gian mà bạn phải xác định hết. Muốn qua trường nào, bạn thay vào đó dấu sao.

Ghi chú: Về mặt kỹ thuật, dấu sao trong một trường **crontab** có nghĩa là "giá trị, hợp lệ bất kỳ" chứ không phải phớt "lờ" vị trí. Thí dụ mục ghi **crontab** 02 00 01 ** date có nghĩa là chạy lệnh date hai phút sau lúc 0 giờ vào ngày đầu tiên của tháng. Bởi vì hai trường tháng và ngày của tuần đều là sao, mục ghi này sẽ chạy lệnh vào ngày đầu tiên của mỗi tháng vào bất kỳ ngày nào trong tuần, miễn là ngày đầu tiên của tháng rơi vào ngày đó.

*Bảng 19.4: Các tuỳ chọn trường thời gian đi với lệnh **crontab***

Trường	Mốc thời gian
phút	00 đến 59
giờ	00 đến 23 (nửa đêm là 00)
ngày của tháng	01 đến 31
tháng của năm	01 đến 12

ngày trong tuần	01 đến 07 (Thứ hai là 01, chủ nhật là 07)

Trong tệp **crontab** bạn muốn nhập bao nhiêu mục cũng được và tuỳ ý ra lệnh chạy vào thời điểm thích hợp. Điều này có nghĩa là chỉ trong một tệp **crontab**, bạn muốn chạy bao nhiêu lệnh cũng được.

Muốn sắp xếp tệp /usr/wwr/office/weekly, sau đó gửi xuất liệu cho user trên twool vào lúc 7: 30 mỗi sáng thứ, bạn nhập mục ghi như sau vào tệp: 30:07**01 sort/usr/wwr/office/weekly |mail-s "kết quả tài vụ hàng tuần" twool

Lệnh kể trên khẳng định phút 30, giờ là 07, bất kỳ ngày nào trong tháng (bởi dấu sao), bất kỳ ngày nào trong năm (dấu sao thứ hai) và ngày trong tuần là 01 (thứ hai). Bạn lưu ý ký hiệu ống dẫn giữa hai lệnh sort và mail. Trường dành cho lệnh có thể chứa dấu mũi tên, ống dẫn, chấm phảy và bất kỳ những gì bạn có thể đặt vào dòng lệnh shell. Đến thời điểm đã hẹn, **cron** sẽ chạy trọn câu lệnh này bằng shell chuẩn (bash).

Muốn xác định một dãy những giá trị cho một trong bốn trường đầu tiên, bạn phân cách giá trị bằng dấu phẩy. Giả sử bạn có chương trình chkquotes để truy cập dịch vụ cung cấp bản báo giá chứng khoán, sau đó nhập múc báo giá vào tệp. Muốn lấy báo giá vào lúc 9 a.m, 11 a.m, 2p.m và 4 p.m vào các ngày thứ hai, thứ ba, thứ năm hàng tuần, cùng với ngày 10 mỗi tháng ba và tháng chín bạn gõ:

```
*09,11,14,16,10 03, 09 01, 02,04 chkquotes
```

Tiếp theo bạn dùng trình soạn thảo văn bản nào đó như **vi** để lưu tệp dưới dạng txt, mang tên giả định là **cron**jobs. Bạn dùng **crontab** để đặt tệp này vào chỗ mà **cron** có thể tìm thấy, bạn gõ lệnh:

```
crontab cronjobs
```

Mỗi lần sử dụng **crontab** bằng cách này, **crontab** sẽ viết chồng lên bất cứ tệp **crontab** nào mà bạn đã kích hoạt rồi.

Lệnh **crontab** có 3 tuỳ chọn:

Tuỳ chọn –e chỉnh sửa nội dung của tệp **crontab** hiện hành. (tuỳ chọn –e sẽ dùng trình soạn thảo **ed** hoặc trình soạn thảo được gán cho chị EDITOR trong shell của bạn để mở tệp)

Tuỳ chọn –r xoá tệp **crontab** hiện hành khỏi thư mục **crontab**s

Tuỳ chọn –l liệt kê nội dung của tệp **crontab** hiện hành

Quản trị viên hệ thống và các user cùng có trách nhiệm làm cho hệ thống được sử dụng tốt. Mỗi khi thời biểu hoá một tiến trình, bạn hãy nghĩ đến tác động đối với toàn bộ hệ thống. Với Linux, quản trị viên hệ thống có quyền cho hoặc không cho mộ hoặc nhiều user truy cập các lệnh **at**, **cron** và **batch.**

Hỏi-Đáp

*Những lệnh tôi nhập vào tệp **crontab** không chịu chạy.* Lệnh **cron** dùng Shell Bourne Again (bash) để chạy các mục ghi **crontab**. Những mục này sẽ không chạy nếu bạn dùng các đặc điểm không được bash chấp nhận. Thí dụ shell Public Domain Korn (pdksh) cho phép bạn dùng dấu sóng (~) để đại diện home directory, hoặc dùng lệnh alias để chỉ định bí danh cho vài lệnh.

*Khi thử chạy lệnh **at**, máy báo rằng tôi không được quyền sử dụng.* Bạn chưa nhập ID đăng nhập vào tệp /etc/cron.d/at.allow

*Tôi thử dùng **at**, now để chạy ngay một lệnh nào đó. Cho dù gõ phím nhanh cách mấy, **at** now luôn luôn báo lỗi EROR: Too late.* Tôt hơn hết bạn là bạn dùng lệnh **batch** để chạy. Tuy nhiên, bạn có thể dùng **at** now +5 min để chạy lệnh trong vòng 5 phút nữa. Sau khi bấm <Enter>, bạn gõ nhanh lệnh muốn chạy trước khi 5 phút trôi qua.

19.4 Theo dõi và báo cáo môi trường multitasking

Bạn biết rằng Linux là hệ điều hành multitasking và multiuser. Bởi vì cùng lúc có quá nhiều người thao tác trên hệ thống, các user muốn biết xem ai đang có mặt, tiến trình nào đang chạy và họ cũng muốn theo dõi tiến trình thực hiện đến đâu.

Biết rằng người khác đang theo dõi những lệnh mình đã ra là việc quan trọng. hầu hết các user không có quyền truy cập các tệp của bạn nếu bạn không cho phép, nhưng họ có thể biết bạn đã gỡ những lệnh nào. Ngoài ra, như bạn đã biết, quản trị viên hoặc bất kỳ ai có mật khẩu gốc đều có khả năng truy cập mọi tệp trên hệ thống.

Mặc dù bạn không nên quá lo sợ về vấn đề bảo mật khía cạnh riêng tư trên một hệ thống Linux, song bạn cũng nên biết rằng ai cũng có thể theo dõi quan sát những diễn biến trên hệ thống ấy.

Nhờ khả năng này, quản trị viên biết công việc nào đang tiến hành, thực hiện đến đâu để dễ dàng thời biểu hoá những công việc khác. Ngoài ra, user có thể biết xem tiến trình của mình vẫn chạy tốt hay đang hỏng hóc.

19.4.1 Dùng lệnh who để biết ai đang trên hệ thống

Mục đích lệnh who là để biết ai đang đăng nhập who liệt kê tên đăng nhập, đường truyền terminal, cùng với thời gian đăng nhập của các user hiện hành.

who có ích trong nhiều trường hợp. Thí dụ trước khi liên lạc với người bằng lệnh write, bạn kiểm tra bằng lệnh who xem user ấy có mặt hay không.

19.4.2 Dùng who để liệt kê các user đăng nhập

Bạn gõ who, máy sẽ hiển thị đại loại như sau:

```
$who
root console Dec 13 08:00
ernie tty02 Dec 13 10:37
bkraft tty03 Dec 13 11:02
jdurum tty05 Dec 13 09:21
ernie ttys 7 Dec 11 18:49
$
```

Danh sách này cho rằng root, ernie, bkratf và jdurun là 9:21. Bạn cũng có thể thấy rằng ernie đã đăng nhập tại hai terminal, trong đó có một lần ngoài giờ làm việc là 18:49 (6:49 chiều) cách đây hai ngày.

19.4.3 Dùng tiêu đề để liệt kê user

who làm việc với nhiều tuỳ chọn, nhưng chương này chỉ mô tả hai tuỳ chọn để theo dõi các tiến trình trên hệ thống:

-u chỉ liệt kê những user đang đăng nhập.

-H Hiển thị tiêu đề (header) trên từng cột

Những tiêu đề hiển thị cùng với tuỳ chọn –H là NAME, LINE, TIME, IDLW, PID và COMMENTS. Bảng 19.5 giải thích những từ ngữ của phần tiêu đề.

Bảng 19.5: Dạng thức xuất liệu của lệnh who

Trường	Mô tả
NAME	Liệt kê tên đăng nhập của user
LINE	Liệt kê đường truyền hoặc terminal
TIME	Liệt kê ngày giờ đăng nhập
IDLE	Liệt kê giờ và phút kể từ lúc hoạt động gần nhất trên đường truyền này. Máy hiển thị dấu chấm nếu có hoạt động trong vòng một phút gần nhất. Nếu quá 24 tiếng kể từ khi đường truyền được sử dụng, máy hiển thị chữ old.
PID	Liệt kê số ID tiến trình của shell đăng nhập của user
COMMENT	Liệt kê nội dung trường chú thích nếu có liên kết mạng hoặc nếu có chú thích trong /etc/inittab.

Ghi chú: Trên những hệ thống Linux gần đây, bạn sẽ ít khi thấy chú thích trong trường COMMENT. Trước kia, các tiến trình giúp trình giúp bạn đăng nhập UNIX (getty hoặc uugetty) được khởi đầu trực tiếp từ các mục ghi của tệp /etc/inittab nữa.

Thí dụ sau đây sử dụng hai tuỳ chọn –u và –H, sau đó là phản hồi của Linux:

$ who – uH

Name	LINE	TIME	IDLE	PID	COMMENT
root	console	Dec 13 08:00	.	10340	
ernie	tty02	Dec 13 10 :37	.	11929	Tech-89.2
bkraft	tty03	Dec 13 11:02	0:04	4761	Office-23.4
jdurum	tty05	Dec 13 09:21	1:07	10426	
ernie	ttys7	Dec 11 18 : 49	old	10770	oreo.coolt.com

$

Từ danh sách này quản trị viên biết rằng phiên làm việc cuối cùng liên quan đến ernie là từ một site mạng mang tên oreo.coolt.com và đã hơn 24 tiếng đồng hồ mà phiên làm việc này chưa có tín hiệu gì (biết đâu có vấn đề?). Phiên của root và phiên đầu tiên của ernie đều diễn ra trong vòng một phút gần đây. Hoạt động cuối cùng trong phiên làm

việc của bkraft là cách nay bốn phút, trong khi đó với phiên của jdurum, hoạt động kết thúc cách nay một giờ bảy phút.

Bạn hãy ghi nhận danh sách này có số ID tiến trình cho shell đăng nhập từng phiên làm việc của user. Đoạn sau đây sẽ hướng dẫn bạn cách dùng PID để theo dõi hệ thống sâu sát hơn.

19.4.4 Dùng finger

finger là lệnh bổ sung cho web. Để có thông tin về một user, bạn gõ finger tên user (hoặc finger –user@tên miền nếu user này đang làm việc trên máy khác). Thí dụ muốn có thêm thông tin về user tên lan_anh, bạn gõ:

finger lan_anh

Máy sẽ hiển thị:

Login: lan_anh Name: Pla Lan_anh Jr

Directory: /home/lan_anh Shell:/bin/tcsh

Office:2440 SWCary Parkway 114 Office Phone: 919 555 1212

Home Phone: 9192 55 1212

Mail last read Fri Jul 3 17:42 1998 (EDT)

Plan:

Phan Lan Anh

In the immotal words of Socrates:

I drank WHAT?

Xuất liệu cho thấy tên đăng nhập và tên thật kết hợp với trương khoản. Ngoài ra, bạn cũng có thể tham khảo một số chi tiết khác như shell mà user này đang sử dụng. Lệnh finger còn cung cấp tệp mà user đã ghi trong tệp .plan ở home directory.

Như bạn vừa xem, lệnh finger hiển thị rất nhiều thông tin của user và một tin tặc có thể lợi dụng những thông tin này. Đó là lý do mà nhiều quản trị viên thích vô hiệu hoá finger để cho người khác không thể truy cập thông tin.

Ghi chú: Nếu hệ thống cho phép mọi người dùng finger, bạn vẫn có thể ra lệnh chfn để thay đổi thông tin do finger hiển thị. Mời bạn xem các trang **man** tương ứng.

19.4.5 Báo cáo trạng thái các tiến trình bằng lệnh ps

Như tên gọi, lệnh ps (process status) cho biết trạng thái hiện hành của một tiến trình. bạn dùng ps để biết xem tiến trình nào đã hoàn tất, cái nào đang chạy, cái nào bị treo hoặc gặp rắc rối, tiến trình nào chạy bao lâu, sử dùng tài nguyên nào, ưu tiên tương đối của một tiến trình và ID của tiến trình ấy. Nếu bạn gõ lệnh không kèm tuỳ chọn, ps liệt kê ID của từng tiến trình kết hợp với shell hiện hành của bạn.

19.4.5.1 Theo dõi tiến trình bằng ps

Công dụng phổ biến nhất của ps là để theo dõi các công việc hậu trường và những tiến trình khác trên hệ thống. Vì các tiến trình hậu trường không liên lạc với màn hình và bàn phím do đó bạn phải dùng ps để theo dõi.

Danh sách ps trình bày bốn tiêu đề mặc định: PID, TTY, TIME và COMMAND. Bảng 19.6 giải thích các tiêu đề ấy.

Trường Giải thích

PID Số hiệu ID của tiến trình

TTY Terminal nơi phát sinh tiến trình

TIME Thời lượng thi hành tiến trình, tính bằng phút và giây

COMMAND Tên của tiến trình đang thi hành.

Giả sử bạn muốn sắp xếp tệp office.dat, lưu một bản của tệp đã sắp xếp xong thành tên mới là office.srt và gửi tệp đến user mang tên sarah. Và vì bạn cũng muốn tiến trình chạy ở hậu trường, nên bạn gõ:

sort office.dat |tee office.srt |mailx-s"Dữ liệu mại vụ đã sắp xếp" sarah &.

Để quan sát tiến trình, bạn gõ ps và máy và máy hiển thị đại loại như sau:

PID TTY TIME COMMAND

16490 tty02 0:15 sort

16489 tty02 0:00 mailx

16492 tty02 0:00 ps

16478 tty02 0:00 bash

16491 tty02 0:06 tee

16480 tty02 96:45 cruncher

bạn thấy tổng thời gian và PID của mỗi tiến trình cho từng lệnh. Bạn cũng thấy thông tin của shell đăng nhập (bash) và cho chính bản thân ps. Bạn lưu ý thấy mọi lệnh trong ống dẫn đều chạy cùng lúc. Mục ghi cuối cùng của phần hiển thị kể trên báo cáo mộ lệnh đã chạy hơn 90 phút. Nếu đó là vấn đề và bạn muốn dừng hẳn tiến trình, hãy gõ kill. Nếu sau khi gõ ps và chỉ thấy máy hiển thị như sau, có nghĩa là công việc mà bạn dặn chạy ở hậu trường đã hoàn tất:

PID TTY TIME COMMAND

16492 tty02 0:00 ps

16478 tty02 0:00 bash

16480 tty02 99:45 cruncher

Ghi chú: Thỉnh thoảng bạn nên chạy ps để kiểm tra trạng thái một lệnh nào đó. Tuy nhiên bạn đừng cứ mỗi giây lại dùng ps một lần, vì nếu như thế thì chẳng việc gì phải chạy trên hậu trường.

19.4.5.2 Lấy thêm thông tin tiến trình bằng ps

Đôi khi bạn cần nhiều thông tin hơn là những gì ps mặc định. Bảng 19.7 liệt kê một số cờ và chức năng của chúng.

Bảng 19.7: Các cờ thường dùng với lệnh ps

Cờ	Mô tả
-a	Hiển thị cả những tiến trình của user khác
-c	Hiển thị tên lệnh từ môi trường task_struct
-e	Hiển thị môi trường sau dòng lệnh và "and"
-f	Hiển thị cấu trúc đầy đủ của cây gia phả (tiến trình và tiến trình thứ cấp)
-h	Không hiển thị tiêu đề
-j	Dạng thức công việc
-l	Dạng thức dài
-m	Hiển thị thông tin bộ nhớ
-n	Xuất liệu dạng số cho USER và WCHAN. WCHAN là tên của chức năng kernel khi tiến trình đang "ngủ" và đã được sys_ra khỏi tên chức năng. Nếu /etc/psdatabase không có thực, số hiển thị sẽ ở dạng thập lục phân
-r	Chỉ chạy tiến trình mà thôi
-s	Dạng thức tín hiệu
-s	Thêm vào lối thời gian và lỗi CPU con
-txx	Chỉ những tiến trình kết hợp với ttyxx mà thôi
-u	Dạng thức user; cho biết tên user và thời gian khởi đầu
-v	Dạng thức vm (bộ nhớ ảo)
-w	Xuất liệu theo dạng rộng, nghĩa là không cắt xén dòng lệnh để vừa khớp với một dòng màn hình
-x	Hiển thị tiến trình mà không kiểm soát terminal

Lệnh ps chỉ cung cấp một hình ảnh gần đúng của trạng thái tiến trình bởi vì mọi việc có thể thay đổi trong khi ps đang chạy. Lệnh ps chỉ trình bày trạng thái tiến trình trong nháy mắt, ngay vào khoảnh khắc thời gian khi ps thi hành. Khoảnh khắc ấy bao gồm cả bản thân của ps.

Thí dụ sau đây có ba lệnh. Thứ nhất là shell đăng nhập (bash), thứ hai là lệnh sort để sắp xếp tệp inventory, thứ ba là lệnh ps mà bạn đang dùng.

Để biết xe bạn đang chạy tiến trình nào, hãy gõ lệnh:

```
$ps
PID   TTY   TIME COMMAND
65    tty01 0:07 -bash
```

71 tty01 0:14 sort inventory

231 tty01 0:09 ps

Muốn có liệt kê đầy đủ bạn gõ:

$ps-uax

UID	PID	PPID	CSTIME	TTY	TIEM	COMD
amanda	65	10	11:40:11	tty01	0:06	-bash
amanda	71	65	61 11:42:01	tty01	0:14	sort inventory
amanda	231	65	80 11:46 :02	tty01	0:00	ps-f

Bạn lưu ý vài mục trong danh sách đầy đủ này: Ngoài PID còn có PPID. Đó là số ID tiến trình của tiến trình mẹ của tiến trình đang chạy. Ở thí dụ này, tiến trình đầu tiên liệt kê (PID 65) là mẹ của hai tiến trình sau đó. Mục ghi ở cột thứ tư (cột tiêu đề C) cho biết lượng thời gian sử dụng CPU gần nhất của tiến trình. Khi tìm kiếm tiến trình tiếp theo để làm việc, hệ điều hành sẽ chọn tiến trình nào có giá trị C thấp hơn. Cột STIME cho biết thời gian khởi hành của tiến trình.

Muốn quan sát mọi tiến trình trên hệ thống, bạn gõ ps-uax. Khi chuyển hướng lệnh bằng grep $LOGNAME, máy sẽ hiển thị tiến trình theo tên đăng nhập của bạn và loại bỏ không hiển thị các tiến trình thuộc những tên đăng nhập khác. Muốn xem liệt kê đầy đủ các tiến trình của mình, bạn gõ:

ps-uax|frap $LOGNAME

Muốn liệt kê tiến trình của hai terminal tty01 và tty02, bạn gõ:

$ps –t “12”

PID	TYY	TIME	COMMAND
32	tty01	0:05	bash
36	tty02	0:09	hash
235	tty02	0:16	**vi** calendar

Ở thí dụ này, tuỳ chọn –t dùng để giới hạn liệt kê tiến trình liên quan đến hai terminal tty01 và tty02 – Terminal tty01 đang chạy shell lệnh (PID32) và sử dụng **vi** để chỉnh sửa lịch (PID235). Tổng thời gian chạy máy của mỗi tiến trình cũng được liệt kê. Nếu đang sử dụng shell của một giao diện đồ hoạ (lệnh xterm), bạn dùng tên thiết bị pts001, pts002...v.v, với tuỳ chọn –t để thử quan sát tiến trình của các phiên ấy.

Đôi khi tiến trình bị đánh dấu <defunct> (chết), nghĩa là tiến trình đã hoàn tất, tiến trình mẹ đã được báo cáo, nhưng chưa ghi nhận là tiến trình con đã “chết”. Một tiến trình như thế được gọi là zombie (quỷ nhập tràng). Có thể bởi vì trong lúc đó tiến trình mẹ đang bận bịu giải quyết công việc gì đó và sớm muộn gì thì quỷ nhập tràng cũng sẽ biến mất. Nếu bạn bắt gặp nhiều tiến trình chết như thế hoặc cứ hiện diện mà chưa chịu biến mất trong khoảng thời gian khá dài, có nghĩa là hệ điều hành đang gặp ít nhiều trở ngại.

Ghi chú: vì quỷ nhập tràng không có mẹ nên bạn không thể giết nó. Cách duy nhất để tống khứ nó đi là khởi động lại hệ thống.

19.5 Điều khiển đa tiến trình

Với Linux bạn có thể chạy một lúc nhiều tiến trình. Linux cũng cho phép một user hoặc quản trị viên nắm quyền điều khiển các tiến trình đang hoạt động. Quyền điều khiển này rất có ích khi bạn cần thực hiện những điều sau đây:

Kích hoạt và duy trì hoạt động một tiến trình ngay cả sau khi tiến trình mẹ đã ngưng hoạt động (dùng lệnh nohup)

Thời biểu hoá một tiến trình với ưu tiên khác với tiến trình khác (dùng lệnh nice)

Chấm dứt hoặc dừng hẳn tiến trình (lệnh kill)

19.5.1 Sử dụng nohup với tiến trình hậu trường

Theo lệ thường, một tiến trình mẹ ngưng hoạt động thì tiến trình con cũng ngưng theo. Có nghĩa là nếu bạn kích hoạt một tiến trình hậu trường, tiến trình này sẽ chấm dứt khi bạn đăng xuất. Muốn tiến trình ấy vẫn âm thầm hoạt động sau khi bạn thoát khỏi hệ thống, hãy dùng lệnh nohup như ở thí dụ sau:

nohup sort office.dat&

Câu lệnh này bắt lệnh sort đừng quan tâm đến việc bạn đăng xuất khỏi hệ thống và hãy chạy cho đến khi hoàn tất. Nhờ vậy, bạn có thể khởi đầu một tiến trình tự động chạy liên tục trong nhiều ngày thậm chí nhiều tuần. Và trong khi tiến trình đang hoạt động, bạn không cần phải đăng nhập.

khi bạn dùng nohup, lệnh sẽ gửi tất cả xuất liệu và thông báo lỗi của một lệnh đến tệp mang tên nohup.out. (Thường thì những báo lỗi ấy xuất hiện trên màn hình). Bạn xem thí dụ sau:

$ nohup sort office.dat&

1252

Sending output to nohup.out

$

Linux sẽ chuyển tệp đã được sắp xếp và tất cả thông báo lỗi vào tệp nohup.out. Mời bạn xem tiếp thí dụ sau:

$ nohup sort office.dat>office.srt&

1257

Sending output to nohup.out

Các thông báo lỗi được đưa vào tệp nohup.out, nhưng tệp office.dat sẽ được đặt vào office.srt

Ghi chú: Khi sử dụng nohup với ống dẫn, bạn gõ nohup với từng lệnh trong ống dẫn:

nohup sort office.dat/nohup mailx –s “Dữ liệu mãi vụ đã sắp xếp”**lan_anh** &

19.5.2 Dùng nice để thời biểu hoá thứ tự ưu tiên các lệnh

Nếu không có sự can thiệp của nice, tiến trình hoạt động theo ưu tiên đã định sẵn. Bạn dùng nice để hạ bậc ưu tiên của một lệnh, sao cho các tiến trình khác được lập thời biểu sử dụng CPU thường xuyên hơn công việc của nice. Ngoài ra superuser cũng có quyền nâng bậc ưu tiên của một tiến trình.

Ghi chú: lệnh nice theo bản phát hành GNU không thể sử dụng các lệnh nice - -help và nice - -version.

Dạng thức chung của nice như sau:

nice -số lệnh

Cấp độ ưu tiên do đối số - số xác định (số càng cao ưu tiên càng thấp). Số mặc định là 10. nếu ở dòng lệnh có -số, cấp độ ưu tiên sẽ tăng cho đến ngưỡng 20. Nếu bạn gõ lệnh sau đây, tiến trình sort sẽ khởi hành với ưu tiên cấp độ 10: sort office.dat > office.srt&

Giả sử bạn muốn kích hoạt một tiến trình khác - chẳng hạn như lệnh **lp** –nhưng vẫn dành ưu tiên cho lệnh sort, bạn gõ:

nice – 5 **lp** mail_list&

Muốn gán cho **lp** ưu tiên thấp nhất bạn gõ:

nice –10 **lp** mail_list&

Ghi chú: Cờ số kể trên đi sau ký hiệu đặc thù của cờ là dấu gạch nối, bạn không nên nhầm lẫn với dấu trừ, dấu âm.

Chỉ superuser mới có quyền tăng bậc ưu tiên của một tiến trình, bằng cách sử dụng một số âm làm đối số cho nice. Muốn gán mức độ ưu tiên hàng đầu cho nice - - 10 job&

Dầu "và" (&) ở thí dụ này chỉ là nhiệm ý. Nếu công việc đang xử lý thuộc dạng tương tác, bạn không nên dùng dấu "và" để đưa tiến trình vào hậu trường.

19.5.3 Dùng renice để thời biểu hoá thứ tự ưu tiên các tiến trình đang hoạt động

Trên một vài hệ thống có lệnh renice và lệnh này giúp bạn định lại cấp độ ưu tiên của một tiến trình đang hoạt động. Các hệ Berkeley INUX có lệnh renice; trong thư mục /usrsr/ucb các hệ Linux System V để tương thích với hệ Berkeley cũng có renice. Dạng thức renice giống như dạng thức nice: renice -số PID

Muốn thay đổi thứ tự ưu tiên của một tiến trình đang hoạt động, bạn phải biết PID của tiến trình ấy. Muốn biết PID của mọi tiến trình của mình, bạn gõ:

ps –e |grep tên

Ở thí dụ này, tên đại diện tên của tiến trình đang hoạt động. Lệnh grep lược bỏ mọi tiến trình không chứa tên tiến trình mà bạn đang tìm kiếm. Nếu nhiều tiến trình dùng tên đều đang hoạt động, bạn phải xác định tiến trình mình tìm bằng cách xem xét ngày giờ mà tiến trình khởi đầu. Nêu muốn tác động đến mọi tiến trình thuộc về một user

hoặc nhóm user nào đó, bạn xác định GID hoặc UID của một tiến trình đang hoạt động ở lệnh renice.

Mục ghi ở cột thứ hai trong danh sách ps là PID của tiến trình. Ở thí dụ tiếp theo, user cho ba tiến trình cùng hoạt động (chưa kể đến shell của cá nhân user ấy). Tên user là pcoco.

```
$ ps –ef |grep $LOGNAME
pcoco 11805 11804 0      Dec 22      ttysb  0;01 sort office.dat > office.srt
pcoco 19955 19938 4      16:13:02    ttyp0  0:00 grep pcoco
pcoco 19938 1     0      16:11:04    tyyp0  0:00 bash
pcoco 19940 19938 142    16:11:04    ttyp0  0:33 find    -name core –exec rm {}
$
```

Muốn hạ bậc ưu tiên của PID 19940 (tiến trình **find**), bạn gõ:

renice –5 19940

Bạn lưu ý ba điểm như sau đối với renice:

Bạn chỉ có thể dùng renice cho các tiến trình do mình sở hữu.

Superuser có thể dùng renice cho bất cứ tiến trình nào

Chỉ superuser mới có quyền tăng bậc ưu tiên của một tiến trình

19.5.4 Dùng kill chấm dứt tiến trình

Đôi khi bạn muốn chấm dứt vì tiến trình:

Chiếm quá nhiều thời gian của CPU

Chạy quá lâu mà không cho kết quả mong đợi

Phát sinh ra màn hình hoặc ra tệp trên đĩa quá nhiều xuất liệu

Có thể khoá chặt một terminal hoặc một phiên làm việc

Sử dụng tệp không thích hợp để làm xuất hoặc nhập liệu

Không còn hữu ích nữa

Ngoài ra có thể có nhiều lý do khác.

Muốn chấm dứt một tiến trình đang chạy hậu trường, bạn dùng lệnh kill

Muốn chấm dứt một tiến trình không đang chạy ở hậu trường, bạn bấm <Ctrl-c>.

Các phím này không hữu hiệu khi tiến trình chạy ở hậu trường, bởi vì tiến trình ở hậu trường không chịu sự điều khiển của terminal, do đó nhập liệu từ bàn phím không có hiệu quả. Vì thế cách duy nhất để chấm dứt tiến trình ở hậu trường là lệnh kill.

19.5.5 Tiến trình hậu trường: chấm dứt bình thường

Lệnh kill gửi tín hiệu đến chương trình yêu cầu chấm dứt (còn gọi là giết) một tiến trình. Có hai dạng thức để dùng kill:

kill PID (s)

hoặc

kill – tín hiệu PID (s)

Muốn chấm dứt một tiến trình có PID là 123, bạn gõ kill 123. Muốn chấm dứt những tiến trình có PID lad 123, 324 và 73 bạn gõ:

kill 123 324 73.

Dùng kill với -tín hiệu, bạn có thể bắt một tiến trình đang chạy phải đọc lại những tệp cấu hình, hoặc dừng hẳn hoạt động của tiến trình chứ không "giết" hẳn.

Lệnh kill –l sẽ liệt kê nhiều tín hiệu tuỳ chọn. Tuy nhiên một user thường chỉ sử dụng kill không kèm tín hiệu hoặc nhiều lắm chỉ là tín hiệu –9.

Cẩn thận: Nhớ gõ đúng PID của tiến trình, vì gõ nhầm PID sẽ "giết oan" mà bạn vẫn muốn hoạt động tiếp, chưa kể hậu hoạ khôn lường. Xin bạn nhớ rằng nếu đăng nhập với tư cách là quản trị viên, bạn có quyền kill bất kỳ tiến trình nào.

Nếu tiến trình được chấm dứt hoàn hảo, shell sẽ thông báo bạn biết: đơn giản là dấu nhắc shell tái xuất hiện.

Nếu thử kill một tiến trình không có thật hoặc bạn không có quyền kill, máy sẽ báo lỗi.

Giả sử tên đăng nhập của bạn là chris bạn đang ở tty01. Muốn xem những tiến trình mình đang chạy, bạn gõ ps –f và máy sẽ hiển thị như sau:

UID	PID	PPID	C	STIME	TTY	TIME	COMMAND
chris	65	1	0	11:40:11	tty01	0:06	-bash
chris	71	65	61	11:42:01	tty01	0:14	total_updt
chris	231	65	80	11:46:02	tty01	0:00	PS –F
chris	187	53	50	15:32:01	tty02	123:45	crunch stats
chris	53	1	0	15:31:34	tty02	1:06	-bash

Bạn lưu ý thấy chương trình total_updt đang chạy ở terminal hiện hành. Một chương trình khác là crunch đang chạy ở terminal khác và bạn nghĩ rằng chương trình này đã chiếm khá nhiều thì giờ của CPU. Muốn chấm dứt tiến trình này bạn có thể gõ:

kill 187

Muốn chấm dứt tiến trình mẹ của crunch, bạn gõ:

kill 53

Giả sử hệ thống của bạn có nhiều terminal từ xa và bạn phát hiện có user bỏ trống một terminal bỏ trống một terminal đang hoạt động mà không đóng tắt, quản trị viên có thể dùng kill để chấm dứt cả mẹ lẫn con một lúc. Sử dụng thí dụ trên, bạn gõ:

kill 187 53

Lưu ý: Trường hợp terminal của bạn bị khoá hoặc bị treo, hãy đăng nhập một terminal ảo bằng cách bấm <Alt> kết hợp với một phím chức năng (F1-F6), rồi gõ lệnh ps –ef | $LOGNAME, sau đó kill shell đăng nhập của terminal bị treo.

19.5.6 Tiến trình hậu trường: Chấm dứt vô điều kiện

Khi gõ kill, lệnh này sẽ gửi tín hiệu đến tiến trình. Các chương trình Linux có thể gửi hoặc nhận hơn 20 tín hiệu, mỗi tín hiệu được đại diện bằng một con số. Thí dụ bạn đăng xuất, Linux phát tín hiệu ngưng kết nối (tín hiệu số 1) đến mọi tiến trình hậu trường do shell đăng nhập của bạn được kích hoạt kèm nohup. Như đã nói bên trên, sử dụng nohup để kích hoạt một tiến trình hậu trường sẽ cho phép tiến trình ấy tiếp tục hoạt động và phớt "lờ" tín hiệu chấm dứt. Bạn có thể sử dụng chương trình hoặc shell script nào đó có khả năng phớt "lờ" tín hiệu. Khi ra lệnh kill mà không kèm tín hiệu nào, Linux tự động gửi tín hiệu 15 đến tiến trình. Lệnh kill 1234 sẽ gửi tín hiệu 15 đến tiến trình mang PID 1234. nhưng nếu đã được lập để phớt "lờ" tín hiệu 15, tiến trình này sẽ không chấm dứt khi bạn gõ lệnh kill. Tuy nhiên bạn vẫn có cách để kill mà bản thân tiến trình không thể "Chống cự"

Tín hiệu 9 là tín hiệu kill vô điều kiện và luôn hoàn thành nhiệm vụ. Cú pháp như sau:

kill –9 PID

Giả sử bạn gõ ps –f và máy hiển thị như sau:

UID	PID	PPID	C	STIME	TTY	TIME	COMMAND
chris	65	1	0	11:40:11	tty01	0:06	-bash
chris	71	65	61	11:42:01	tty01	0:14	total_updt inventory
chris	231	65	80	11:46:02	tty01	0:00	PS –F
chris	187	53	50	15:32:01	tty02	123:45	crunch stats
chris	53	1	0	15:31:34	tty02	1:06	-bash

Muốn kill tiến trình 187, thương thì bạn gõ kill 187. Sau đó bạn lại gõ ps –f và thấy tiến trình vẫn đang chạy. Bạn hiểu là tiến trình đã được lập để phớt "lờ" kill. Muốn kill vô điều kiện, bạn gõ kill –9 187. Khi gõ lại lệnh ps –f, bạn thấy tiến trình này đã chấm dứt.

Cẩn thận: Việc killl vô điều kiện cũng có điểm bật tiện: lệnh sẽ không cho phép tiến trình hoàn tất những gì sắp làm xong. Nếu một chương trình cập nhật tệp đang chạy mà bạn gõ kill –9, có khả năng tệp không được cập nhật, hoặc bạn mất luôn cả tệp.

Vì vậy bạn nên cẩn trọng và ở hầu hết mọi trường hợp, bạn không nhất thiết phải dùng –9 vì bản thân kill cũng khá đủ.

19.5.7 Chấm dứt tất cả mọi tiến trình hậu trường

Muốn chấm dứt mọi tiến trình hậu trường, bạn gõ:

kill 0

Những tiến trình đang chạy ở hậu trường đôi khi kích hoạt nhiều tiến trình khác và thật vất vả khi phải truy lùng hết mọi PID của các tiến trình bạn muốn chấm dứt. Do đó dùng kill 0 nhanh hơn, vì mọi tiến trình do shell hiện hành kích hoạt sẽ chấm dứt đồng loạt. Trước đó bạn nên gõ lệnh jobs để biết xem shell hiện hành đã kích hoạt những tiến trình hậu trường nào.

Chương 20. In ấn

Trái với suy nghĩ của nhiều người trước đây, cuộc cách mạng tin học đã không thực hiện được viễn cảnh văn phòng làm việc không giấy. Lượng giấy bạn sử dụng hiện nay còn nhiều hơn cách đây 20 năm, khi hệ điều hành UNIX còn ở thời kỳ chập chững, người ta thường soạn thảo và in nhiều tài liệu kỹ thuật. Nhờ vậy UNIX và Linux sẵn có nhiều tiện ích liên quan đến in ấn (hay ít ra cũng là định dạng dữ liệu sắp đi in). Chương này chủ yếu bàn về cách in tệp.

Các chủ đề chính sẽ được đề cập đến:

-Chọn lựa máy in

-Lập cấu hình máy in

-Tiến trình in ấn theo Linux

-Những chương trình in ấn quan trọng

-Những thư mục quan trọng

-Những tệp quan trọng

-Tệp /etc/printcap

-Tạo ra mục ghi thử nghiệm printcap

-Tổng hợp kiến thức

-Lập cấu hình máy in theo Red Hat

20.1 Chọn lựa máy in

Các hệ thống in ấn chung cho BSD UNIX/Linux được gọi là các hệ thống lpr (Line PrinteR). Nếu truy cập được máy in từ DOS, bạn sẽ chuyển được những ký tự ASCII từ Linux về máy in. Nhược điểm duy nhất là Linux không giúp bạn tận dụng những đặc điểm của máy in. Một trong những nguyên nhân chủ yếu là thoạt tiên Linux gửi tệp cần in đến một tệp khác. Các tệp được gửi đến một vùng tạm bởi vì máy in là thiết bị ngoại **vi** tương đối chậm, trong khi đó Linux không muốn cả hệ thống phải làm việc chậm lại chỉ vì in một tệp. Tiến trình này được gọi là spooling (ống dẫn ra máy in) và từ đó máy in được gọi là thiết bị spool. Khi bạn ra lệnh in trong Linux, thay vì đi thẳng đến máy in, tệp sẽ chạy vào xếp hàng chờ tới phiên mình. Nếu tệp của bạn đứng vị trí đầu tiên, hầu như nó sẽ được in ngay tức khắc.

Nhờ thừa hưởng nhiều tính năng của UNIX, Linux hỗ trợ nhiều loại máy in.

GHI CHÚ Spool đại diện cho cụm từ Simultaneous Peripheral Operation Off Line (Thao tác Ngoại **vi** Đồng hành Ngoại tuyến), ra đời vào thời sơ khai của những máy in cỡ lớn IBM mainframe. Lúc này các máy tính nhỏ in báo cáo ngoại tuyến từ mainframe. Kỹ thuật này cho phép các máy tính lớn đắt tiền tiếp tục công việc quan trọng của mình mà không mất thời gian cho những chuyện in ấn vặt vãnh.

Chương này giả định rằng bạn đã am tường cách soạn thảo một tệp văn bản với Linux và bạn cũng có hiểu biết cơ bản về quyền sở hữu, cùng với các quyền hạn (quyền hạn) đối với tệp. Đặc biệt nếu bạn dự định tiến hành tác vụ in ấn từ xa, các hệ thống thứ cấp mạng của bạn phải cài đặt sẵn và chạy tốt. Bạn nên xem thêm thông tin tại các trang **man** của hai lệnh **chmod** và chown. Ngoài ra bạn xem Chương 8, "Sử dụng trình soạn thảo **vi**" để hâm lại các kiến thức hữu quan, bởi vì bạn sẽ phải soạn thảo nhiều tệp khi lập cấu hình cho máy in

20.2 Lập cấu hình máy in

GHI CHÚ Linux xem printer như là một tệp bình thường (xem minh giải tiếp theo sau đây). Nhưng vì máy in là phần cứng vật lý nên phải được kê khai trong thư mục /dev. Linux thích đối xử các thiết bị vật lý như là thành phần của hệ thống tệp.

Từ printer (máy in) dùng để chỉ một máy in được xác định trong /etc/printcap. Từ physical printer (máy in vật lý, vật chất) để chỉ thiết bị in chữ ra trang giấy. Do đó /etc/printcap có thể mang nhiều mục ghi mô tả chỉ một máy in vật lý, nhưng với nhiều cách in khác nhau. Nếu những lời giải thích này vẫn còn tối nghĩa, mời bạn tham khảo thêm đoạn nói về /etc/printcap.

20.3 Tiến trình in ấn theo Linux

Cách đơn giản nhất để in ấn theo Linux là gửi dữ liệu in trực tiếp đến thiết bị in. Lệnh sau đây sẽ gửi một bản liệt kê thư mục đến máy in song song đầu tiên (mà DOS thường gọi là LPT1):

ls> /dev/lp0

Tuy đơn giản nhưng lệnh này không tranh thủ được đặc tính multitasking của Linux, bởi vì thời gian hoàn tất mệnh lệnh in cũng dài bằng thời gian mà máy tiêu tốn để in xong dữ liệu. Nếu rơi nhằm một máy in chậm chạp, hoặc máy không được đánh dấu chọn, hoặc nữa là máy không được kết nối, bạn sẽ tốn khối thì giờ. Tốt hơn là bạn nên spool dữ liệu, nghĩa là tập hợp dữ liệu in vào một tệp, rồi kích hoạt một tiến trình hậu trường để gửi dữ liệu đến máy in. Spool tệp để xếp hàng chờ in là cách làm chủ yếu của Linux. Mỗi máy in được dành riêng một vùng spool. Dữ liệu chờ in được tập kết tại vùng này theo dạng một tệp cho mỗi tác vụ in. Một tiến trình hậu trường (gọi là daemon máy in) thường xuyên thăm viếng vùng in xem có tệp nào mới vào xếp hàng hay không. Khi có tệp mới, dữ liệu được gửi đến máy in thích hợp. Thao tác này được gọi là despool, giống như gạch tên ra khỏi sổ điểm danh. Khi có nhiều tệp đang xếp hàng, thì ai đến trước sẽ được phục vụ trước. Do đó vùng spool thực sự là một hàng đợi và những công việc in đang chờ đến lượt được gọi là xếp hàng chờ in.

Với trường hợp in ấn từ xa, thoạt tiên dữ liệu được spool tại chỗ như thường lệ, sau đó tiến trình hậu trường sẽ gửi dữ liệu đến một máy in từ xa thuộc một máy **vi** tính từ xa.

Daemon máy in cần có các thông tin về thiết bị vật lý sắp sử dụng, vùng spool nào cần thăm viếng, máy **vi** tính và máy in từ xa nào để chuyển dữ liệu,... Những thông tin này trữ trong tệp /etc/printcap. Bạn sẽ bàn bạc cụ thể về chi tiết tệp này ở đoạn "Tệp /etc/printcap".

20.4 Những chương trình in ấn quan trọng

Hệ thống in ấn của UNIX gồm năm chương trình. Bảng sau mô tả vị trí mặc định và các quyền hạn của chúng. Năm chương trình này là thành viên của nhóm daemon và thuộc quyền thao tác của root.

Bảng Các chương trình in ấn quan trọng

Quyền hạn của tệp	Vị trí của tệp
-rwxr-x-x	/usr/bin/lpr
-rwxr-xr-x	/usr/bin/lpq
-rwxr-xr-x	/usr/bin/lpc
-rwxr-xr-x	/usr/bin/lprm
-rwxr-xr---	/usr/bin/lpd

Bốn quyền hạn đầu tiên trong bảng là để đăng ký, huỷ bỏ và kiêm tra công việc in ấn. /usr/bin/lpd là daemon máy in.

GHI CHÚ Bảng trên không nhất thiết phải mô tả chính xác vị trí, quyền sở hữu và các quyền hạn y như trên máy của bạn. Do đó bạn chỉ nên ghi nhận tệp và quyền hạn lpd.

Mọi lệnh trên đều có trang **man** tương ứng, bạn nên xem qua để lấy thêm thông tin. Những điểm quan trọng cần lưu ý là theo mặc định, cả lpr, lprm, lpc và lpq đều hoạt động trên một printer gọi là **lp**. Nhưng nếu bạn định nghĩa một biến môi trường gọi là PRINTER, máy sẽ dùng định nghĩa này. Bạn có thể ghi đè **lp** và biến PRINTER bằng cách xác định tên printer phải sử dụng:

lpc –PMYPRINTER

Daemon lpd

Linux xử lý mọi công việc in ấn qua trung gian lpd. Nếu tiến trình này không chạy thì không thể in ấn gì được: các tệp chờ in sẽ nằm im trong thư mục spool tương ứng, đợi đến khi tiến trình lpd được kích hoạt.

Xem "Tìm hiểu các tiến trình"

Nếu hệ thống của bạn không nạp lpd khi khởi động, hoặc nếu vì lý do nào đó khiến bạn phải kill và khởi động lại lpd, lệnh sau đây sẽ kích hoạt daemon máy in:

lpd [tuỳ chọn]

Danh sách tuỳ chọn nằm ở trang **man** của lpd. Khi lập cấu hình máy in cho Linux, tuỳ chọn -l rất quan trọng bởi vì sẽ tạo ra một tệp ghi sổ (tệp biên bản - log file) lưu lại mọi yêu cầu in ấn của hệ thống. Tệp này có ích khi bạn gỡ rối hệ thống in ấn.

Lệnh lpr

Lệnh lpr đăng ký công việc với máy in, hoặc xếp một công việc vào hàng chờ in. Khi bạn gõ lệnh, máy sẽ chép tệp cần in vào thư mục spool. Mỗi máy in trên hệ thống Linux đều phải có thư mục spool riêng cho mình. Kích cỡ thư mục spool được xác

định bởi tệp minfree nằm trong từng thư mục. Tệp này xác định số block đĩa cứng để spool tệp đến máy in. Cần khai báo kích cỡ để cho daemon lpr không thể chiếm hết cả ổ cứng khi spool yêu cầu in. lpr sẽ truy tìm ra tệp, sau đó chuyển dữ liệu đến máy in vật lý.

Nếu bạn không ghi tên tệp nào, lpr sẽ sử dụng stdin.

Lệnh lpq

Lệnh lpq liệt kê nội dung thư mục spool của một máy in nhất định. Một trong những thông tin quan trọng được hiển thị là ID công việc. Khi cần huỷ bỏ một công việc in ấn, bạn phải khai báo số ID này. lpq cũng dùng con số để chỉ thứ tự của công việc trong hàng chờ đợi in. active có nghĩa là tệp đang được in - hoặc ít ra cũng đang được lpd chuẩn bị đưa lên dàn in.

Lệnh lprm

Lệnh lprm gỡ bỏ công việc khỏi hàng chờ in - nghĩa là gỡ bỏ khỏi thư mục spool những tệp chưa kịp in. Bạn xác định các ID công việc cần gỡ bỏ (bạn có thể dùng lệnh lpq để máy liệt kê ID công việc)

Nếu bạn ra lệnh lprm với tư cách là root, tất cả công việc in ấn sẽ bị huỷ bỏ. Nếu là root và bạn muốn gỡ bỏ tất cả công việc in ấn của một user, bạn chỉ việc xác định username.

Lệnh lpc

Lệnh lpc giúp bạn kiểm tra trạng thái máy in và điều khiển một số chi tiết sử dụng máy in. Đặc biệt với lpc bạn có thể kích hoạt và dừng hẳn việc despool các máy in. Ngoài ra bạn cũng có thể vô hiệu hoá hoặc hữu hiệu hoá máy in và sắp xếp lại thứ tự in ấn. Lệnh sau đây sẽ vô hiệu hoá công việc in ấn trên myprinter, cho phép spool trên yourprinter và di chuyển công việc số 37 lên hàng đầu dãy xếp hàng chờ in:

lpc down myprinter

lpc enable yourprinter

lpc topq 37

Nếu bạn nhập lệnh lpc không kèm đối số, lpc sẽ trở thành tương tác và chờ bạn xác định hành động. Một vài lệnh quan trọng hơn được liệt kê ở Bảng dưới đây, bạn nên xem thêm các trang **man**. Hầu hết các lệnh lpc đều lấy tên của máy in làm tham số (mô tả trong /etc/printcap)

Bảng Vài lệnh lpc phổ biến

Lệnh	Tham số	Mô tả
stop	máy in	Dừng hẳn máy in, nhưng vẫn spool yêu cầu in ấn
start	máy in	Cho phép máy bắt đầu in những tệp đã được spool từ trước. và in những tệp

		nào vừa được spool đến máy in ấy.
exit, quit	(không)	Thoát khỏi chế độ tương tác của lpc
status	máy in	Hiển thị trạng thái hiện hành của máy in, status cho biết xem hàng chờ đợi (queue) có được bật lên hay không, máy in có được phép hoạt động không và số lượng công việc in ấn đang chờ trong hàng đợi

Lưu ý: Một số user cho biết lpc có thể hiển thị sai trạng thái, thậm chí đôi khi treo nguyên cả hệ thống.

20.5 Những thư mục quan trọng

Công việc in ấn chỉ có một thư mục quan trọng, đó là vùng spool, nơi dữ liệu in được tập hợp trước khi /usr/sbin/lpd tiến hành in. Tuy nhiên, nhiều khi hệ thống thiết lập từng thư mục spool cho từng máy in, để cho tác vụ quản lý in ấn dễ dàng hơn. Chẳng hạn, một hệ thống dùng /var/spool/lpd làm vùng spool chính và bên dưới là thư mục cho từng máy in riêng lẻ, cùng tên với máy in. Do đó máy in mang tên ps_nff sẽ có thư mục spool là /var/spool/lpd/ps_nff.

Quản trị viên nên đặt thư mục spool trong nhóm daemon, đồng thời nên để cho tất cả user, nhóm và mọi người khác được quyền đọc và ghi. Nói cách khác, sau khi tạo ra thư mục, bạn hãy dùng lệnh **chmod** ghi quyền hạn là rwxrwxr-x(0775). Đối với thư mục myprinter, lệnh như sau là thích hợp:

chmod ug=rwx, o=rx myprinter chgrp daemon myprinter

Xem "Quyền hạn đối với tệp", Chương 20.

Ghi chú: Vị trí, quyền sở hữu và các quyền hạn kể trên đều mang tính thí dụ.

20.6 Những tệp quan trọng

Ngoài những chương trình liên quan đến in ấn mà bạn vừa xem qua ở bên trên, mỗi thư mục spool còn chứa các tệp có quyền hạn -rw-rw-r-:

Tệp /etc/printcap mô tả đặc tính của mỗi máy in đã được đặt tên trong hệ thống

Tệp .seq chứa bộ đếm số công việc in ấn cho lpr dùng để gán.

Tệp trạng thái chứa thông báo cho lpc start dùng làm báo cáo.

Tệp khoá để bắt lpd dừng thực hiện hai công việc in ấn cùng lúc trên một máy in.

Tệp báo lỗi với nhiệm vụ ghi nhận tất cả hỏng hóc của máy in.

Khi tiến hành in, Linux không yêu cầu có tệp báo lỗi, nhưng lpd cần để ghi chép các hỏng hóc. Bạn có thể đặt cho tệp này bất kỳ tên gì cũng được, miễn là có khai báo tên ấy trong /etc/printcap. Tệp báo lỗi thường được user khai báo thủ công khi thiết kế vùng spool. Mời bạn xem thêm chi tiết ở đoạn "Tổng hợp kiến thức" trong chương này.

20.7.1 Tệp /etc/printcap

/etc/printcap là một tệp văn bản, phải thuộc quyền sở hữu của root và ghi quyền hạn -rw=r--r--.

Thoạt trông, nội dung /etc/printcap có vẻ bí hiểm, nhưng khi đã tìm hiểu hoạt động của tệp, bạn sẽ rành mạch.

Tuy nhiên, vấn đề phức tạp ở chỗ vài bản Linux lại không có các trang **man** của printcap. Hơn nữa, nhiều chương trình và nhiều lập trình viên khi tạo ra các tệp printcap đều không chủ ý làm cho tệp dễ đọc và dễ hiểu. Do đó vì lợi ích bản thân, bạn nên thiết kế tệp printcap của mình sao cho càng lô gích càng tốt, với thật nhiều ghi chú kèm theo. Và nhớ tham khảo các trang **man** của lpd.

Mỗi mục ghi printcap mô tả một máy in, gồm một tên logic cho một thiết bị vật lý, sau đó mô tả cách thao tác những dữ liệu gửi đến thiết bị ấy. Chẳng hạn mục ghi sẽ xác định thiết bị vật lý nào được sử dụng, dữ liệu cho thiết bị ấy được lưu vào thư mục spool nào, sẽ dùng công đoạn tiền xử lý nào để thao tác dữ liệu, các báo lỗi của thiết bị vật lý sẽ được ghi vào đâu,.v.v.. Quản trị viên có thể hạn chế lượng dữ liệu gửi đến máy in cho từng công việc in ấn, hoặc giới hạn quyền truy cập máy in đối với những dạng user nào đó. Đoạn sau đây minh hoạ một máy in được mô tả trong tệp printcap.

Mục ghi printcap mẫu với hai bí danh myprinter | laserwriter:\#**lp** là thiết bị nhận dữ liệu in - ở đây chính là máy in song song thứ nhất :**lp**=/dev/lp0:\#sd có nghĩa là thư mục spool - nơi tập kết dữ liệu :sd=/var/spool/lpd/myprinter.

Bạn có thể tạo ra nhiều mục ghi printcap để mô tả những cách xử lý khác nhau đối với những dữ liệu gửi đến cùng một máy in. Thí dụ một máy in vật lý có thể chấp nhận dạng thức dữ liệu của cả PostScript và HP LaseJet, tuỳ thuộc thông tin thiết lập máy in mà mỗi user gửi về trước khi in. Tốt hơn nên định nghĩa hai máy in: một máy để tiền xử lý dữ liệu in của HP LaseJet và một máy khác để cho PostScript. Sau này chương trình nào phát sinh dữ liệu HP sẽ gửi đến HP và chương trình nào phát sinh dữ liệu PostScript sẽ gửi về cho PostScript.

Những chương trình nào thay đổi dữ liệu trước khi gửi đến máy in vật lý được gọi là bộ lọc (filter). Có trường hợp bộ lọc không gửi dữ liệu nào đến máy in vật lý cả, đó là khi bộ lọc này đã lượt hết mọi thứ.

Ghi chú: Nếu bạn không dùng biến môi trường để chỉ định một máy in, mặc định, hoặc không mô tả một máy in trên dòng lệnh lpr, Linux sẽ chuyển công việc in ấn đến máy in **lp**. Vì vậy bạn nên chỉ định một trong những máy in ở tệp printcap làm máy in **lp**.

Tìm hiểu các trường trong /etc/printcap

Khuôn khổ chương này không đủ để mô tả mọi trường của tệp printcap, do đó bạn chỉ tìm hiểu những trường quan trọng nhất mà thôi. Ngoại trừ tên của máy in, tất cả trường của /etc/printcap đều nằm giữa hai dấu hai chấm và có mã hai mẫu tự. Tiếp theo sau mã này là một giá trị, tuỳ theo loại trường. Có ba loại trường: chuỗi, boolean và số. Bảng sau mô tả những trường phổ biến và quan trọng nhất.

Bảng Các trường của /etc/printcap

Trường	Loại	Mô tả
lp	Chuỗi	Xác định thiết bị để gửi dữ liệu về, thí dụ /dev/lp0
sd	Chuỗi	Xác định tên thư mục spool cho máy in hiện hành
lf	Chuỗi	Xác định tệp ghi nhận báo lỗi cho máy in hiện hành
if	Chuỗi	Xác định tên bộ lọc nhập liệu
rm	Chuỗi	Xác định tên của máy chủ tiếp nhận in từ xa
rp	Chuỗi	Xác định tên của máy in từ xa
sh	Boolean	Xác định không in tiêu đề
sf	Boolean	Xác định không đưa thêm giấy in vào khi hết việc
mx	Số	Xác định số block tối đa cho phép đối với công việc in

Trường lp

Nếu bạn khai báo thiết bị in là /dev/null, mọi tiến trình khác đều diễn biến bình thường, nhưng dữ liệu sau cùng lại chui vào cái sô đựng bít (bit bucket), nghĩa là chẳng đi đến đâu cả. Xuất dữ liệu in về cõi mông lung ít khi có lợi ích gì, trừ trường hợp thử nghiệm cấu hình máy in. Khi thiết kế một máy in từ xa (nghĩa là khi khai báo các trường **rm** và rp), bạn hãy xác định:

:**lp**=:

Không nên bỏ trống trường này trừ khi bạn sử dụng một máy in từ xa. Nếu bỏ trống, máy sẽ báo lỗi.

Trường lf

Bất kỳ tệp nào được khai báo phải có thật, nếu không máy sẽ không ghi nhận báo lỗi nào cả.

Trường if (p526,s494)

Bộ lọc nhập liệu là những chương trình lấy dữ liệu in từ stdin, sau đó phát sinh xuất hiện ở stdout. Công việc tiêu biểu của bộ lọc nhập liệu là dò ra các ký tự ASCII sau đó chuyển mã thành PostScript. Nói cách khác ở đầu vào là văn bản thô, đầu ra là văn bản PostScript.

Khi bạn đã khai báo bộ lọc nhập liệu, daemon máy in không gửi thẳng dữ liệu in ở trong spool đến thiết bị. Thay vào đó, daemon sẽ dùng dữ liệu ấy làm stdin cho bộ lọc nhập liệu và thiết bị in là stdout.

Trường rm và rp

Trước khi gửi dữ liệu in đến một máy in kết nối đến một máy **vi** tính khác, bạn khai báo máy **vi** tính từ xa là **rm** và máy in từ xa là rp và nhớ chừa trống trường thiết bị in **lp**.

Ghi chú: Dữ liệu in vẫn được spool tại chỗ trước khi chuyển đến máy từ xa. Do đó các bộ lọc nhập liệu vẫn hoạt động.

Trường sh và sf

Làm việc một mình một máy, có lẽ bạn ít để ý đến các trang tiêu đề, do đó bạn nên xác định sh.

Nếu máy in của bạn đặc biệt dùng để in dữ liệu từ những gói phần mềm soạn thảo văn bản, bạn nên xác định sf để huỷ việc đưa giấy vào máy. Hầu hết các gói soạn thảo đều tạo ra nguyên trang văn bản đầy đủ dữ liệu, do đó nếu để cho daemon in ấn (của máy bạn) đưa thêm giấy vào mỗi lần xong một công việc in, thì mỗi lần như thế máy in sẽ nhả ra một tờ giấy trắng. Tuy nhiên, nếu máy in đặc biệt dùng để in danh sách liệt kê chương trình hay thư mục, bạn nên bắt máy đưa thêm tờ giấy mỗi lần hoàn tất một công việc in, như thế khi in ra, mỗi danh sách mới sẽ khởi hành từ đầu tờ giấy mới.

Trường mx

Trường mx giúp bạn giới hạn kích cỡ dữ liệu đi vào spool. Bạn khai con số BUFSIZE block (Linux lấy đơn vị 1KB). Nếu khai zero thì sẽ không còn giới hạn nào cho công việc in, trừ dung lượng còn lại của ổ cứng.

Ghi chú: Giới hạn ấn định kích cỡ vùng spool, chứ không áp dụng cho dữ liệu gửi về máy in vật lý.

Nếu user nào định vượt quá giới hạn này, Linux sẽ tự động xén bớt tệp và user sẽ thấy máy thông báo:

(lpr: tên_tệp: copy file is too large)

Đối với những máy in phi-PostScript, giới hạn này rất hữu ích khi có những user hoặc chương trình vô tình hay cố ý tạo ra lượng xuất liệu khổng lồ. Giới hạn này không quan trọng đối với máy in PostScript bởi vì ít có dữ liệu đã spool của PostScript tạo ra được lượng xuất liệu lớn.

Lập biến môi trường PRINTER

Bạn nên thêm một dòng vào script đăng nhập của mình - thậm chí vào thẳng script đăng nhập mặc định user - để lập biến môi trường PRINTER. Chẳng hạn đối với shell bash, bạn viết export PRINTER = myprinter. Nhờ vậy mọi người sẽ khỏi phải khai báo - Pmyprinter mỗi lần đăng ký công việc in ấn.

Muốn thêm máy in vào, chỉ cần lặp lại thao tác kể trên với nhiều tân máy in khác nhau. Hãy nhớ là bạn có quyền ghi nhiều mục vào printcap, mọi mục ấy đều sử dụng cùng một thiết bị vật lý. Như thế bạn có thể xử lý một thiết bị bằng nhiều cách, tuỳ vào tên mà bạn gọi thiết bị khi đăng ký công việc in.

20.7.2 Tạo ra mục ghi thử nghiệm printcap

Shell script sau đây là một bộ lọc nhập liệu đơn giản - nó làm thao tác đơn giản là xếp nhập liệu tiếp theo sau đoạn kết của một tệp trong /tmp, sau một tiêu đề (banner). Bạn xác định bộ lọc này tại mục ghi printcap và khai báo /dev/null là thiết bị in. Thực ra thiết bị in không bao giờ được dùng đến những bạn phải khai báo, nếu không daemon máy in sẽ báo lỗi.

```
#!/bin/sh
# This file should be placed in the print's spool
# directory and named input_filter. It should be
#owned by root, group daemon, and be world executable
#(-rwxr-xr-x).
echo----------->>/tmp/
date>>/tmp/
echo----------->>/tmp/
cat >> /tmp/
```

Ở mục ghi printcap sau đây, bạn để ý thấy cách viết tương đối dễ hiểu và tác giả sử dụng ký tự tiếp nối (\) ở mọi dòng, trừ dòng cuối:

```
myprinter|myprinter:\
:lp=/dev/null:\
:sd=/var/spool/lpd/myprinter:\
:lf=/var/spool/lpd/myprinter/errs:\
:if=/var/spool/lpd/myprinter/input_filter:\
:mx#0:\
```

:sh:\

:sf:\

20.8 Tổng hợp kiến thức

Để tổng hợp tất cả những kiến thức kể trên, những bước sau đây sẽ hướng dẫn bạn lập cấu hình một máy in đơn lẻ trên /dev/lp0. Bạn có thể mở rộng thí dụ này sang những máy in khác. Nhân tiện cũng nhắc nhở rằng phải là root mới thực hiện được công việc này.

Kiểm tra vị trí và quyền hạn của lpr, lprm, lpc, lpq và lpd (Tham khảo Bảng Vài lệnh lpc phổ biến)

Tạo thư mục spool cho máy in (đặt tên là myprinter). Bảo đảm rằng cả thư mục và máy in đều thuộc quyền sở hữu của root, thuộc về nhóm daemon, rằng user và nhóm được quyền ghi, nhưng tất cả những người khác chỉ đọc mà thôi (-rwxrwxr-x). Bạn gõ những lệnh sau: (thật ra bạn không cần thiết phải tạo thư mục, các trình cấu hình hệ thống đã làm thay cho bạn đằng sau hậu trường)/

mkdir /var/spool/lpd

mkdir /var/spool/lpd/myprinter

chown root.daemon /var/spool/lpd/var/spool.lpd/myprinter

chmod ug=rwx, o=rx /var/spool/lpd/var/spool.lpd/myprinter

Tại thư mục /var/spool.lpd/myprinter, tạo ra các tệp cần thiết, gán quyền hạn và quyền sở hữu. Bạn gõ lệnh sau:

cd /var/spool.lpd/myprinter

touch .seq errs status lock

chown root.daemon.seq errs status lock

chmod ug-rw, o=r .seq errs status lock

Tạo shell script input_filter trong thư mục /var/spool.lpd/myprinter. Bạn tạm mượn bộ lọc nhập liệu của đoạn "Tạo ra mục ghi thử nghiệm printcap" làm bộ lọc cho thí dụ này. Phải bảo đảm tệp thuộc quyền sở hữu của root, thuộc về nhóm daemon và mọi người có quyền hạn thi hành. Bạn gõ:

cd /var/spool.lpd/myprinter

chmod ug=rwx, o=rx input_filter

Nếu chưa có, hãy tạo ra tệp /etc/printcap. Xoá mọi mục ghi bên trong và gõ vào mục ghi printcap như ở đoạn "Tạo ra mục ghi thử nghiệm printcap". Bảo đảm tệp thuộc quyền sở hữu của root và những người khác chỉ có quyền đọc mà thôi. Bạn dùng **chmod** để lập quyền hạn -rw-r--r-- (hoặc 644 bát phân).

Chỉnh sửa tệp rc.local (dùng trình soạn thảo ASCII nào cũng được, chẳng hạn như **vi** hoặc **emacs**). Thêm dòng /urs/sbin/lpd ở cuối tệp để chạy daemon máy in mỗi khi hệ thống khởi động. Đến đây bạn chưa cần phải khởi động, song bạn chạy thủ công bằng lệnh lpd thử xem.

Gõ lệnh sau để in thử:

ls -l | lpr - Pmyprinter

Gõ lệnh **ls** để tìm tệp testlp.out trong /tmp. Tệp này phải chứa liệt kê thư mục của bạn. Bạn có thể kiểm tra lại bằng lệnh **more**, **less**, hoặc **cat**. Nếu quên, bạn xem lại Chương 17, "Quản lý tệp và thư mục"

Bạn dùng một trình soạn thảo ASCII chẳng hạn như **vi** để chỉnh sửa /etc/printcap như sau:

Ở mục ghi thứ nhất, đổi mọi myprinter trên dòng thứ nhất thành testlp.

Ở mục ghi thứ hai, đổi /dev/null thành thiết bị thật của máy bạn, thí dụ như /dev/lp0

Ở mục ghi thứ ba, xoá toàn bộ dòng **if**

Đến đây bạn chép mục ghi myprinter để có hai mục ghi giống nhau trong tệp.

Bạn khởi động lại hệ thống, hoặc hãy kill daemon máy in và kích hoạt lại. Bạn làm thao tác này là vì khi khởi động lần vừa rồi, daemon máy in chỉ tham khảo tệp /etc/printcap.

Bạn dùng lệnh **ls** -l | lpr - Pmyprinter để in thử lại một lần nữa. Lần này phải có kết quả ở đầu ra máy in vật lý.

GỜ RỐI:

Máy thông báo lpd:connect: No such file or directory. Daemon máy in /etc/rc.d/init.d/lpd không chạy. Có thể bạn quên ghi thêm daemon này vào tệp /etc/local. Hoặc bạn có thêm nhưng chưa khởi động máy lại. Như vậy, bạn hãy ghi thêm và khởi động lại, hoặc bạn gõ lệnh chạy /etc/rc.d/init.d/lpd. Bạn phải là root mới có quyền làm điều này.

Máy thông báo Job queued, but cannot start daemon. Cùng vấn đề với câu hỏi trên.

Máy thông báo lpd: cannot create spooldir/.seq. Bạn chưa tạo ra thư mục spool như đã mô tả tại mục ghi printcap hoặc là bạn có tạo ra nhưng lại đặt tên sai. Còn một khả năng nữa là đĩa cứng còn quá ít chỗ trống, tuy nhiên khả năng này rất yếu.

Máy thông báo lpr: Printer queue is disabled. Nếu là root, bạn gõ lệnh lpc enable tênmáyin để làm cho máy in có hiệu lực. ***Ghi chú:*** Là root, bạn có thể chuyển công việc đến ngay cả một máy in đang bị vô hiệu hoá.

Tôi chuyển công việc in đến máy in, máy không báo lỗi gì cả nhưng máy in vật lý không in ra. Bạn xem xét một trong những vấn đề sau đây:

Máy in bạn đã bật chưa, có được chỉ định là máy làm việc chưa và có kết nối vật lý với thiết bị đã khai báo trong tệp /etc/printcap chưa.

Gõ lệnh lpq xem công việc đã xếp hàng chờ in chưa. Nếu có, thiết bị có thể đang bận, hoặc máy in hỏng hóc hoặc bị lỗi. Nếu bị lỗi, bạn kiểm tra tệp báo lỗi.

Bạn gõ lệnh trạng thái lpc để xem có phải máy in hỏng không và có thể dùng lệnh lpc up tênmáyin hoặc lpc restart tênmáyin để kích hoạt lại.

Nếu sau khi kiểm tra xong mà công việc in ấn vẫn không tiến hành, bạn rà soát lại bộ lọc nhập liệu đã khai báo xem có hiện diện đúng thư mục thích hợp và có đủ các quyền hạn cần thiết. Nếu đang chạy syslogd, bạn tìm đọc thông báo của lpd trong những tệp ghi sổ. Tại đây nếu bạn thấy máy bảo không thể execv têncủabộlọcnhậpliệu, thì gần như chắc chắn rằng đấy là vấn đề.

Một khả năng khác là máy in của bạn là máy PostScript nhưng bạn lại không chuyển PostScript về đấy. Hầu hết các máy in PostScript không hiểu dữ liệu phi-PostScript. Bạn nên cài đặt một bộ lọc nhập liệu chuyển văn bản thành PostScript. Cuối cùng, kiểm tra xem bộ lọc nhập liệu của bạn có thực sự phát sinh được xuất liệu hay không và thiết bị kết xuất không phải là /dev/null.

Máy in của tôi có vẻ như bị treo. Những hướng dẫn vừa kể không giải quyết được vấn đề. Có lẽ bạn thử kill daemon lpd rồi kích hoạt lại. Nếu vẫn không được, bạn phải khởi động lại Linux bằng shutdown- r now. Trước đó nhớ kiểm tra xem còn ai trên hệ thống hay không và lưu mọi tệp. Bạn cũng có thể dùng DOS hoặc Windows kiểm tra xem bản thân thiết bị vật lý có thực sự hoạt động hay không.

20.9 Lập cấu hình máy in theo Red Hat

Minh hoạ 20.1 Màn hình cấu hình máy in

Nếu đã cài đặt XFree86 theo Red Hat, bạn có thể dùng công cụ thiết lập cấu hình máy in như minh hoạ 20.1 nhằm thêm, bớt máy in, cũng như duy trì hoạt động của các thư mục và tệp spooler và /etc/printcap. Công cụ này nằm trong Control Panel. Bảng sau mô tả những đề mục của **menu** PrintTool:

*Bảng Các **menu** PrintTool*

Tên **menu**	Đề mục **menu**	Mô tả
PrintTool	Reload	Rà soát lại thư mục tìm tệp printcap
	About	Trình bày thông tin về PrintTool
	Quit	Thoát khỏi PrintTool
Lpd	Restart	Khởi hành lại lpd sau khi thay đổi
Tests	Print ASCII **test** page	Chuyển một trang in thử dạng văn bản ròng về máy in được chọn
	Print PostScript **test** page	Chuyển một trang in thử dạng PostScript về máy in được chọn
	Print ASCII directly to port	Chuyển một trang in thử trực tiếp đến thiết bị và không qua trung gian lpd

Help	General	Cung cấp thông tin trợ giúp về PrintTool
	Troubleshouting	Cung cấp trợ giúp về các vấn đề in ấn

Minh hoạ 20.2 Thêm một máy in mới - bước 1 Dẫn nhập

Xem "Sử dụng Samba"

Minh hoạ 20.3 Thêm một máy in mới - bước 2 Định tên và kiểu hàng đợi

Muốn thêm máy in mới, bạn bấm nút Add. Như ở minh hoạ 20.2, bạn phải khai báo đó là máy in tại chỗ, từ xa, hoặc SMB. Máy in tại chỗ kết nối với cổng song song hoặc nối tiếp máy bạn. Máy in từ xa kết nối với mạng. Máy in LAN Manager kết nối với hệ thống khác qua trung gian giao thức Server Message Block (SMB-Samba), một hệ thống của Microsoft Windows.

Muốn chỉnh sửa cấu hình máy in có sẵn, chọn mục ghi rồi bấm nút Edit. Cả hai thao tác vừa kể sẽ hiển thị hộp thoại như minh hoạ 20.3 để bạn nhập giá trị vào mỗi trường. Bảng sau mô tả từng trường:

Trường	Mô tả
Names	Tên của máy in và hàng chờ in. Bạn có thể khai nhiều tên khác nhau và dùng ký hiệu | để phân cách.
Spool Directory	Thư mục spool dành cho máy in hữu quan, chẳng hạn như /var/spool.lpd/myprinter.
File Limit	Kích cỡ tối đa của tài liệu in (tính bằng KB). Giá trị 0 là vô giới hạn.
Printer Device	Kết nối vật lý của máy in, chẳng hạn như lp0
Input Filter	Gõ vào tên tệp và đường dẫn đầy đủ của bộ lọc bạn chọn. Nếu cần lập cấu hình máy in, bạn bấm nút Select.
Suppress Headers	Đánh dấu vào ô này nếu bạn không muốn mỗi trang in đều mang tiêu đề
Remote Host	Khai báo tên của máy chủ từ xa có kết nối với máy in.
Remote Queue	Khai báo hàng chờ in của máy từ xa. Bạn gõ vào đường dẫn đầy đủ.

Muốn lập cấu hình bộ lọc, bạn bấm nút Select, máy sẽ hiển thị hộp thoại . Bảng sau mô tả các trường của hộp thoại Configure Filter.

Bảng Trường của bộ lọc

Tên trường	Mô tả
Printer Type	Loại máy in cho bộ lọc này
Driver Description	Mô tả máy in được chọn
Resolution	Độ phân giải cho máy in
Paper Size	Kích cỡ giấy máy in
Color Depth	Các đặc điểm màu sắc
Printing Options	Gửi tín hiệu EOF (cuối tệp) lệnh cho máy in nhả giấy ra. Fix Stair-Stepping Text giải quyết hiệu ứng bậc thang. Fast Text Printing cho phép máy in phi-PostScript in nhanh hơn.
Margins	Khai báo lề
Extra GS Options	Khai báo tuỳ chọn ghostscript phụ trợ cho máy in

Sau khi thêm bớt mục ghi máy in, bạn nên kích hoạt lại daemon lpd. Trên RHS Linux Print System Manager, bạn chọn **menu** lpd rồi bấm vào mục lpd.

Chương 21. Sử dụng X Window

Nếu bạn đã từng quen với GUI như Microsoft Windows hoặc Macintosh, bạn cũng sẽ quen thuộc với X Window, X Window trình bày cho người sử dụng thấy nhiều cửa sổ, mỗi cửa sổ như thế lại hiển thị xuất liệu của một ứng dụng X Window, được gọi là client. Client này có thể đang chạy trên PC khác, hoặc trạm làm việc khác của mạng hệ thống.

Bạn di chuyển trên X Window nhanh hay chậm tuỳ thuộc vào trình quản lý cửa sổ. Hầu hết các cửa đều sử dụng một con chạy (pointer) màn hình gọi là cursor (con chạy, con nháy) để bạn nhận biết mình đang dừng ở đâu. Cursor có nhiều hình dáng khác nhau, tuỳ vào lúc ấy bạn đang làm gì và sử dụng chương trình quản lý cửa sổ nào.

Điều hướng X Window

Sử dụng Windows Manager.

Chạy các ứng dụng X với Red Hat.

21.1. Điều hướng X Window

Cũng như hầu hết các GUI khác, X Window hỗ trợ nhập dữ liệu từ bàn phím và thiết bị trỏ, thông thường đó là con chuột. Muốn nhập liệu, cửa sổ phải ở chế độ hoạt động. Cửa sổ hoạt động thường có hình dáng khác cửa sổ không hoạt động chẳng hạn như bờ viền được soi sáng, được tô màu.

Mỗi trình quản lý cửa sổ kích hoạt cửa sổ theo cách của mình. Với trình này, người sử dụng chỉ cần đưa con chạy vào là cửa sổ tự động kích hoạt, trong khi với trình khác, bạn phải bấm chuột vào đấy, chẳng hạn như trình quản lý cửa sổ của Microsoft Windows.

21.1.1 Sử dụng menu

Nhiều GUI và công cụ PC hiện nay sử dụng **menu** kéo xuống (Drop-down menu) hoặc bật ra (pop-up). Hầu hết các trình quản lý cửa sổ X Window không có thanh **menu** chính chạy ngang phía trên màn hình, nhưng lại dùng **menu** trôi nổi (floating). Trình này sẽ xuất hiện mỗi khi bạn bấm chuột vào khoảng trống trên màn hình. Tiếp theo bạn bấm nút chuột và giữ như thế để kéo rê cursor đi khắp những tuỳ chọn của **menu**. Khi bạn bấm tiếp nút chuột và giữ thế nào để kéo rê cursor đi khắp những tuỳ chọn của **menu**. Khi nào đến tuỳ chọn ưng ý, bạn thả nút chuột ra, giống như thao tác trên máy Macintosh.

21.1.2 Sử dụng terminal ảo với X Window

X server của bạn chạy trên một terminal ảo do Linux gán và đó là terminal ảo thứ bảy (cho RedHat). Nếu đang ở terminal ký tự, bạn bấm <Ctrl + Alt +F7> là đến terminal ảo này. Khi chạy X Window, bạn có thể đến các terminal khác bằng bộ phím <Ctrl-Alt-Fx>, với x là số thứ tự của terminal ảo mà bạn muốn đến. Ngoài ra X Window còn tạo điều kiện cho bạn kích hoạt chương trình phỏng tạo terminal ký tự gọi là *phiên xterm.*

Ghi chú: Nếu X server đang chạy, bạn phải dùng bộ phím <Ctrl-Alt-Fx> để đến với terminal ảo. Sau đó bạn vẫn có thể dùng <Alt-Fx> để di chuyển giữa các terminal ảo.

21.2 Sử dụng Windows Manager

X Window không bắt buộc phải sử dụng một trình quản lý cửa sổ đặc trưng nào. Hình dáng của X Window được để cho người sử dụng toàn quyền quyết định. Hầu như tất cả mọi "tính nết" của GUI đều tuỳ thuộc vào sự điều khiển của bạn. Trên tinh thần này, Linux không chỉ cung cấp cho bạn một trình duy nhất quản lý cửa sổ cho X Window, mặc dù việc cài đặt mặc định Red Hat và Slackware sẽ dùng fvwm làm trình mặc định (RedHat5.x), còn RedHat 7.x mà bạn đang sử dụng sẽ dùng sawish, bảng 21.1 liệt kê vài trình quản lý cửa sổ sẵn có cho Linux.

Bảng 21.1: Vài trình quản lý cửa sổ sẵn có cho Linux

Tên	Mô tả
twm	Trình quản lý cửa sổ Tom
fvwm	Trình quản lý cửa sổ ảo cho X11
fvwm95	Trình quản lý cửa sổ ảo cho X11 trông giống như Windows 95
mwm	Trình quản lý cửa sổ theo tiết tố (motif) (windowsMaker)
olwm	Trình quản lý cửa sổ Openlook, dựa theo Open Look của sun
olvm	Trình quản lý cửa sổ ảo Openlook
enlightenment	Một trình quản lý cửa sổ rát đẹp và phổ biến
sawfish	Trình quản lý cửa sổ mới của RedHat.

Minh hoạ 21.1: Màn hình "Quản lý cửa sổ"

Trong phần này bạn xét qua các trình quản lý cửa sổ thông thường:

21.2.1 twm

Trình quản lý cửa sổ twm cho X Window có những yếu tố như: thanh tiêu đề, cửa sổ có hình dáng riêng, nhiều cách quản lý biểu tượng, chức năng macro do user quy định, bấm chuột vào vùng văn bản để nhập liệu, đặt trọng tâm bằng con chạy, cùng với một số các chọn lựa khác liên quan đến bàn phím và chuột do user quy định.

Thông thường thì twm sẽ do startup script hoặc trình quản lý phiên làm việc của user kích hoạt. Nếu kích hoạt từ xdm hoặc xinit mà không có trình quản lí phiên làm việc, thường twm sẽ chạy ở mặt tiền với tư cách là client sau cùng. Do đó nếu chạy theo cách này và thoát khỏi twm, phiên làm việc sẽ xem là chấm dứt (đăng xuất). Theo mặc định, các cửa sổ ứng dụng đều có khung, gồm thanh tiêu đề ở trên cùng và đường viền quanh cửa sổ. Thanh tiêu đề mang tên cửa sổ sẽ sáng lên khi nhận nhập liệu từ bàn phím. Ngoài ra còn có các hộp chức năng gọi là nút tiêu đề (Title button) hai bên trái phải của thanh tiêu đề. Khi bạn bấm nút Button 1 của chuột (thường đó là nút trái con chuột, trừ khi bạn đã thay đổi tuỳ chọn bằng xmodmap) vào nút tiêu đề, chức năng hữu quan sẽ vận hành. Theo giao diện mặc định, cửa sổ sẽ thu nhỏ lại và biến thành biểu tượng (biểu tượng) khi bạn bấm chuột vào nút tiêu đề bên trái. Ngược lại, biểu tượng sẽ phình to thành cửa sổ khi bạn bấm vào biểu tượng, hoặc vào mục tương ứng trên trình quản lý biểu tượng.

Các cửa sổ sẽ thay đổi kích thước khi bạn bấm vào nút tiêu đền phải, rồi kéo rê con chạy qua bờ viền, cho đến khi cửa sổ có đúng kích thước mong muốn thì bạn nhả nút chuột ra. Tương tự như thế, bạn trỏ vào thanh tiêu đề để rồi bấm chuột, sau đó kéo rê toàn bộ cửa sổ đến vị trí mới. Còn nếu chỉ bấm vào thanh tiêu đề mà không kéo, cửa sổ chỉ to lên một chút.

Minh hoạ 21.2.: Cửa sổ dạng twm

Khi tạo ra cửa sổ mới, twm sẽ làm đúng theo kích cỡ và vị trí mà user yêu cầu. Nếu không twm sẽ hiển thị một khung tổng quát của cửa sổ, thanh tiêu đề và những đường kẻ chia cửa sổ thành ô vuông với chức năng theo dấu con chuột. Mỗi nút chuột sẽ ảnh hưởng khác nhau đến cửa sổ:

Bấm nút 1, cửa sổ sẽ nằm tại vị trí hiện hành và mang kích cỡ mặc định;

Bấm nút 2 (thường là nút giữa) và kéo rê khung ngoài sẽ thay đổi kích thước cửa sổ (như vừa nói bên trên) song vị trí cửa sổ không thay đổi;

Bấm nút 3 (thường là nút phải chuột) sẽ làm cho cửa sổ dài ra gần đến mép dưới màn hình, nhưng vị trí không thay đổi.

21.2.2 fvwm

Trình quản lý cửa sổ fvwm cho X11 xuất thân từ twm, nhưng được thiết kế lại để ít tốn bộ nhớ. fvwm làm cho cửa sổ trông giống ba chiều và cung cấp một desktop (nền bàn giấy làm việc) ảo. fvwm chỉ dùng khoảng từ 1/3 đến ½ lượng bộ nhớ do twm dùng, vì twm không hiệu quả khi lưu những lệnh liên quan đến chuột. Ngoài ra fvwm cũng bỏ bớt nhiều tuỳ chọn của twm.

Minh hoạ 21.3: Cửa sổ dạng fvwm

Vì trình quản lý cửa sổ ảo fvwm sử dụng ít tài nguyên bộ nhớ, màn hình ảo của XFree86 sẽ lúng túng khi chạy trình này. Những cửa sổ XFree86 khi xuất hiện trên màn hình ảo sẽ tuỳ thuộc rất nhiều vào bộ nhớ video.

Với desktop ảo của fvwm, những cửa sổ nào không xuất hiện trên mang hình sẽ không bị tác động của bộ nhớ video. Kích thước của desktop ảo được giới hạn ở 32.000x32.000 pixel. Bạn không thể bung một desktop ảo lớn gấp 5 lần kích cỡ của một mang hình thực tế.

Ghi chú: việc tiêu thụ tài nguyên bộ nhớ tuỳ thuộc vào số lượng cửa sổ được mở trên màn hình. Kích cỡ của từng cửa sổ không tác động.

Khi đang làm quen với fvwm, bạn nên vô hiệu hoá màn hình ảo của Xfree86 bằng cách chỉnh màn hình ảo bằng với kích cỡ màn hình thật. Đợi khi nào sử dụng fvwm quen tay bạn hãy chọn màn hình ảo Xfree86 hoạt động trở lại.

fvwm cung cấp nhiều desktop ảo và desktop có thể bằng hoặc lớn hơn màn hình thật. Lúc này màn hình chỉ là một phân cảnh (viewport) của desktop lớn hơn của màn hình thật.

Ghi chú: Thông thường màn hình cho phép bạn có một cái nhìn toàn cảnh (full view) của desktop, nên nhóm biên soạn giáo trình này mượn từ "phân cảnh" để diễn tả ý niệm "một phần của toàn cảnh" (viewport).

Bạn có thể mở nhiều desktop càng lúc. Mỗi dự án hoặc ứng dụng có riêng cho mình một desktop.

Bạn phải khai báo kích cỡ mỗi desktop ảo ngay vào lúc cấu hình X Server. Tuỳ vào các khai báo trong XF86Config, độ phân giải màn hình và việc có xác lập desktop ảo phải có cùng kích cỡ. Bạn không cần phải khai báo số lượng desktop riêng biệt, song giới hạn là gần 4 tỉ. Tất cả cửa sổ trên desktop hiện hành có thể được hiển thị trong trang nhớ, phiên bản thu nhỏ, hoặc ngay trên desktop hiện hành bằng **menu** popup. Cửa sổ sẽ được liệt kê cùng với hình học (geometry) của chúng. (Ở đây, từ "hình học" chỉ các toạ độ và độ phân giải của một cửa sổ chạy với trình quản lý cửa sổ X).

Cửa sổ dính (Sticky windows) là loại cửa sổ không bị desktop khống chế và luôn hiện diện trên màn hình cho dù desktop có di chuyển. Việc này tiện lợi khi bạn dùng tiện ích đồng hồ hoặc xbiffs (ứng dụng này báo bạn biết khi có thư đến).

Hình học cửa sổ (Windows geometry) luôn được khai báo tương ứng với phân cảnh hiện hành, nghĩa là xterm-geometry +0+0 luôn xuất hiện ở góc cao bên trái màn hình. Bạn có thể khai báo những hình học làm cho cửa sổ xuất hiện trên desktop ảo, nhưng không xuất hiện trên màn hình. Thí dụ nếu màn hình nhìn thấy được là 1.000x1.000 pixel, kích thước desktop là three –by- three (3x3) (có 9 desktop ảo) và phân cảnh hiện hành đang ở góc trên bên trái desktop. Nếu bạn chọn xterm-geometry +1000+1000 sẽ đặt cửa sổ nằm vừa sát bên ngoài màn hình, phía góc dưới bên phải. Muốn nhìn cửa sổ này, bạn di chuyển con chạy chuột đến góc dưới phải màn hình rồi chờ nó cuốn vào. Bạn chỉ có thể ánh xạ một cửa sổ vào desktop hoạt động chứ không thể vào desktop thụ động.

Hình học xterm-geometry –5-5 thường đặt góc phải dưới của cửa sổ cách góc phải dưới của phần nhìn thấy được của màn hình khoảng 5 pixel. Chỉ một vài ứng dụng chấp nhận hình học cửa sổ với số âm.

21.2.3 fvwm 95

Trình quản lý cửa sổ fvwm95 cho X11 xuất xứ từ fvwm2.x. Những người triển khai muốn phỏng tạo các đặc điểm chủ yếu của GUI một hệ điều hành nổi tiếng dễ sử dụng hơn cho các user trong môi trường UNIX, đồng thời tránh tăng kích thước mã GUI gọn sạch của fvwm. Mời bạn đến địa chỉ sau đây để cập nhật thông tin: http://mitac11.uia.ac.be/html-**test**/fvwm95.html.

21.2.4 Olwm

Trình quản lý cửa sổ olwn X Window sử dụng một số yếu tố của GUI Openlook. Là trình quản lý cửa sổ chuẩn cho Open Windows của Sun nhưng chương trình này vẫn hoạt động tốt với X11, kể cả XFree86. Điều kiện hoạt động là server phải có trình OPENLOOK cùng với các phông con chạy.

21.2.5 Enlightenment

Enlightenment là trình quản lý cửa sổ rất phổ biến và dễ dùng vì chạy nhanh và ổn định. Mặc dù những phiên bản cũ lấy nền fvwm, phiên bản gần đây được viết lại hoàn toàn. Mời bạn tham khảo chi tiết của Enlightenment tại www.rasterman.com vì khuôn khổ giáo trình này không thể phân tích hết.

Minh hoạ 21.4: Cấu hình Sawfish.

21.2.6 Sawfish

Đây là trình quản lý windows mặc định của RedHat. Đó là một trình quản lý cửa sổ viết bằng ngôn ngữ có thể mở rộng Lisp. Mục tiêu của sawfish là làm thế nào để mọi vùng của việc quản lý cửa sổ đều có thể được chỉnh sửa và nhanh chóng hơn các trình quản lý cửa sổ đang hiện hành.

Các chức năng cao cấp của việc quản lý cửa sổ được thiết lập trong Lisp chuẩn bị cho việc mở rộng hay định nghĩa lại.

21.3. Chạy các ứng dụng X với RED HAT

Phiên bản thương phẩm của REDHAT chứa một bản X server (bản quyền dành cho một người sử dụng) gọi là Metro-X. Khi kích hoạt X theo Red Hat bằng lệnh startx&

Bạn sẽ trông thấy một màn hình rất giống Micrsoft Windows 98. Nút Start trình bày nhiều chương trình Linux hữu ích, cùng với lệnh thuộc hệ thống và các tiến trình. Cũng như đối với hầu hết những bản X khác, bạn có thể hiển thị những thành phần vừa kể khi bấm chuột vào desktop.

Minh hoạ 21.5: Màn hình giao diện GNOME

Bảng 21.2 Mô tả từng đề mục.

*Bảng 21.2: Những đề mục của **menu** Start*

Tiết mục	Mô tả
New Shell	Cung cấp cho user một cửa sổ mới về shell lệnh, nghĩa là một cửa sổ xterm.
Applications	Ngõ truy cập các ứng dụng, chẳng hạn như pine (e-mail), Xpaint (đồ hoạ) và irc (chat).
Utilities	Cung cấp các những tiện ích như máy tính bỏ túi, lịch bàn, xterm màu và ngõ vào các trang **man**
Multimedia	Cung cấp đầu máy CD audio và mixer audio
Games	Cung cấp các trò chơi như tảtí hoặc Doom
Hosts	Cung cấp ngõ truy cập các hosts khác trên mạng nội bộ hoặc Internet
System	Ngõ truy cập các tiện ích vào hệ thống với tư cách là root và cũng để quản lý các cửa sổ tốt hơn.
Utilities	Cũng để quản lý các cửa sổ tốt hơn.

Windows Operations	Cung cấp những đề mục để đóng, tắt và di chuyển cửa sổ quanh desktop.
Preferences	Để căn chỉnh X desktop.
Screensaver	Tuỳ chọn hình ảnh cho chương trình làm mát màn hình.
Lock Screen	Trình bày một loạt kiểu mẫu màn hình để bạn chọn, màn hình sẽ khoá. Muốn mở khoá, bạn nhập mật khẩu.
About Fvwm	Hiển thị hộp thoại chứa thông tin về fvwm.
Help Fvwm	Hiển thị trình duyệt HTML có phần trợ giúp fvwm
Exit Fvwm	Giúp bạn thoát khỏi X và trở về terminal nào đã kích hoạt X hoặc khởi hành lại X server.

21.3.1 nxterm

Nếu bạn chọn tiết mục New Shell trên màn hình đơn, máy sẽ kích hoạt một phiên làm việc nxterm, mà Red Hat Linux gọi là phiên nxterm. nxterm là ứng dụng phổ biến của X Window nhằm phỏng tạo một terminal video chẳng hạn như DEC vt 100. Khi kích hoạt phiên làm việc nxterm, bạn có thể chạy mọi chương trình dùng lệnh hoặc thi hành mọi lệnh Linux như khi bạn thực hiện trên terminal ảo của Linux.

Minh hoạ 21.3: Trình bày một phiên xterm

*Minh hoạ 21.6: Hệ **menu** của giao diện GNOME*

Minh hoạ 21.7: Cửa sổ nxterm.

21.3.2 Screen Capture (KSnapshot)

Screen Capture là một chương trình chụp màn hình của Red Hat.

Minh hoạ 21.4: Hộp thoại của Screem Capture.

Ghi chú: Shareware là những chương trình bạn có thể tải về dùng miễn phí, song sau một thời gian nếu quyết định giữ lại dùng luôn, bạn phải trả một khoản phí cho tác giả. Thường thì khoản phí này không quá cao.

Bảng 21.3: Các nút lệnh Screen Capture

Nút	Mô tả
Filename	Tên tệp ảnh sẽ lưu
Deley	Thời gian trễ để chụp ảnh

Grab	Bắt đầu chụp ảnh
Save	Hình ảnh hiện hành đang bị chụp sẽ được lưu vào tệp đĩa. bạn có thể chọn từ những loại sau: GIF, JPEG, TIFF, PostScript, PBM (thô), PBM (ASCII), bitmap X11, XPM, BMP, IRIS RGB, PM, tệp mành, targa (24 bit) và Fist

Nút Grab ở góc trên bên phải hộp thoại giúp bạn chụp bất kỳ của cửa sổ nào của desktop. Bấm chuột vào đấy, con chạy sẽ chuyển thành chữ thập. Nếu chọn mục check box, máy chỉ chụp cửa sổ nào được chữ thập con chạy bấm lên.

Minh hoạ 21.8: Cửa sổ Ksnap

21.4. Chạy các ứng dụng X khác

Trên Internet có rất nhiều ứng dụng cho X Window

21.4.1. xterm

xterm cũng như nxterm của Red Hat và cũng không khác lắm GNOME-terminal. xterm là một ứng dụng X Window phỏng tạo một terminal video chẳng hạn như DEC vt100. Khi kích hoạt phiên làm việc xterm, bạn có thể chạy mọi chương trình dòng lệnh hoặc thi hành mọi lệnh Linux như bạn thực hiện trên terminal ảo của Linux. Minh hoạ 21.9 trình bày một phiên xterm.

Minh hoạ 21.9: Cửa sổ xterm.

xterm là chương trình phỏng tạo terminal cho hệ X Window, chuyên cung cấp terminal tương thích với DEC vt102 và Tektronik 4014 cho những chương trình không đủ khả năng sử dụng hệ cửa sổ một cách trực tiếp. Nếu hệ điều hành nền chấp nhận các khả năng xác định lại kích thước terminal, xterm sẽ dùng những tiện ích ấy để báo cho các chương trình đang chạy trong cửa sổ ấy biết khi nào việc tái xác định xảy ra.

Các terminal vt102 và Tektronik có cửa sổ riêng, do đó bạn có thể vừa soạn thảo văn bản ở cửa sổ này vừa xem xét đồ hoạ trong cửa sổ khác. Để duy trì tỷ lệ co chính xác, đồ hoạ Tektronik được giới hạn trong các hộp lớn nhất với tỷ lệ Tektronik 4014 vừa khít cửa sổ. Hộp này nằm tại vùng phía cao bên trái cửa sổ. Tỷ lệ co là chiều cao của màn hình chia cho chiều rộng, tất cả tính bằng đơn vị pixel.

Mặc dù máy có thể hiển thị cùng lúc các cửa sổ văn bản và đồ hoạ, song cửa sổ nào chứa con chạy văn bản được xem là cửa sổ "hoạt động", tức là đặt trong tình trạng sẵn sàng nhận nhập liệu từ bàn phím và xuất liệu từ terminal. Bạn có thể

chọn cửa sổ hoạt động bằng cách bấm escape nhiều lần. **Menu** tuỳ chọn vt nằm ở cửa sổ vt102 và **menu** tuỳ chọn Tek ở cửa sổ 4014.

21.4.1.1 Phỏng tạo

Những mục ghi $TERMCAP làm việc với xterm bào gồm xterm, vt 102, vt100 và ANI. Biến môi trường $TERMCAP khai báo loại terminal mà máy bạn đang phỏng tạo. xterm tự động duyệt tìm tệp cơ sở dữ liệu termcap theo thứ tự kể trên, sau đó lập các biến môi trường TERM và $TERCAP .

Ghi chú: Mời bạn xem các trang **man** của termcap để có thêm thông tin về các mục ghi termcap và những trình tự thoát (escape sequence).

Bạn có thể chỉnh sửa nhiều đặc điểm xterm bằng một bộ những trình tự thoát khác với trình tự thoát chuẩn vt102.

Phỏng tạo Tektronik 4014 hỗ trợ bốn kích cỡ phông và năm loại line khác nhau. Nếu bạn phát đi trình tự thoát COPY của Tektronik, tất cả những lệnh điều khiển văn bản và đồ hoạ Tektronik sẽ được xterm ghi nhận bên trong và có thể ghi vào tệp.

21.4.1.2 Những đặc điểm khác của xterm

xterm tự động biến thành cursor (con nháy) văn bản khi con chạy chạy vào vùng cửa sổ. Trong trường hợp đó là cửa sổ trọng tâm, thì cursor văn bản đương nhiên hiện ra cho dù con chạy đang ở đâu.

Chế độ vt102 có nhiều trình tự thoát để kích thước với vùng hiển thị của cửa sổ. Khi được kích hoạt, màn hình hiện hành được sao lưu và thay bằng màn hình thay thế.

Những dòng cuốn khỏi phạm **vi** trông thấy ở phía trên màn hình sẽ bị vô hiệu hoá cho đến khi màn hình bình thường được phục hồi. Mục ghi termcap của xterm cho phép trình soạn thảo văn bản **vi** chuyển sang màn hình thay thế để làm việc và sau khi thoát khỏi xong sẽ phục hồi màn hình.

Chế độ vt102 và Tektronik có những trình tự thoát để thay đổi tên cửa sổ.

21.4.1.3 Dùng con chuột với xterm

Khi cửa sổ vt102 được tạo ra, xterm cho phép bạn chọn văn bản và sao chép bên trong cửa sổ hoặc sang cửa sổ khác.

Khi bạn dùng các nút con chạy không kèm modifier (trợ phím) hoặc khi dùng kèm với <Shift>, chức năng chọn sẽ được gọi (Điều này cũng có thể hiểu là “Khi vừa dùng nút con chạy vừa bấm Shift, coi như di chuyển con chạy đến đâu thì bạn đã

chọn đến đó"). Bạn có thể gán và thay đổi chức năng của phím và nút bằng cơ sở dữ liệu tài nguyên.

Button 1 của chuột (thường là nút trái) sẽ sao chép văn bản vào vùng đệm cắt. Bạn đưa cursor đến điểm đầu vùng văn bản được chọn, bấm và giữ nút chuột trong khi rê cursor đến điểm cuối cùng của văn bản được chọn, rồi thả nút chuột ra. Lúc này vùng chọn sẽ được soi sáng và lưu vào vùng đệm cắt. Khi bạn thả nút chuột ra, vùng văn bản được chọn trở thành sự chọn lựa chính.

Bấm đôi chuột sẽ được chọn nguyên chữ, bấm ba chọn dòng, bấm tư trở về mức ký tự.

Button 2 của chuột (thường là nút giữa) dán nội dung văn bản của vùng chọn chính. Nếu không, nội dung văn bản được chèn sẽ là nội dung vùng đệm cắt, được coi như là nhập liệu từ bàn phím chèn vào.

Khi cắt dán nhiều mẩu văn bản mà không viết ra dòng mới nào, bạn có thể chép văn bản từ nhiều chỗ của nhiều cửa sổ khác nhau, sau đó hình thành một lệnh cho shell. Chẳng hạn bạn lấy xuất liệu của một chương trình, rồi chèn vào trình soạn thảo văn bản. Vì có nhiều ứng dụng dùng chung vùng đệm cắt, bạn nên xem nó như một tệp mà bạn đã biết trước nội dung. Thiết bị phỏng tạo terminal và chương trình văn bản khác sẽ xem vùng đệm cắt như là một tệp văn bản, nghĩa là vùng văn bản được giới hạn bởi những dòng mới gõ vào.

Đến đây bạn quan sát vùng cuốn bên trong cửa sổ đang hiển thị xterm. Vị trí và số lượng văn bản trong cửa sổ được vùng cuốn này trình bày sẽ tương đương với số lượng văn bản đang được lưu. Số lượng văn bản lưu càng tăng (đến giới hạn do hệ thống quy định) thì kích cỡ của vùng soi sáng càng giảm.

Nếu bạn bấm Button1 với con chạy trong vùng cuốn, máy sẽ di chuyển dòng tiếp theo đến đỉnh của cửa sổ hiển thị. Bấm Button2 bạn sẽ di chuyển phần hiển thị đến vị trí trong văn bản được lưu tương đương với vị trí của con chạy trong thanh cuốn. Bấm Button3 sẽ di chuyển dòng đầu tiên của cửa sổ hiển thị xuống đến vị trí con chạy.

Không giống như cửa sổ vt102, cửa sổ Tektronik không cho chép văn bản. Tuy nhiên cửa sổ Tektronik lại hỗ trợ chế độ Tektronik GIN. Ở chế độ này cursor hình mũi tên biến thành dấu thập. Nếu bấm bất kỳ phím nào, bạn sẽ phát đi phím ấy cùng với toạ độ hiện hành của cursor hình dấu thập. Bấm Button1, Button2, hoặc Button3 sẽ tuần tự trả về các ký tự 1. m và r. Nếu bấm một nút ký tự nào cùng với nút <Shift>, coi như bạn đã phát đi hình thức viết hoa của ký tự ấy. Máy lập một high bit của ký tự để phân biệt con chạy với phím ký tự ấy.

21.4.. xcale

Minh hoạ 21.7. trình bày xcale, một công cụ desktop phỏng tạo máy tính khoa học bỏ túi TI-30 hoặc HP-10C. Bạn có thể thao tác bằng Button 1 hoặc khi bằng bàn phím.

Minh hoạ 21.10: Trình xcale

Nhiều thao tác tính toán phổ biến hiện nay có bộ tăng tốc bàn phím. Muốn thoát, bạn dùng Button 3 bấm phím AC của máy tính TI hoặc nút OFF của máy HP. Chế độ TI sử dụng các phím số, phím +/-, các phím +-*/ và = giống như trên các máy tính thường.

Chi chú: Các toán tử tuân theo quy định ưu tiên. Do đó nếu bạn gõ 3+4*5, kết quả sẽ là 23 chứ không phải là 35. Vì vậy muốn vượt qua chế độ ưu tiên, bạn phải dùng ngoặc đơn. Thí dụ (1+2+3)*(4+5+6) = sẽ cho kết quả là 90 thức là (6*9).

Bạn có thể chọn và dán dãy số trong phần hiện thị kết quả của máy tính vào văn bản. Bảng 21.4 liệt kê những chức năng khác nhau của phỏng tạo TI.

Bảng 21.4. Phỏng tạo TI

Phím/Chức năng	Mô tả
1/x	Thay thế con số trong khung hiển thị bằng số nghịch đảo. (Từ đây về sau, ý niệm "con số trong khung hiển thị của máy tính" sẽ được gọi tắt là "con số".
x^2	Bình phương con số.
SQR	Lấy căn bậc hai của con số.
CE/C	Khi bấm chuột một lần, sẽ xoá con số nhưng không xoá trạng thái của máy, cho phép bạn gõ vào số khác nếu trước đó bạn gõ nhầm số. Bấm chuột hai lần sẽ xoá luôn trạng thái máy tính. (Bấm AC sẽ xoá con số, trạng thái và bộ nhớ). Bấm CE/C bằng Button3 sẽ tắt máy tính bỏ túi và thoát khỏi xcale.
INV	Đảo ngược chức năng. Mời bạn xem sự mô tả của từng phím chức năng để biết chức năng đảo của chúng.
sin	Tính sin của con số theo chế độ DRG hiện hành. Nếu đảo lại, máy sẽ tính arcsine.
cos	Tính cos. Nếu đảo với phím INV sẽ tính arccosin.
tan	Tính tang, nếu đảo sẽ tính arctang

DRG	Chuyển chế độ DRG, căn cứ theo DEG, RAD, hoặc GRAD hiển thị ở phía dưới máy tính. Khi ở chế độ DEG, các con số được tính theo đơn vị "độ", RAD tính theo radian và GRAD tính theo grad. Khi đảo ngược, phím DRG sẽ chuyển đổi độ thành radian, thành grad và ngược lại. Thí dụ khi máy tính ở DEG, bạn gõ 45 INV DRG, xcale sẽ hiển thị .2285398, bằng với 45 độ chuyển sang radian.
e	Hằng e, bằng 2.22182818.
EE	Dùng để nhập các số mũ. Thí dụ muốn nhập –2.3E-4, bạn gõ 1.3+/-EE4+/-
log	Tính toán logarit (cơ số 10) của con số trong vùng nhập. Khi đảo ngược, sẽ cho kết quả là 10.0 luỹ thừa con số đã chọn. Thí dụ bạn nhập 3 INV log, kết quả sẽ là 10 mũ 3 = 1000.
1n	Tính toán log (cơ số e) của con số. Khi đảo ngược sẽ nâng e lên luỹ thừa với số mũ là con số đã cho. Thí dụ bạn gõ e 1n sẽ có kết quả là 1.
y^x	Nâng con số phía bên trái lên luỹ thừa của con số bên phải. thí dụ gõ 2ỹ^3-, kết quả sẽ là 8, tương ứng với 2^3
PI	Hằng = (3.14159222) (thường ký hiệu Π)
x!	Tính toán giai thừa của con số. Con số phải là số nguyên từ 0 đến 500, tuỳ thuộc và thư viện toán của bạn, nhưng có khả năng đã tràn trước đó.
(	Ngoặc đơn trái
)	Ngoặc đơn phải
/	Chia
*	Nhân
-	Trừ
+	Cộng
=	Thực hành tính toán
STO	Chép con số từ khung hiển thị vào bộ nhớ
RCL	Chép con số từ bộ nhớ vào khung hiển thị

SUM	Cộng con số hiển thị dồn vào con số trong bộ nhớ
EXC	Hoán chuyển con số của bộ nhớ và khung hiển thị
+/-	Chuyển toán tử
.	Dấu chấm thập phân

Ở chế độ RPN hoặc chế độ HP, các phím số, phím CHS (đổi toán tử), các phím +-*/ và ENTER hoạt động đúng theo ý nghĩa bạn thường dùng. Nhiều phím khác giống như ở chế độ TI. Những điểm khác biệt được liệt kê ở Bảng 21.5.

Bảng 21.5. Phỏng tạo HP

Phím/Chức năng	Mô tả
<	Phím backspace dùng khi bạn lỡ gõ sai con số. Phím sẽ xoá ký số khỏi phần hiển thị. Nếu đảo ngược backspace, thanh ghi x sẽ bị xoá.
ON	Xoá phần hiển thị, trạng thái và bộ nhớ. Bấm ON bằng Button3 sẽ tắt máy tính toán và thoát khỏi xcale.
INV	Đảo ngược ý nghĩa của các phím chức năng. Đây là phím f trên máy tính toán HP, nhưng xcale không hiển thị chú thích về công dụng khác nhau của phím. Mời các bạn xem các phím chức năng.
10^x	Luỹ thừa cơ số 10 với mũ là con số ở đỉnh stack. Khi đảo ngược, máy tính toán log (cơ số e) của con số đó
e^x	Luỹ thừa cơ số e với số mũ là con số ở đỉnh của stack. Khi đảo ngược, máy sẽ tính toán log (cơ số e) của con số đó.
STO	Lưu con số ở đỉnh stack vào bộ nhớ. Bộ nhớ có 10 vị trí. Muốn xác định vị trí nào, bạn gõ một ký số theo sao phím STO
RCL	Đưa con số ở vị trí đã chọn của bộ nhớ vào stack.
SUM	Cộng con số ở đỉnh stack dồn vào con số của vị trí đã chọn trong bộ nhớ.
x:y	Hoán chuyển những con số ở hai vị trí đỉnh stack, đó là hai thanh ghi x và y.
Rv	Lập úp stack xuống. Khi đảo ngược, máy sẽ lật đứng stack lên.

Phím trắng	Những phím này có chức năng lập trình đối với HP-10C. xcale chưa sao y các chức năng này.

21.4.3 KSpread

Chương trình KSpread, như trình bày ở Minh hoạ 21.8, là phần mềm bảng biểu dạng spreadsheet chạy với X Window. KSpread phải được cài đặt với bộ KDE (nằm trong bộ KOfice- tương tự bộ MS Ofice của Microsoft và bộ StarOfice của Sun) và bạn phải dùng terminal X Window. Cẩm nang tham khảo (Reference Manual) KSpread là tài liệu chi tiết của chương trình này.

KSpread hỗ trợ nhiều đặc điểm bảng biểu chuẩn, bao gồm:

Ghi vào ô và chỉnh sửa

Phiếu làm việc 2202 cột với số lượng dòng vô tận.

Đọc và ghi vào tệp

Mã hoá tệp

Tham chiếu ô tuyệt đối và tương đối

Dữ liệu số và nhãn (chuỗi chữ) trong ô

Bằng lề bên trái hoặc phải cho nhãn

Chèn và huỷ dòng và cột

Giấu và thôi giấu dòng và cột

Tên khối

Tính toán lại bằng thủ công hay tự động

Toán tử số (+-*/^%)

Toán tử quan hệ (< <=>>= +!=)

Toán tử logic (hay Boolean) như & |

Tham chiếu chức năng

Đồ thị (XY, thanh, thanh chồng, hình bánh và đường nét)

Toán tử ma trận (chuyển vị, nhân, cộng, trừ và đảo ngược)

Định vị con nháy bằng chuột

Chọn tiết mục **menu** bằng chuột

Tham chiếu những chương trình bên ngoài, được gọi là chức năng tham chiếu ngoài

Cấu trúc và hoạt động của bảng biểu dạng spreadsheet tương tự (nhưng không giống) các bảng biểu thông dụng như Lotus 1-2-3 và những bản nhái Lotus hay trông cũng không khác Excel lắm. Như những spreadsheet khác, vùng làm việc (diện tích thao tác) ở đây được chia thành ô bởi dãy và cột. Mỗi ô có thể chứa một số, nhãn, hoặc công thức để cho ra nhãn hoặc số.

Minh hoạ 21.11: Trình bảng tính KSpread

Bạn có thể kích hoạt KSpread kèm với tên tệp hay không cũng được. Có tên tệp, KSpread sẽ tìm đọc hiển thị nội dung tệp tại vùng làm việc. Nếu không tìm ra tệp hoặc nếu bạn không gõ tên tệp tại dòng lệnh, KSpread sẽ khởi hành bằng vùng làm việc trống.

Muốn được hướng dẫn thực hành, bạn kích hoạt một trong những tệp biểu diễn (demo) như demo, demo_math, hoặc demo_matrix, hoặc tham khảo tệp Sample_Run tại thư mục doc.

21.4.4 xlock

Tác giả của xlock là Patrick J Naughton. Chương trình xlock khoá màn hình X Window tại chỗ user nhập mật khẩu từ bàn phím. Trong khi xlock đang chạy, máy sẽ từ chối mọi kết nối mới vào server, trình làm mát màn hình và cursor chuột không hoạt động, đồng thời màn hình được xoá sạch, sau đó lại hiển thị một hình vẽ luân phiên thay đổi. Lúc này nếu bấm phím hoặc bấm chuột, user sẽ được mời nhập mật khẩu của user nào đã kích hoạt xlock.

Nếu mật khẩu đúng, màn hình được mở khoá và X Window phục hồi. Trong khi đang gõ mật khẩu, hai bộ phím <Ctrl- Shift-u> và <Ctrl- Shift-h> sẽ giữ vài trò chấm dứt và xoá lệnh. Muốn trở về màn hình bị khoá, bạn bấm vào biểu tượng nhỏ trên hình vẽ đang thay đổi.

21.5 Chơi game với Linux

Minh hoạ 21.12: Trò chơi Freecell trong Linux

Thêm một lý do nữa biện minh cho việc chạy XFree86 với Linux là có khá nhiều game cho Linux. Từ các trò chơi Solitaire, Tic-Tac-Toe, đến FreeCell...

21.6 Tổ chức công việc cho cá nhân:

Bạn cũng có các công cụ phục vụ cho cá nhân tương tự MS Outlook như KOrganizer (task management), Address Book,...

Hay có cả các công cụ chơi nhạc:

Minh hoạ 21.13: Một công cụ Multimedia trong Linux

Minh hoạ 21.14: Trình quản lý công việc cá nhân Korganizer

Giáo trình

HỆ ĐIỀU HÀNH LINUX

Trung tâm TCCN&DN

MỤC LỤC

LỜI MỞ ĐẦU

Môn học này sẽ giúp các bạn làm quen với **hệ điều hành Linux** – hệ điều hành mô phỏng Unix cho các máy tính cá nhân – Linux hỗ trợ đầy đủ khả năng đa nhiệm, đa người dùng, hệ thống giao diện *X Window*, mạng *TCP/IP* và rất nhiều tính năng khác.

Linux là hệ điều hành thu hút được nhiều sự chú ý nhất trong vòng vài năm trở lại đây. Ngay từ khi xuất hiện, nó đã được lan rộng một cách nhanh chóng và biết tới như một hệ điều hành Unix – với mã nguồn mở. Thật ngạc nhiên, sự thành công của Linux có được nhờ sự làm lại một trong những hệ điều hành lâu đời nhất và hiện đang được sử dụng rộng rãi – hệ điều hành Unix. Linux bao gồm cả các công nghệ cũ và mới.

Nhìn từ góc độ kỹ thuật, Linux chỉ là một nhân hệ điều hành, nó hỗ trợ đầy đủ các phục vụ cơ bản về quản lý tiến trình, bộ nhớ ảo, quản lý file và vào ra thiết bị. Nói cách khác, bản thân Linux là phần thấp nhất của hệ điều hành.

Tuy nhiên, phần lớn người dùng đều coi "*Linux*" như một hệ thống hoàn chỉnh gồm nhân hệ điều hành kèm theo các trình ứng dụng khác: một môi trường làm việc và phát triển đầy đủ bao gồm trình dịch, các hệ soạn thảo, giao diện đồ họa, xử lý văn bản, … Cho tới phiên bản *Linux RedHat 6.1*, Linux đã trở thành một hệ điều hành đầy đủ cho thương mại, giáo dục hoặc người dùng cá nhân.

Điều làm cho Linux trở nên khác biệt là việc viết mã tự do của Unix. Việc này do một nhóm phát triển tự nguyên trên mạng Internet, họ trao đổi mã nguồn, phát hiện và sửa lỗi trong một môi trường mở.

Linux có thể được cài đặt trên một máy tính cá nhân và trở thành một trạm làm việc với đầy đủ sức mạnh của Unix. Linux cũng có thể được sử dụng với mục đích thương mại trên một mạng máy tính như một môi trường tính toán và truyền tin. Trong các trường đại học, Linux được sử dụng để giảng dạy về hệ điều hành và lập trình hệ điều hành. Và tất nhiên, Linux cũng có thể được sử dụng trên các máy tính cá nhân như các hệ điều hành khác.

CHƯƠNG I. GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH LINUX

I.1 Khái niệm Hệ điều hành

I.1.1 Khái niệm Hệ điều hành

Hệ điều hành là tập hợp các chương trình được tổ chức thành một hệ thống với nhiệm vụ đảm bảo tương tác giữa người dùng với máy tính, cung cấp các phương tiện và dịch vụ để điều phối việc thực hiện các chương trình, quản lí chặt chẽ các tài nguyên của máy, tổ chức khai thác chúng một cách thuận tiện và tối ưu.

I.1.2 Chức năng của Hệ điều hành

- Tổ chức giao tiếp giữa người sử dụng và hệ thống.
- Cung cấp bộ nhớ, các thiết bị ngoại vi, … cho chương trình và tổ chức thực hiện các chương trình đó.
- Tổ chức lưu trữ thông tin trên bộ nhớ ngoài, cung cấp các công cụ tìm kiếm và truy cập thông tin.
- Hỗ trợ phần mềm cho các thiết bị ngoại vi.
- Cung cấp các dịch vụ tiện ích hệ thống.

I.2 Lịch sử và các bản phân phối Hệ điều hành Linux

I.2.1 Lịch sử Hệ điều hành Linux

Linux bắt nguồn từ một hệ điều hành lớn hơn có tên là Unix. Unix là một trong những hệ điều hành được sử dụng rộng rãi nhất thế giới do tính ổn định và khả năng hỗ trợ của nó. Ban đầu hệ điều hành Unix đã được phát triển như một hệ điều hành đa nhiệm cho các máy mini và các máy lớn (*mainframe*) trong những năm 70. Cho tới nay nó đã được phát triển trở thành một hệ điều hành phổ dụng trên toàn thế giới, mặc dù với giao diện chưa thân thiện và chưa được chuẩn hóa hoàn toàn.

Linux là phiên bản Unix được cung cấp miễn phí, ban đầu được phát triển bởi **Linus Torvald** năm 1991 khi còn là một sinh viên của trường đại học Helssinki

Phần Lan. Hiện nay, Linus làm việc tại tập đoàn Transmeta và tiếp tục phát triển nhân hệ điều hành Linux (*Linux kernel*).

Khi Linus tung ra phiên bản miễn phí đầu tiên của Linux trên Internet, vô tình đã tạo ra một làn sóng phát triển phần mềm lớn nhất từ trước đến nay trên phạm vi toàn cầu. Hiện nay, Linux được phát triển và bảo trì bởi một nhóm hàng nghìn lập trình viên cộng tác chặt chẽ với nhau qua Internet. Nhiều công ty đã xuất hiện, cung cấp Linux dưới dạng gói phần mềm dễ cài đặt, hoặc cung cấp các máy tính đã cài đặt sẵn Linux.

Tháng 11 năm 1991, Linus đưa ra bản chính thức đầu tiên của Linux, phiên bản *0.02*. Ở phiên bản này, Linus đã có thể chạy *bash* và *gcc* (*trình dịch C GNU*) nhưng mới chỉ dừng lại ở đó. Hệ thống chưa có các hỗ trợ người dùng và tài liệu hướng dẫn. Các số hiệu phiên bản không ngừng gia tăng cùng với việc bổ sung thêm các tính năng mới.

Sau ba năm nhân Linux ra đời, đến ngày 14-3-1994, hệ điều hành *Linux phiên bản 1.0* được phổ biến, đây là phiên bản tương đối ổn định. Thành công lớn nhất của Linux 1.0 là nó đã hỗ trợ giao thức mạng *TCP/IP* chuẩn Unix, sánh với giao thức *socket BSD* – tương thích cho lập trình mạng. Trình điều khiển thiết bị đã được bổ sung để chạy IP trên một mạng Ethernet hoặc trên tuyến đơn hoặc qua modem. Hệ thống file trong *Linux 1.0* đã vượt xa hệ thống file của Minix thông thường, ngoài ra đã hỗ trợ điều khiển SCSI truy nhập đĩa tốc độ cao. Điều khiển bộ nhớ ảo đã được mở rộng để hỗ trợ điều khiển trang cho các *file swap* và ánh xạ bộ nhớ của file đặc quyền (*chỉ có một ánh xạ bộ nhớ chỉ đọc được thi hành trong Linux 1.0*)

Vào tháng 3-1995, nhân *1.2* được phổ biến. Điều đáng kể của Linux 1.2 so với Linux 1.0 ở chỗ nó hỗ trợ một phạm vi rộng và phong phú phần cứng, bao gồm cả kiến trúc tuyến phần cứng PCI mới. Nhân Linux 1.2 là nhân kết thúc dòng nhân Linux chỉ hỗ trợ PC.

Một điều cần lưu ý về cách đánh chỉ số các dòng nhân Linux. Hệ thống chỉ số được chia thành một số mức, chẳng hạn hai mức như 2.4 hoặc ba mức 2.2.5. Trong cách đánh chỉ số như vậy, quy ước rằng với các chỉ số từ mức thứ hai trở đi,

nếu là số chẵn thì dòng nhân đó đã khá ổn định và tương đối hoàn thiện, còn nếu là số lẻ thì dòng nhân đó vẫn đang được phát triển tiếp.

Tháng 6-1996, nhân *Linux 2.0* được phổ biến. Có hai đặc trưng nổi bật của Linux 2.0 là hỗ trợ kiến trúc phức hợp, bao gồm cả cổng *Alpha 64-bit* đầy đủ, và hỗ trợ kiến trúc đa bộ xử lý. Phân phối nhân Linux 2.0 cũng chỉ thi hành được trên bộ xử lý *Motorola 68000* và kiến trúc *SPARC* của SUN. Các thi hành của Linux dựa trên vi nhân *GNU Mach* cũng chạy trên PC và *PowerMac*.

Tới năm 2000, nhân *Linux 2.4* được phổ biến. Một trong đặc điểm được quan tâm của nhân này là nó hỗ trợ mã ký tự *Unicode 32 bit*, rất thuận lợi cho việc xây dựng các giải pháp toàn diện và triệt để đối với vấn đề ngôn ngữ tự nhiên trên phạm vi toàn thế giới.

Với phiên bản *Linux 2.2.6*, bạn có thể làm việc trên môi trường đồ hoạ với các ứng dụng cao cấp như: các tiện ích đồ hoạ và nhiều tiện ích khác.

Linux khó có thể thành công được như hiện nay nếu không có các công cụ GNU của Tổ chức phần mềm miễn phí (*Free Software Foundation*). Trình dịch *gcc* của GNU đã giúp cho việc viết mã của Linux dễ dàng hơn rất nhiều. Thậm chí tổ chức này đã yêu cầu các bản Linux với các tiện ích kèm theo phải gọi là GNU/Linux.

Hệ điều hành *Berkley Unix (BSD)* cũng đóng một vai trò quan trọng đối với Linux trong việc làm cho hệ điều hành này trở nên phổ biến như hiện nay. Hầu hết các tiện ích đi kèm với Linux được chuyển sang từ BSD, đặc biệt là các công cụ về mạng và các tiện ích.

Hiện nay, Linux là một hệ điều hành Unix đầy đủ và độc lập. Nó có thể chạy X Window, TCP/IP, Emacs, Web, thư điện tử và các phần mềm khác. Hầu hết các phần mềm miễn phí và thương mại đều được chuyển lên Linux. Rất nhiều các nhà phát triển phần mềm đã bắt đầu chuyển sang viết trên Linux. Người ta đã thực hiện các phép đo *benchmarks* trên các hệ Linux và thấy rằng chúng thực hiện nhanh hơn khi thực hiện trên các trạm làm việc *Sun Microsystems* và *Compaq*, thậm chí nhiều khi còn nhanh hơn cả trên *Windows 98* và *Windows NT*. Thật khó có thể hình dung được hệ điều hành Unix "*tí hon*" này phát triển nhanh như thế nào.

I.2.2 Vấn đề phân phối và giấy phép

Về lý thuyết, mọi người có thể khởi tạo một hệ thống Linux bằng cách tiếp nhận bản mới nhất các thành phần cần thiết từ các *site ftp* và biên dịch chúng. Trong thời kỳ đầu tiên, người dùng Linux phải tiến hành toàn bộ các thao tác này và vì vậy công việc là khá vất vả. Tuy nhiên, do có sự tham gia đông đảo của các cá nhân và nhóm phát triển Linux, họ đã tiến hành thực hiện nhiều giải pháp nhằm làm cho công việc khởi tạo hệ thống đỡ vất vả. Một trong những giải pháp điển hình nhất là cung cấp tập các gói chương trình đã tiền dịch, chuẩn hóa.

Những tập hợp như vậy hay những bản phân phối là lớn hơn nhiều so với hệ thống Linux cơ sở. Chúng thường bao gồm các tiện ích bổ sung cho khởi tạo hệ thống, các thư viện quản lý, cũng như nhiều gói đã được tiền dịch, sẵn sàng khởi tạo của nhiều bộ công cụ Unix dùng chung, chẳng hạn như phục vụ tin, trình duyệt web, công cụ xử lý, soạn thảo văn bản và thậm chí các trò chơi.

Cách thức phân phối ban đầu rất đơn giản song ngày càng được nâng cấp và hoàn thiện bằng phương tiện quản lý gói tiên tiến. Các bản phân phối ngày nay bao gồm các cơ sở dữ liệu tiến hóa gói, cho phép các gói dễ dàng được khởi tạo, nâng cấp và loại bỏ.

Nhà phân phối đầu tiên thực hiện theo phương châm này là *Slakware*, và chính họ mang lại những chuyển biến mạnh mẽ trong cộng đồng Linux đối với công việc quản lý gói khởi tạo Linux. Tiện ích quản lý gói RPM (*RedHat Package Manager*) của công ty Red Hat là một trong những phương tiện điển hình.

Nhân Linux là phần mềm tự do được phân phối theo *Giấy phép sở hữu công cộng phần mềm GNU GPL*.

Vấn đề bản quyền của dự án GNU:

Các chương trình tuân theo *GNU Copyleft* hay *GPL (General Public License)* có bản quyền như sau:

- Tác giả vẫn là sở hữu của chương trình của mình.

- Ai cũng có quyền bán copy của chương trình với giá bất kỳ mà không phải trả cho tác giả ban đầu.

- Người sở hữu chương trình tạo điều kiện cho người khác sao chép chương trình nguồn để phát triển tiếp chương trình.

Phần mềm ứng dụng chạy trên nền Linux tuy đã phong phú song so với một số hệ điều hành khác đặc biệt là so sánh với MS Windows, thì cẫn còn khoảng cách. Với sự hỗ trợ của nhiều công ty tin học hàng đầu thế giới (*IBM, SUN, HP, ...*) và sự tham gia phát triển của hàng vạn chuyên gia trên toàn thế giới thuộc cộng đồng Linux, các khó khăn của Linux chắc chắn sẽ nhanh chóng được khắc phục.

Chính vì lẽ đó đã hình thành một số nhà cung cấp Linux trên thế giới. Bảng dưới đây là tên của một số nhà cung cấp Linux có tiếng nhất và địa chỉ website của họ.

Đáng chú ý nhất là *Red Hat Linux* (tại Mỹ) và *Red Flag Linux* (tại Trung Quốc). Red Hat được coi là lâu đời và tin cậy, còn Red Flag là một công ty Linux của Trung Quốc, có quan hệ với cộng đồng Linux Việt Nam và chúng ta có thể học hỏi một cách trực tiếp kinh nghiệm cho quá trình đưa Linux vào Việt Nam.

Tên công ty	**Địa chỉ Website**
Caldera OpenLinux	www.caldera.com
Corel Linux	www.corel.com
Debian GNU/Linux	www.debian.com
Linux Mandrake	www.mandrake.com
Red Hat Linux	www.redhat.com
Red Flag Linux	www.redflag-linux.com
Slackware Linux	www.slackware.com
SuSE Linux	www.suse.com
TurboLinux	www.turbolinux.com

I.3 Tại sao dùng Linux?

Nếu bạn đã có máy tính trong tay, đầu tiên bạn phải có một hệ điều hành cài đặt trên đó bạn mới có thể sử dụng được các chương trình ứng dụng. Hệ điều hành là chương trình điều hành mọi hoạt động trong máy tính của bạn, mọi chương trình ứng dụng khác đều chạy trên nền của hệ điều hành này.

Sau đây là những lý do cho bạn lựa chọn hệ điều hành Linux cài đặt trên máy tính của mình:

✓ *Linux là hệ điều hành mã nguồn mở*, với nhiều tính năng giống các hệ điều hành khác và được cung cấp miễn phí cho người sử dụng.

✓ *Linux đầy đủ:* Tất cả những gì có ở IBM, SCO, Sun, … đều có ở Linux, như: *C compiler, perl interpeter, shell, TCP/IP, Proxy, firewall, tài liệu hướng dẫn, ...*

✓ *Linux rất mềm dẻo trong cấu hình*, thông qua các tiện ích, dễ dàng sửa đổi ngay cả nhân. Linux là hệ điều hành linh động, tin cậy, an toàn và được tiếp tục phát triển với hàng ngàn lập trình viên trên toàn thế giới.

✓ *Linux được trợ giúp*. Tài liệu giới thiệu Linux ngày càng nhiều, không thua kém bất cứ một hệ điều hành nào khác. Linux được nhiều tổ chức và công ty lớn trên thế giới sử dụng: *IBM, HP, Cisco, Google, Amazon.com*, …

Ngoài ra khi sử dụng hệ điều hành Linux các bạn còn có được các tính năng sau:

✓ *Tính ổn định:* Linux có tính ổn định cao, đây là một trong những ưu điểm của Linux so với các hệ điều hành khác. Tính ổn định ở đây có nghĩa là nó ít bị lỗi khi sử dụng so với hầu hết các hệ điều hành khác. Người sử dụng Linux sẽ không phải lo lắng đến chuyện máy tính của mình bị hiện tượng "*treo cứng*" khi đang sử dụng nữa. Thông thường lý do để bạn bắt buộc phải khởi động lại hệ thống là do mất điện, nâng cấp phần cứng hoặc phần mềm. Ngay cả server Linux phục vụ những mạng lớn (hàng trăm máy trạm) cũng hoạt động rất ổn định.

✓ *Tính bảo mật:* Khi làm việc trên Linux người dùng có thể yên tâm hơn về tính bảo mật của hệ điều hành. Linux là hệ điều hành đa nhiệm, đa người dùng, điều này có nghĩa là nhiều người sử dụng có thể vào phiên làm việc của mình trên cùng một máy vào tại cùng một thời điểm. Linux cung cấp các mức bảo mật khác nhau cho người sử dụng. Mỗi người sử dụng chỉ làm việc trên một không gian tài nguyên dành riêng, chỉ có người quản trị hệ thống mới có quyền thay đổi trong máy.

✓ *Tính hoàn chỉnh:* Bản thân Linux đã được kèm theo các trình tiện ích cần thiết. Tất cả các trình tiện ích mà bạn mong đợi đều có sẵn ở một dạng tương đương

rất giống. Trên Linux, các trình biên dịch như C, C++, … các hạt nhân hay TCP/IP đều được chuẩn hoá.

✓ *Tính tương thích:* Linux tương thích hầu như hoàn toàn với một số chuẩn UNIX như *IEEE POSIX.1*, *UNIX System V* và *BSD UNIX*. Trên Linux bạn cũng có thể tìm thấy các trình giả lập của DOS và Windows cho phép bạn có thể chạy các ứng dụng quen thuộc trên DOS và Windows. Linux cũng hỗ trợ hầu hết các phần cứng máy PC.

✓ *Hệ điều hành 32 bit đầy đủ:* Ngay từ đầu Linux đã là hệ điều hành 32 bit đầy đủ. Điều đó có nghĩa là bạn không còn phải lo về các giới hạn bộ nhớ, các trình điều khiển EMM hay các bộ nhớ mở rộng, … khi sử dụng Linux. Hiện nay đã có những phiên bản Linux 64 bits chạy trên máy *Alpha Digital* hay *Ultra Sparc.*

✓ *Dễ cấu hình:* Bạn không còn phải bận tâm về các giới hạn 640K và tiến hành tối ưu hoá bộ nhớ mỗi lần cài đặt một trình điều khiển mới. Linux cho bạn hầu như toàn quyền điều khiển về cách làm việc của hệ thống.

✓ *Khả năng làm việc trên nhiều loại máy:* Cấu hình phần cứng tối thiểu mà Linux cần chỉ là chip 80386, 2MB bộ nhớ, 10-20 MB không gian đĩa để bắt đầu. Khi bạn càng bổ sung phần cứng thì Linux chạy càng nhanh. Linux có khả năng chạy trên nhiều dòng máy khác nhau như *Apple Macintosh, Sun, Dec Alpha* và *Power PC*.

I.4 Kí hiệu nhân Linux

Một trong những điểm dễ nhầm lẫn của những người mới làm quen với Linux là các version của Linux. Khi bạn bắt đầu tiếp cận với Linux, thông thường bạn tìm các phiên bản Linux trên CD ROM như "*Red Hat version 6.1*" hay "*SuSE Linux version 6.3*". Các số gắn với một phiên bản chỉ liên quan tới các lần phân phối, một phiên bản với số hiệu lớn hơn không có nghĩa là phần mềm được cập nhật mới hơn. Thông thường các nhân Linux (*Linux kernel*) có một số hiệu phiên bản riêng. Tại mỗi một thời điểm có hai phiên bản mới nhất – phiên bản ổn định (*stable*) và phiên bản phát triển (*development*). Phiên bản ổn định dành cho hầu hết người dùng, còn phiên bản phát triển thay đổi rất nhanh và được chạy thử bởi các nhà phát triển trên Internet. Mỗi phiên bản kernel có thêm một số hiệu thứ ba mô tả lần cập

nhật cuối cùng. Chẳng hạn phiên bản Linux kernel ổn định mới nhất vào tháng 7/99 là *2.2.10* và phiên bản phát triển tương ứng là *2.3.11*.

Các phiên bản của hệ điều hành Linux được xác định bởi hệ thống số dạng **X.YY.ZZ**.

- Số "**X**" chỉ tăng khi xảy ra những thay đổi rất quan trọng, những thay đổi làm cho phần mềm không thể hoạt động đúng đắn với những phần mềm khác.

- Nếu **YY** là số chẵn => phiên bản ổn định. **YY** là số lẻ => phiên bản thử nghiệm.

- Số "**Z**" xác định chính xác phiên bản của hạt nhân bạn dùng, nó được tăng mỗi phiên bản.

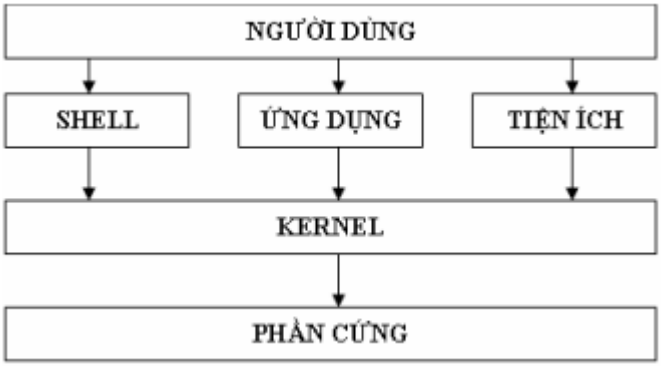
Ví dụ: Kernel 2.4.2

2 là Số chính

.4 là số phụ, phiên bản ổn định

.2 Patch Level.

I.5 Kiến trúc của Hệ điều hành Linux



I.5.1 Hạt nhân (Kernel)

Là trung tâm điều khiển của hệ điều hành Linux, chứa các mã nguồn điều khiển hoạt động của toàn bộ hệ thống. Hạt nhân được phát triển không ngừng, thường có 2 phiên bản mới nhất, một bản dạng phát triển mới nhất và một bản ổn định mới nhất. Kernel được thiết kế theo dạng modul, do vậy kích thước thật sự của Kernel rất nhỏ. Chúng chỉ tải những bộ phận cần thiết lên bộ nhớ, các bộ phận khác sẽ được tải nếu có yêu cầu sử dụng. Nhờ vậy so với các hệ điều hành khác Linux

không sử dụng lãng phí bộ nhớ nhờ không tải mọi thứ lên mà không cần quan tâm nó có sử dụng không.

Kernel được xem là trái tim của hệ điều hành Linux, ban đầu nhân được phát triển cho các CPU Intel 80386. Điểm mạnh của loại CPU này là khả năng quản lý bộ nhớ. Kernel của Linux có thể truy xuất tới toàn bộ tính năng phần cứng của máy. Yêu cầu của các chương trình cần rất nhiều bộ nhớ, trong khi hệ thống có ít bộ nhớ, hệ điều hành sử dụng không gian đĩa hoán đổi (*swap space*) để lưu trữ các dữ liệu xử lý của chương trình. *Swap space* cho phép ghi các trang của bộ nhớ xuất các vị trí dành sẵn trong đĩa và xem nó như phần mở rộng của vùng nhớ chính. Bên cạnh sử dụng *swap space*, Linux còn hỗ trợ các đặc tính sau:

- Bảo vệ vùng nhớ giữa các tiến trình, điều này không cho phép một tiến trình làm tắt toàn bộ hệ thống.

- Chỉ tải các chương trình khi có yêu cầu.

I.5.2 Shell

Shell cung cấp tập lệnh cho người dùng thao tác với kernel để thực hiện công việc. Shell đọc các lệnh từ người dùng và xử lý. Ngoài ra shell còn cung cấp một số đặc tính khác như: chuyển hướng xuất nhập, ngôn ngữ lệnh để tạo các tập tin lệnh tương tự tập tin ***bat*** trong DOS.

Có nhiều loại shell được dùng trong Linux. Điểm quan trọng để phân biệt các shell với nhau là bộ lệnh của mỗi shell. Ví dụ, *C shell* thì sử dụng các lệnh tương tự ngôn ngữ C, *Bourne Shell* thì dùng ngôn ngữ lệnh khác.

Shell sử dụng chính trong Linux là *GNU Bourne Again Shell (bash)*. Shell này là shell phát triển từ *Bourne Shell*, là shell sử dụng chính trong các hệ thống Unix, với nhiều tính năng mới như: điều khiển các tiến trình, các lệnh history, tên tập tin dài, ….

I.5.3 Các tiện ích

Các tiện ích được người dùng thường xuyên sử dụng. Nó dùng cho nhiều thứ như thao tác tập tin, đĩa, nén, sao lưu tập tin, … Tiện ích trong Linux có thể là các lệnh thao tác hay các chương trình giao diện đồ họa. Hầu hết các tiện ích dùng trong Linux là sản phẩm của chương trình GNU. Linux có sẵn rất nhiều tiện ích như trình

biên dịch, trình gỡ lỗi, soạn văn bản,… Tiện ích có thể được sử dụng bởi người dùng hoặc hệ thống. Một số tiện ích được xem là chuẩn trong hệ thống Linux như *passwd, ls, pa, vi …*

I.5.4 Chương trình ứng dụng

Khác với các tiện ích, các ứng dụng như chương trình word, hệ quản trị cơ sở dữ liệu,… là các chương trình có độ phức tạp lớn và được các nhà sản xuất viết ra.

I.6 Các đặc tính cơ bản của Linux

Linux hỗ trợ các tính năng cơ bản thường thấy trong các hệ điều hành Unix và nhiều tính năng khác mà không hệ điều hành nào có được. Linux cung cấp môi trường phát triển một cách đầy đủ bao gồm các thư viện chuẩn, các công cụ lập trình, trình biên dịch, debug,… như bạn mong đợi ở các hệ điều hành Unix khác. Hệ thống Linux trội hơn các hệ thống khác trên nhiều mặt mà người dùng quan tâm như sự phát triển tốc độ, dễ sử dụng và đặc biệt là sự phát triển và hỗ trợ mạng. Một số đặc điểm của Linux chúng ta cần quan tâm:

I.6.1 Đa tiến trình

Là đặc tính cho phép người dùng thực hiện nhiều tiến trình đồng thời. Ví dụ bạn vừa in, vừa soạn văn bản, vừa nghe nhạc,… cùng một lúc. Máy tính sử dụng chỉ một CPU nhưng xử lý đồng thời nhiều tiến trình cùng lúc. Thực chất là tại một thời điểm CPU chỉ xử lý được một mệnh lệnh, việc thực hiện cùng lúc nhiều công việc là giả tạo bằng cách làm việc xen kẽ và chuyển đổi trong thời gian nhanh. Do đó người dùng cứ ngỡ là thực hiện đồng thời.

I.6.2 Tốc độ cao

Hệ điều hành Linux được biết đến như một hệ điều hành có tốc độ xử lý cao, bởi vì nó thao tác rất hiệu quả đến tài nguyên như: bộ nhớ, đĩa,…

I.6.3 Bộ nhớ ảo

Khi hệ thống sử dụng quá nhiều chương trình lớn dẫn đến không đủ bộ nhớ chính (RAM) để hoạt động. Trong trường hợp đó, Linux dung bộ nhớ từ đĩa vào *partition swap*. Hệ thống sẽ đưa các chương trình hoặc dữ liệu nào chưa có yêu cầu

truy xuất xuống vùng swap này, khi có nhu cầu thì hệ thống chuyển lên lại bộ nhớ chính.

I.6.4 Sử dụng chung thư viện

Hệ thống Linux có rất nhiều thư viện dùng chung cho nhiều ứng dụng. Điều này sẽ giúp hệ thống tiết kiệm được tài nguyên cũng như thời gian xử lý.

I.6.5 Sử dụng các chương trình xử lý văn bản

Chương trình xử lý văn bản là một trong những chương trình rất cần thiết đối với người sử dụng. Linux cung cấp nhiều chương trình cho phép người dùng thao tác với văn bản như *vi, emacs, nroff*.

I.6.6 Sử dụng giao diện cửa sổ

Giao diện cửa sổ dùng hệ thống *X Window*, có giao diện như hệ điều hành Windows. Với hệ thống này người dùng rất thuận tiện khi làm việc trên hệ thống. *X Window System* hay còn gọi tắt là X được phát triển tại viện *Massachusetts Institute of Technology*. Nó được phát triển để tạo ra môi trường làm việc không phụ thuộc phần cứng. X chạy dưới dạng *client – server*. Hệ thống *X Window* hoạt động qua hai bộ phận:

- Phần server còn gọi là *X server.*

- Phần client được gọi là *X Window manager* hay *desktop environment.*

X Server sử dụng trong hầu hết các bản phân phối của Linux là *Xfree86*. Client sử dụng thường là KDE (*K Desktop Environment*) và GNOME (*GNU Network Object Model Environment*).

Dịch vụ Samba sử dụng tài nguyên đĩa, máy in với Windows. Tên Samba xuất phát từ giao thức *Server Message Block* (SMB) mà Window sử dụng để chia sẻ tập tin và máy in. Samba là chương trình sử dụng giao thức SMB chạy trên Linux. Sử dụng Samba bạn có thể chia sẻ tập tin và máy in với các máy Windows

I.6.7 Network Information Service (NIS)

Dịch vụ NIS cho phép chia sẻ các tập tin *password* và *group* trên mạng. NIS là một hệ thống cơ sở dữ liệu dạng client – server, chứa các thông tin của người

dùng và dùng để chứng thực người dùng. NIS xuất phát từ hãng *Sun Microsystem* với tên là *Yellow Pages*.

I.6.8 Lập lịch hoạt động chương trình, ứng dụng

Chương trình lập lịch trong Linux xác định các ứng dụng, script thực thi theo một sự sắp xếp của người dùng như: *at*, *cron*, *batch*.

I.6.9 Các tiện ích sao lưu dữ liệu

Linux cung cấp các tiện ích như *tar*, *cpio* và *dd* để sao lưu và backup dữ liệu. Red Hat Linux còn cung cấp tiện ích *Backup and Restore System Unix* (BRU) cho phép tự động backup dữ liệu theo lịch.

I.6.10 Hỗ trợ nhiều ngôn ngữ lập trình

Linux cung cấp một môi trường lập trình Unix đầy đủ bao gồm các thư viện chuẩn, các công cụ lập trình, trình biên dịch, chương trình debug mà bạn có thể tìm thấy trong các hệ điều hành Unix khác. Ngôn ngữ chủ yếu sử dụng trong các hệ điều hành Unix là C và C++. Linux dùng trình biên dịch cho C và C++ là *gcc*, chương trình biên dịch này rất mạnh, hỗ trợ nhiều tính năng. Ngoài C, Linux cũng cung cấp các trình biên dịch, thông dịch cho các ngôn ngữ khác như *Pascal, Fortan, Java, …*

I.7 Câu hỏi và bài tập

Câu 1:

Phiên bản chính thức đầu tiên của Linux xuất hiện đầu tiên vào năm nào? Tên tác giả đầu tiên phát triển hệ điều hành Linux?

Câu 2:

Trình bày quá trình phát triển của hệ điều hành Linux?

Câu 3:

Kể tên một vài bản phân phối của hệ điều hành Linux?

Câu 4:

Vì sao Linux được gọi là một hệ điều hành "*mã nguồn mở*"? Điều đó có ảnh hưởng như thế nào tới quá trình phát triển của Linux?

Câu 5:

Trình bày những lý do để lựa chọn cài đặt hệ điều hành Linux?

Câu 6:

Sự khác nhau giữa phiên bản phân phối của Linux và nhân Linux?

Câu 7:

Kiến trúc của hệ điều hành Linux gồm mấy phần? Hãy kể tên?

Câu 8:

Trình bày các đặc tính cơ bản của hệ điều hành Linux?

CHƯƠNG II. CÀI ĐẶT HỆ ĐIỀU HÀNH LINUX

Thông thường trên các đĩa của bản phân phối Linux đã có hướng dẫn ngắn gọn cách cài đặt Linux. Ngoài ra, trên Internet bạn có thể tìm thấy rất nhiều cuốn sách nói về vấn đề cài đặt hệ điều hành Linux. Và tất cả các bản phân phối lớn *(Debian, Slackware, RedHat, …)* đều đã có cuốn hướng dẫn cài đặt rất chi tiết, cho mọi tình huống sử dụng. Chương II sẽ giới thiệu cho các bạn cách cài đặt đơn giản nhất phiên bản Red Hat của Linux.

II.1 Công tác chuẩn bị

II.1.1 Yêu cầu hệ thống

Linux không đòi hỏi máy có cấu hình mạnh. Tuy nhiên nếu phần cứng có cấu hình thấp quá thì có thể không chạy được X-Window hay các ứng dụng có sẵn. Cấu hình tối thiểu nên dùng:

- CPU: Pentium MMX trở lên.
- RAM: 64MB trở lên cho Text mode, 192MB cho Graphics mode.
- Đĩa cứng: Dung lượng đĩa còn phụ thuộc vào loại cài đặt.
    - Custum Installation (mininum): 520MB
    - Server (minimum): 870MB
    - Personal Desktop: 1.9GB
    - Workstation: 2.4GB
    - Custum Installation (everything): 5.3GB
- 2MB cho card màn hình nếu muốn sử dụng chế độ đồ họa.

II.1.2 Lựa chọn phiên bản cài đặt

Trước khi bạn chọn phiên bản muốn dùng, hãy đọc các mô tả cẩn thận và so sánh với nhu cầu của bạn.

Mỗi phiên bản được thiết kế cho một loại người dùng riêng biệt. Vài phiên bản được tối ưu để hoạt động như server, vài phiên bản khác được dùng để chơi

game, vài phiên bản khác lại được dùng cho máy để bàn và các ứng dụng văn phòng.

Có một số ít phiên bản được xem là sự lựa chọn cho người dùng mới:

- *RedHat* đặc biệt tốt cho server.
- *Mandrake* là hệ thống để bàn xuất sắc.
- *SuSE* cũng là hệ thống để bàn xuất sắc.

Các thông tin và tài nguyên (resource) của Linux có thể tìm thấy ở khắp nơi trên Internet và hầu hết đều *free*. Mặt khác các gói phần mềm ứng dụng cũng được cho miễn phí.

II.1.3 Lựa chọn giữa cài đặt mới hoặc nâng cấp

II.1.4 Phân vùng đĩa

- Điều này cho phép bạn tạo các phân vùng mới trên không gian trống của đĩa, hoặc để sử dụng các phân vùng Linux đã tồn tại.

- Cấp phát không gian trao đổi thích hợp.

- Xác định hệ thống file nào để sử dụng.

➢ ***Lưu ý:***

- Cần chuẩn bị phân vùng đĩa còn trống để cài Linux.

- Linux cần tối thiểu hai phân vùng là *Linux Native* (ext3) và *Linux swap*. Đơn giản, bạn có thể dùng *Partition Magic* để phân chia đĩa.

- Một partition là *Linux native* ext3. Cần khoảng 2GB trở lên để cài Linux, bao gồm cả KDE và GNOME, các tiện ích đồ họa, multimedia, và lập trình. Tối thiểu bạn cần 400MB và cài toàn bộ là 4,5GB.

- Một partition là *Linux swap*, là phân vùng trao đổi của Linux dành cho việc sử dụng bộ nhớ ảo làm không gian trao đổi. Thông thường, dung lượng bộ nhớ ảo tối ưu sẽ gấp đôi dung lượng bộ nhớ RAM của hệ thống.

II.1.5 Lựa chọn phương thức cài đặt

Có thể cài đặt Linux bằng một trong các cách sau:

- Từ ổ đĩa CD-ROM.

- Từ bản sao chép Linux trên ổ đĩa cứng hoặc USB.

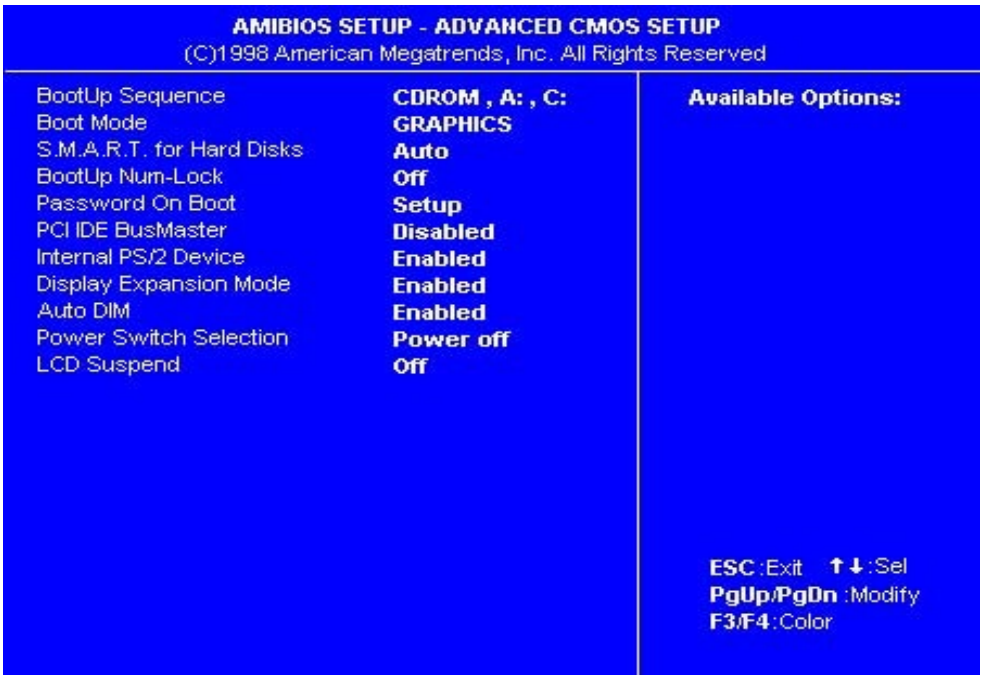
- Cài đặt từ server mạng, sử dụng HTTP, FTP, hoặc NFS, …

II.2 Quá trình cài đặt

II.2.1 Quá trình cài đặt hệ điều hành Red Hat Linux 9.0

- **Boot từ CD-ROM** ***(Hình 2.1)***

Nếu máy bạn có CD-ROM, bạn hãy khởi động máy tính, chỉnh lại BIOS thứ tự boot đầu tiên là CD-ROM và đưa đĩa cài đặt vào ổ CD.



Hình 2.1

- **Chọn chế độ cài đặt** ***(Hình 2.2)***

Chúng ta có thể chọn các chế độ:

- Chương trình hệ điều hành Linux đặt dưới chế độ đồ họa (*Graphical mode*) -> [Enter].

- Linux text: Chương trình hệ điều hành Linux đặt dưới chế độ text (*Text mode*).

Hình 2.2

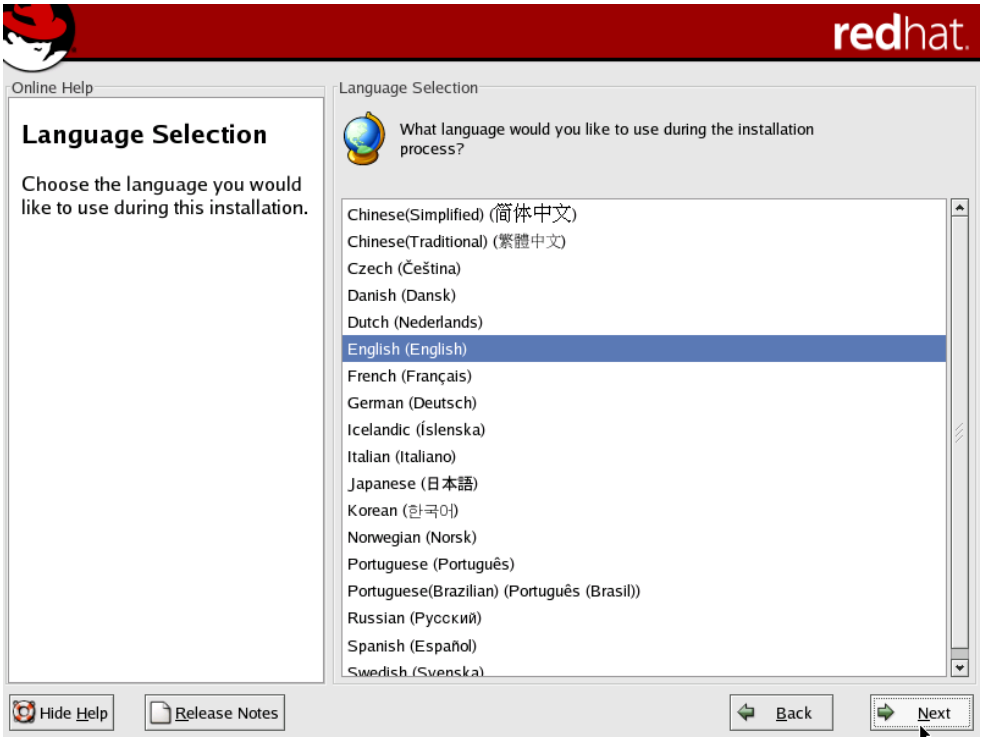
- **Kiểm tra đĩa trước khi cài đặt** ***(Hình 2.3)***

Hình 2.3

Chọn ***OK*** để test đĩa, hoặc chọn ***Skip*** để bỏ qua quá trình test.

- **Chọn ngôn ngữ hiển thị trong quá trình cài đặt** ***(Hình 2.4)***

Ta chọn ngôn ngữ ***"English"***. Sau khi chọn nhấn ***Next*** để tiếp tục.



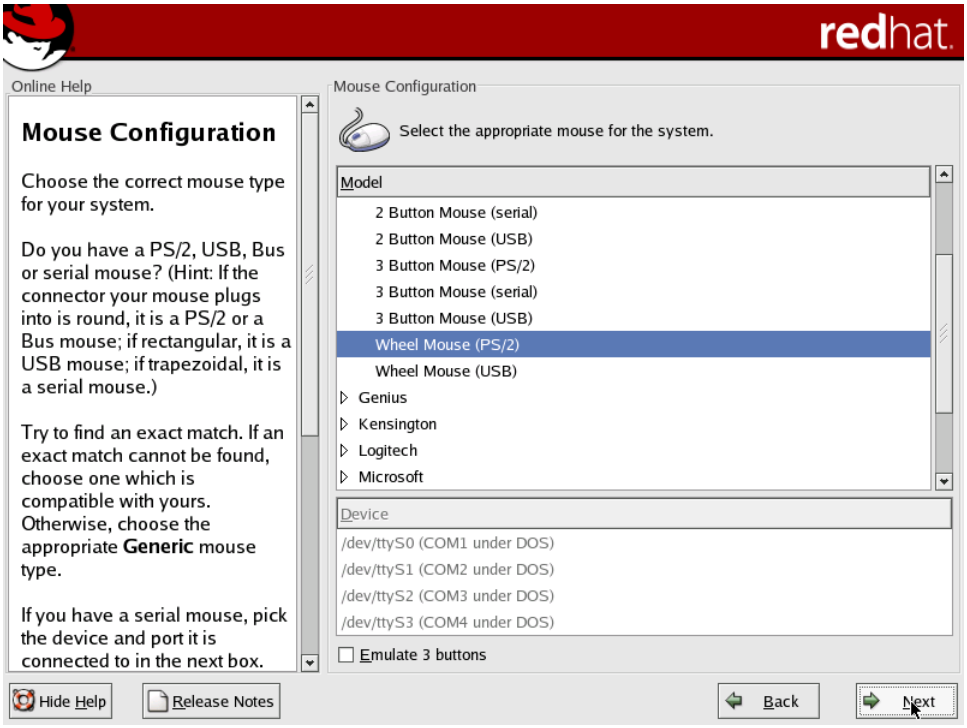
Hình 2.4

- **Lựa chọn Keyboard** ***(Hình 2.5)***

Hình 2.5

Chọn kiểu bàn phím thích hợp với hệ thống, chọn ***Next.***

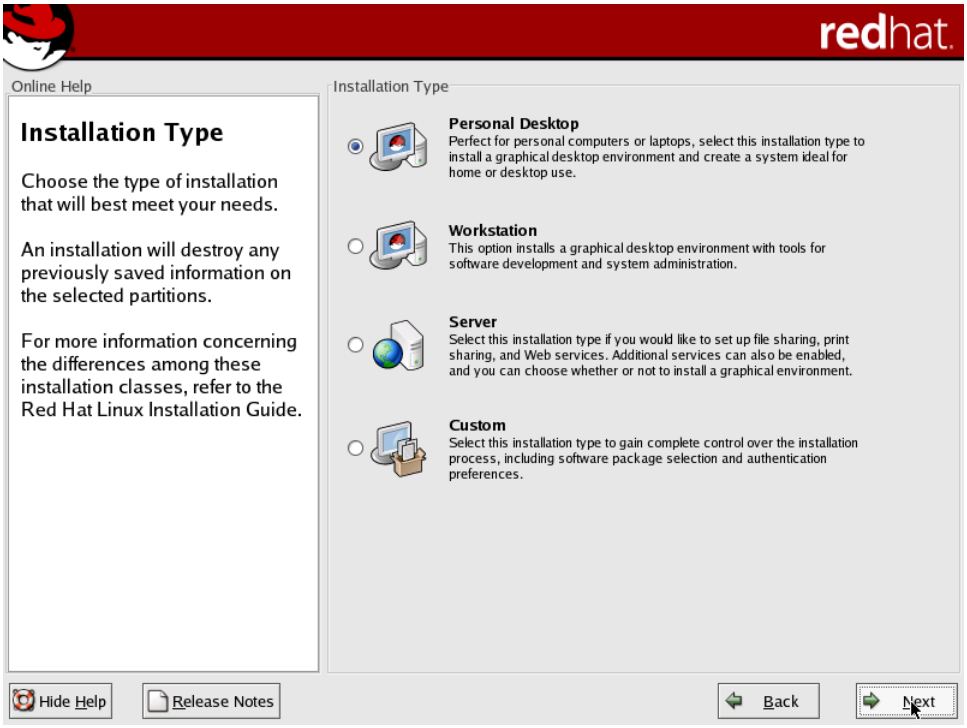
- **Lựa chọn chuột** ***(Hình 2.6)***



Hình 2.6

Chọn loại mouse phù hợp với mouse của mình. Khi chọn lưu ý cổng gắn mouse kà serial hay PS/2, chọn ***Next.***

- **Lựa chọn kiểu cài đặt *(Hình 2.7)***



Hình 2.7

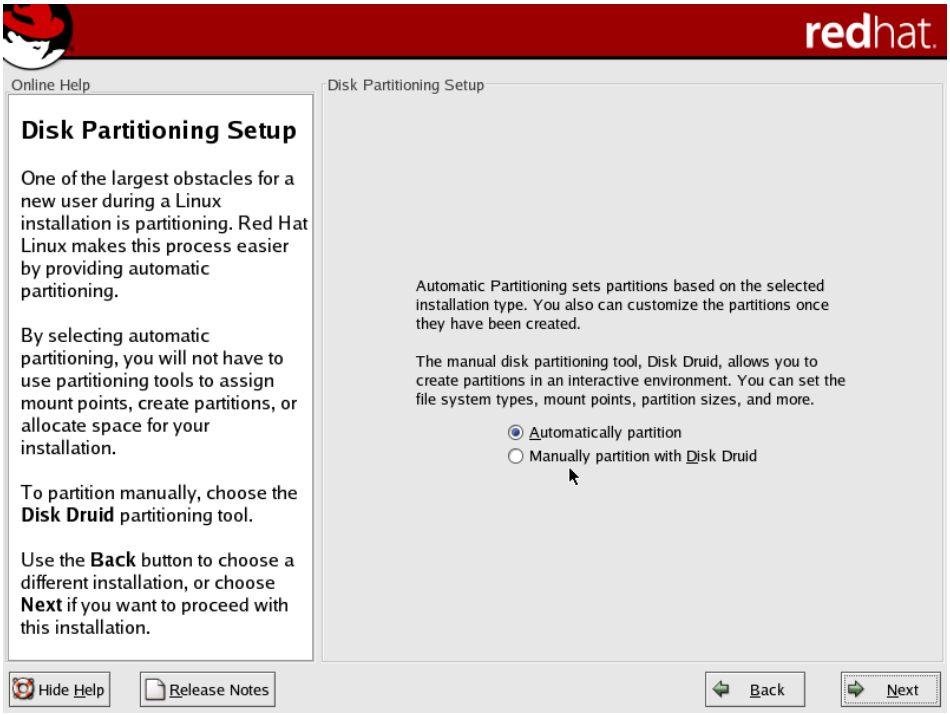
Personal Desktop: dành cho người mới bắt đầu với Linux hoặc cho những hệ thống desktop cá nhân. Chương trình cài đặt sẽ chọn lựa những gói phần mềm cần thiết nhất cho cấu hình này. Dung lượng đĩa cần cho kiểu cài đặt này chiếm khoảng 1.5GB, bao gồm cả môi trường đồ hoạ.

WorkStation: dành cho những trạm làm việc với chức năng đồ hoạ cao cấp và các công cụ phát triển.

Server: cài đặt hệ thống đóng vai trò máy chủ như *web server, ftp sever, SQL server,...*

Custom: đây là lựa chọn linh hoạt cho bạn trong quá trình cài đặt. Bạn có thể chọn các gói phần mềm, các môi trường làm việc, *boot loader* tuỳ theo ý bạn.

- **Chọn cách chia partition *(Hình 2.8)***



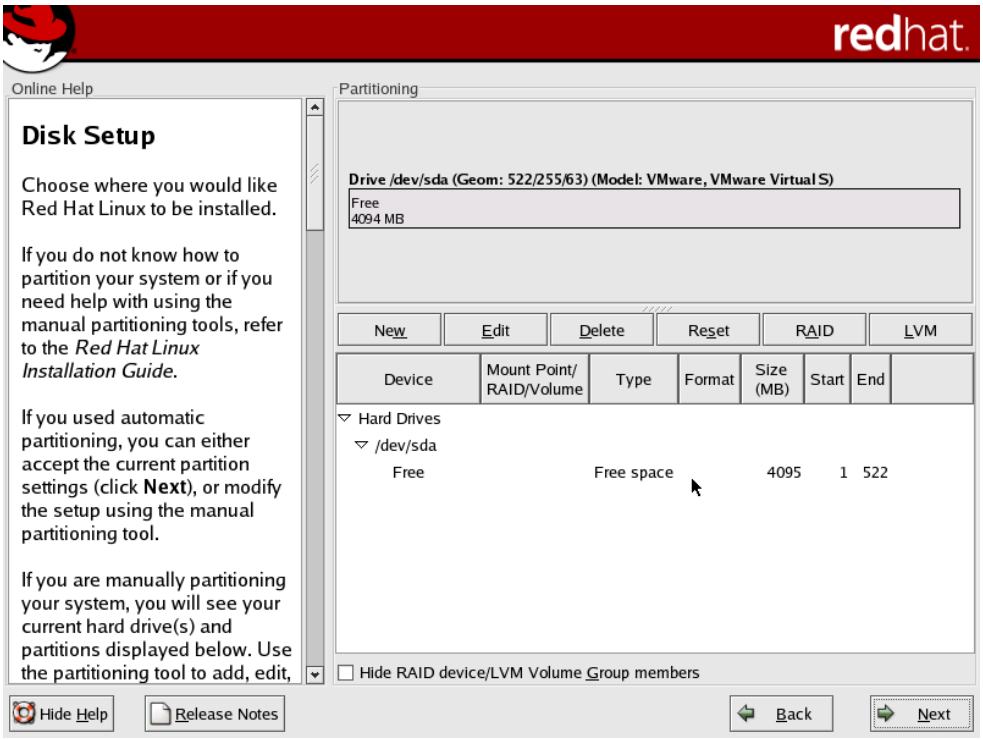
Hình 2.8

- ***Automatically partition***: cho phép hệ thống tự động phân vùng ổ đĩa hợp lý để cài hệ điều hành (thông thường theo cách này thì hệ thống sẽ tạo ra hai phân vùng: */boot*, */swap*).

- ***Manually partition with Disk Druid***: chia partition bằng tiện ích Disk Druid. Đây là cách chia partition dưới dạng đồ họa dễ dùng.

Nếu ta là người mới học cách cài đặt thì nên lựa chọn ***Automatically partition.***

✓ ***Chia partition bằng tay***



Hình 2.9

- ***Remove all Linux partitions on this system:*** khi ta muốn loại bỏ tất cả các Linux partition có sẵn trong hệ thống.

- ***Keep all partitions and use existing free space:*** khi ta muốn giữ lại tất cả các partition có sẵn và chỉ sử dụng không gian trống còn lại để phân vùng.

- Tùy theo từng yêu cầu riêng mà ta có thể lựa chọn các yêu cầu trên cho phù hợp, sau đó chọn ***Next***.

Khi ta chọn cách chia partition bằng tay ta sẽ sử dụng tiện ích ***Disk Druid***.

Disk Druid hiển thị các partition của đĩa dưới chế độ đồ họa ở phía trên. Bạn có thể chọn từng partition để thao tác.

Chi tiết các partition gồm kích thước, loại hệ thống tập tin, thư mục được mount vào.

- ***New***: tạo một partition mới, chỉ định tên phân vùng (*mount point*), loại filesystem (*ext3*) và kích thước (*size*) tính bằng MByte (tùy chọn).

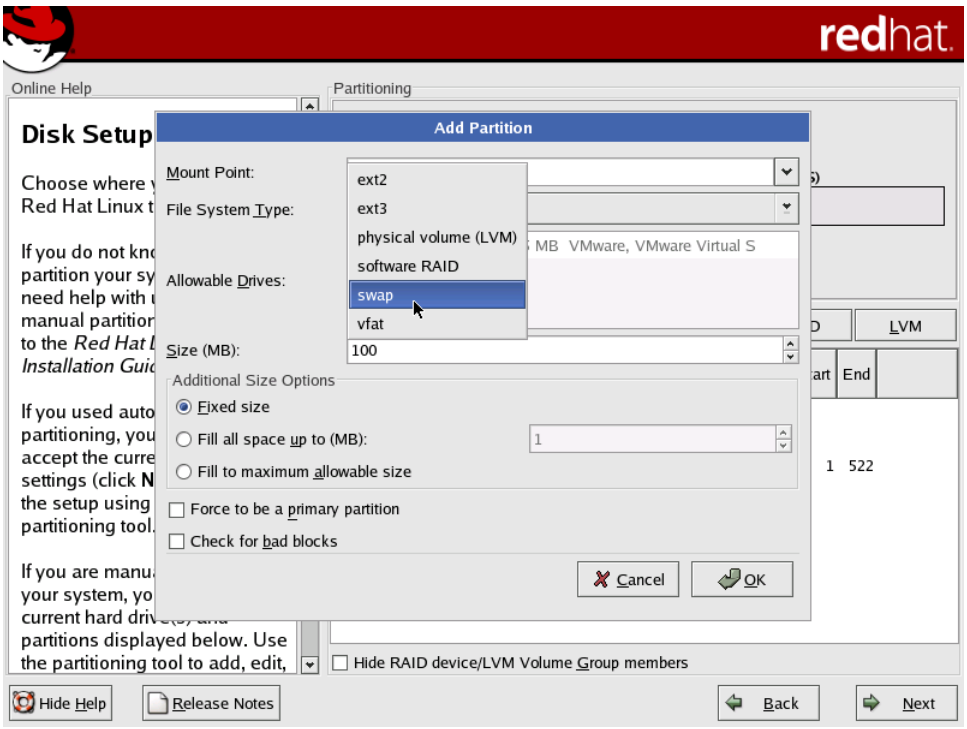
- ***Edit***: thay đổi lại các tham số của phân vùng được chọn.

- ***Delete***: xóa phân vùng được chọn.

- ***Reset***: phục hồi lại trạng thái đĩa như trước khi thao tác.

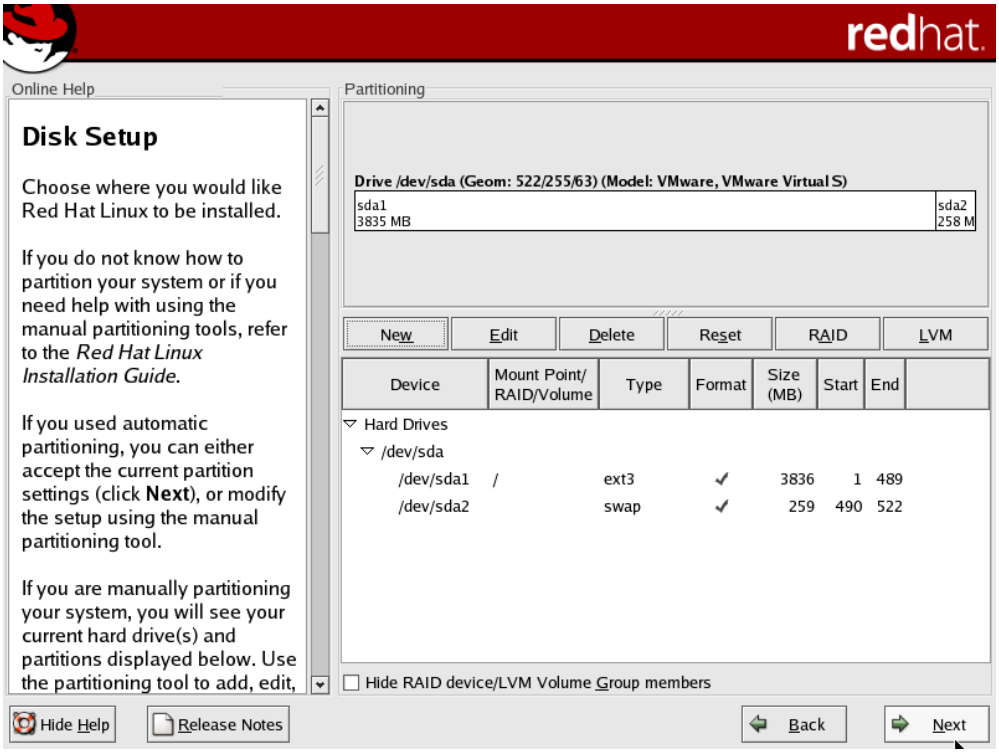
- ***Make RAID***: sử dụng với RAID (*Redundant Array of Independent Disks*) khi ta có ít nhất 3 đĩa cứng.

Tạo ra vùng swap *(Hình 2.10)*



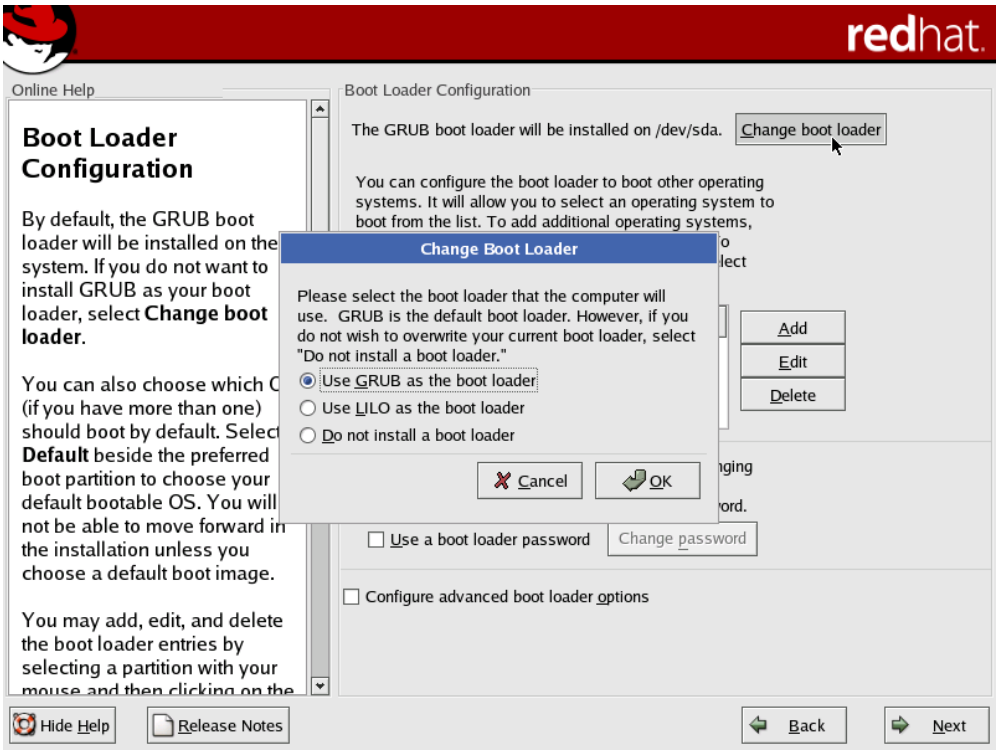
Hình 2.10

Tạo partition ext3, sau khi tạo swap partition *(Hình 2.11)*



Hình 2.11

➢ **Cấu hình Boot Loader**



Hình 2.12

Đây là chương trình dùng để khởi động Linux khi bạn có hơn một hệ điều hành được cài đặt trên hệ thống. Boot Loader cho phép bạn chọn các hệ điều hành để khởi động qua menu. Khi chúng ta chọn, chúng sẽ xác định các tập tin cần thiết để khởi động hệ điều hành và giao quyền điều khiển lại cho hệ điều hành. Boot Loader có thể được cài vào Master Boot Record hoặc vào sector đầu tiên của partition.

Linux cho phép bạn sử dụng chương trình Boot Loader là ***GRUB*** hoặc ***LILO***. Cả 2 Boot Loader đều có thể hỗ trợ quản lý nhiều hệ điều hành trên một hệ thống.

- Bạn chọn cài Boot Loader vào *Master Boot Record (MBR)* khi chưa có chương trình Boot Loader nào (ví dụ như của Windows) được cài, hoặc bạn chắc chắn chương trình Boot Loader của bạn có thể khởi động được các hệ điều hành khác trong máy của mình. Khi cài lên MBR thì các chương trình Boot Loader trước đó sẽ bị thay thế bằng Boot Loader mới.

- Chọn cài Boot Loader vào sector đầu tiên của partition cài đặt khi bạn đã có chương trình Boot Loader tại MBR và không muốn thay thế nó. Trong trường hợp này, chương trình Boot Loader kia nắm quyền điều khiển trước và trỏ đến chương trình Boot Loader của Linux khi có yêu cầu khởi động hệ điều hành này.

- Bạn không cài chương trình Boot Loader, khi đó bạn phải sử dụng đĩa mềm boot để khởi động hệ điều hành.

- Ta có thể đặt mật khẩu cho Boot Loader thông qua nút ***Change password***.

GRUB: Là boot loader mặc định, đây là chương trình rất mạnh và uyển chuyển.

GRUB tự động dò các hệ điều hành hiện có trên hệ thống và thêm vào danh sách khi khởi động.

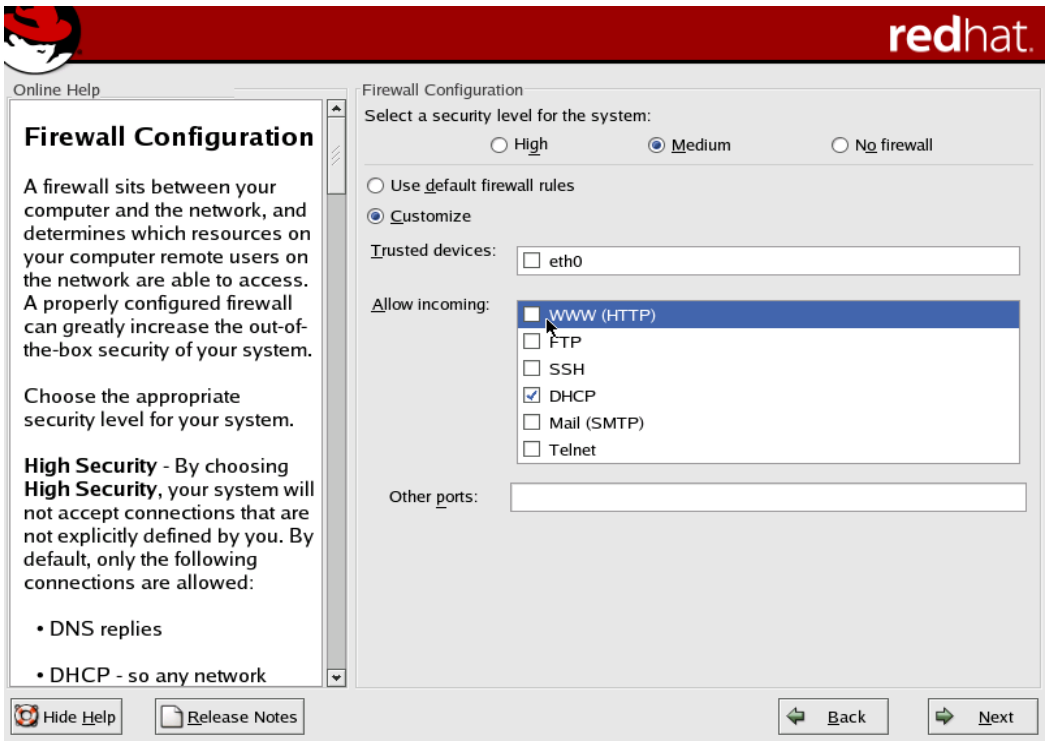
Tuỳ chọn "***configure advance boot loader option***" cho phép bạn chọn việc cài GRUB lên đâu trong ổ cứng.

Nếu chọn GRUB để khởi động hệ thống, GRUB sẽ được cài lên Master Boot Record (/*dev/hda*).

Nếu chọn một chương trình khác để khởi động như ***system commander*** chẳng hạn, bạn hãy chọn cài GRUB lên "***first sector of boot partition***". Như vậy, system commander sẽ tự động nhận ra Linux và thêm vào mục nhập khởi động cho Linux.

➢ **Cấu hình Firewall**

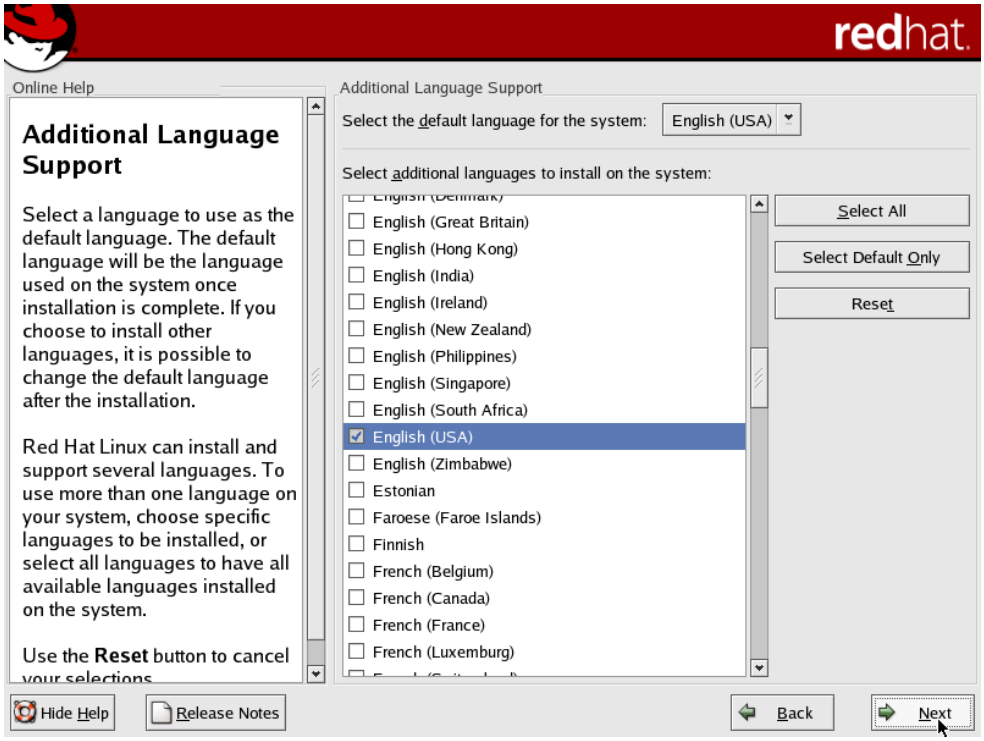
Trong Linux có tích hợp Firewall để bảo vệ hệ thống chống lại một số truy xuất bất hợp pháp từ bên ngoài. Ta chọn ***Enable Firewall***, sau đó chọn loại dịch vụ cần cho phép bên ngoài truy cập vào Firewall.



Hình 2.13

➢ **Lựa chọn ngôn ngữ cho hệ thống *(Hình 2.14)***

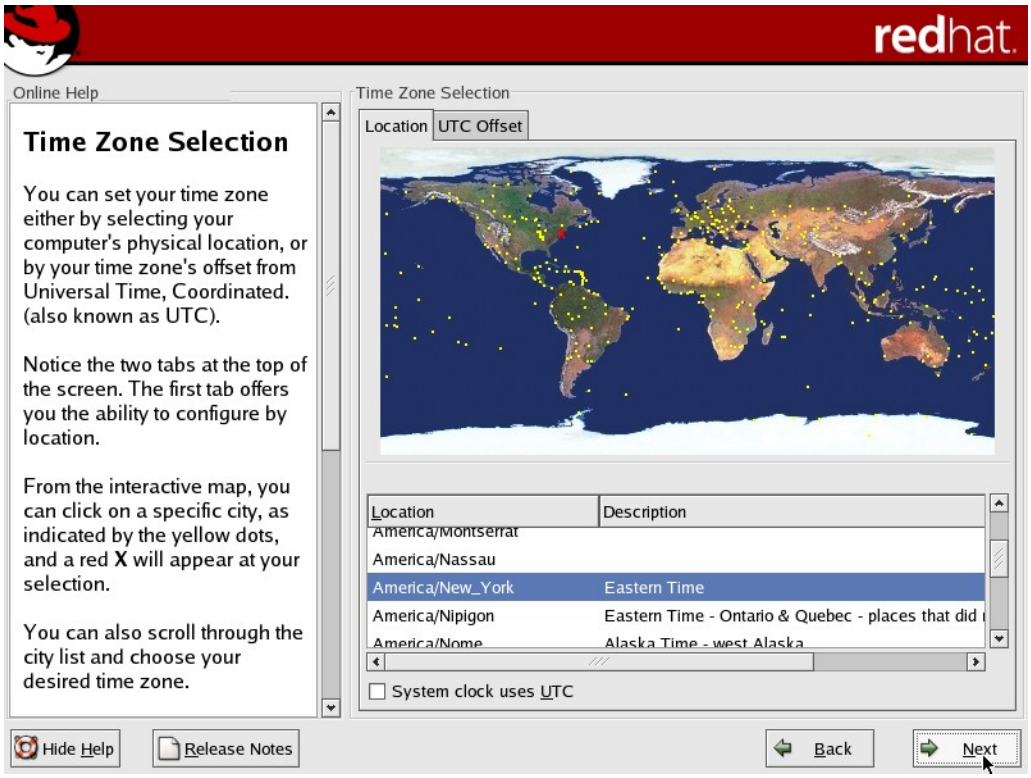
Bạn có thể cài đặt và sử dụng nhiều ngôn ngữ trong Linux. Có thể chọn ngôn ngữ mặc định (*English (USA)*) và các ngôn ngữ khác để sử dụng.



Hình 2.14

➢ **Lựa chọn vùng định thời gian** ***(Hình 2.15)***

Các vị trí chia theo châu lục. Ở Việt Nam là ***Asia/Saigon***, ta có thể chọn mục này một cách dễ dàng thông qua việc định vị chuột tại đúng vị trí trên bản đồ.



Hình 2.15

➢ **Thiết lập password cho user *root (Hình 2.16)***

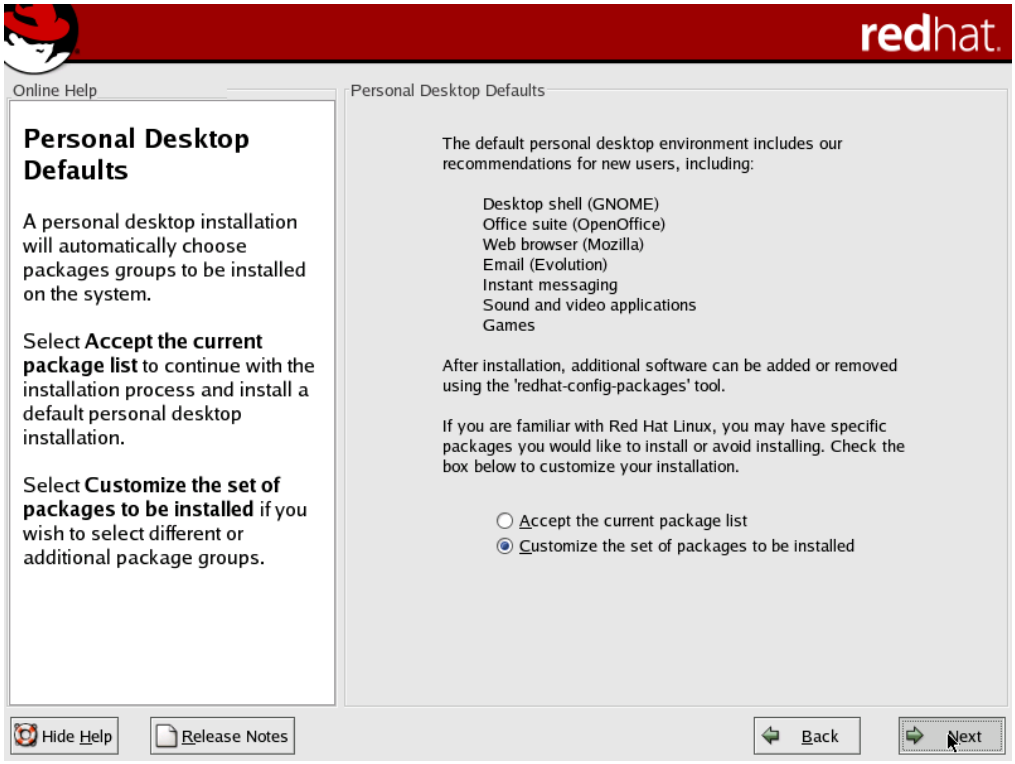
Trên Linux người quản trị thường được gọi là người dùng ***root***. Mật khẩu của *user root* bắt buộc có chiều dài tối thiểu của password là 6 ký tự. Bạn nên đặt password gồm có ký tự, số và các ký tự đặc biệt để đảm bảo an toàn. Lưu ý password phân biệt chữ hoa và thường. Bạn phải đánh vào 2 lần, khi dòng chữ bên dưới xuất hiện "***Root password accepted***" thì được.

Hình 2.16

➢ **Lựa chọn các gói cài đặt**

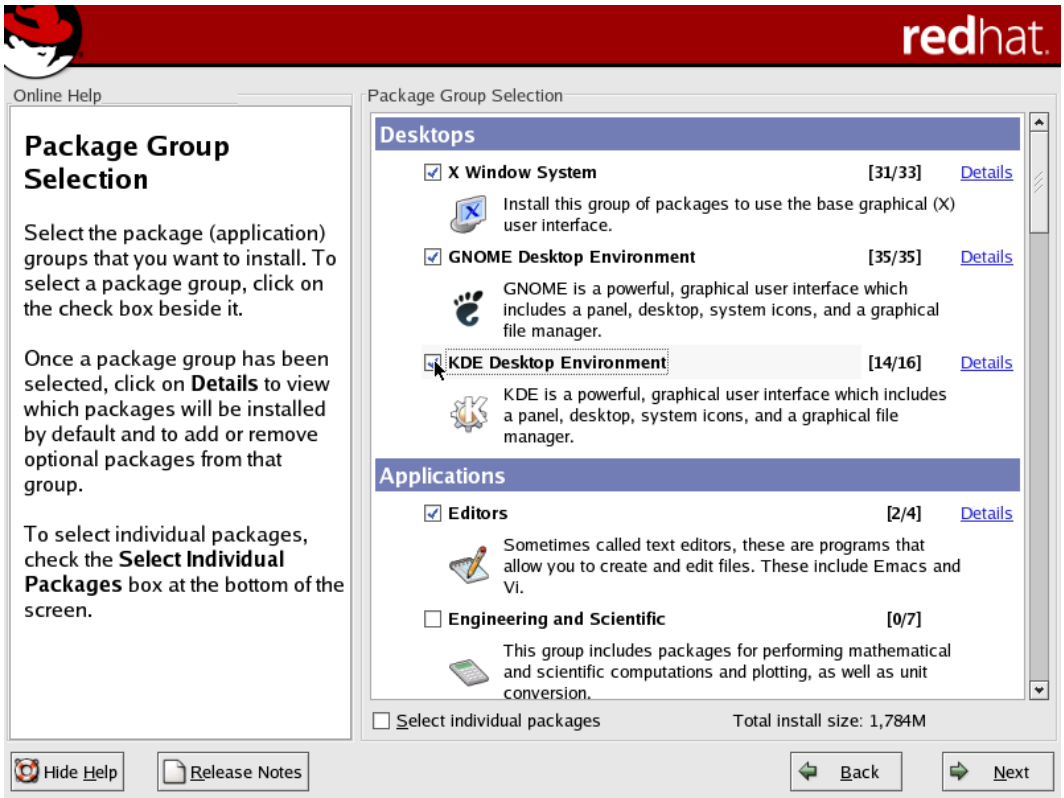
Bạn chọn các chương trình cần cài đặt, nếu chọn ***everything*** là cài tất cả các chương trình, chọn ***Minimal*** là chỉ cài một số chương trình hoặc phần mềm thông dụng.

Nếu bạn nắm rõ các package cần thiết cho các chương trình mình mong muốn thì chọn ***Select individual packages***.



Hình 2.17

Ta có thể chọn ***Details*** để chọn chi tiết các thành phần trong từng phần mềm hoặc nhóm các công cụ.

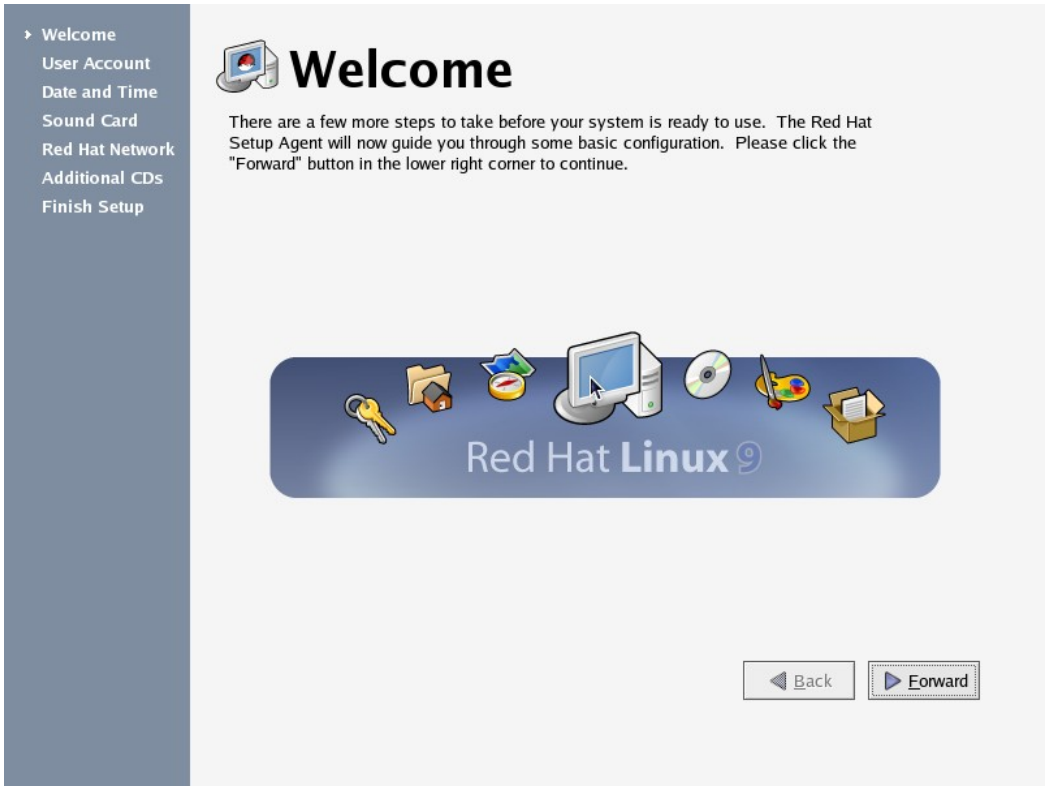


Hình 2.18

➢ **Quá trình cài đặt các gói *(Hình 2.19)***

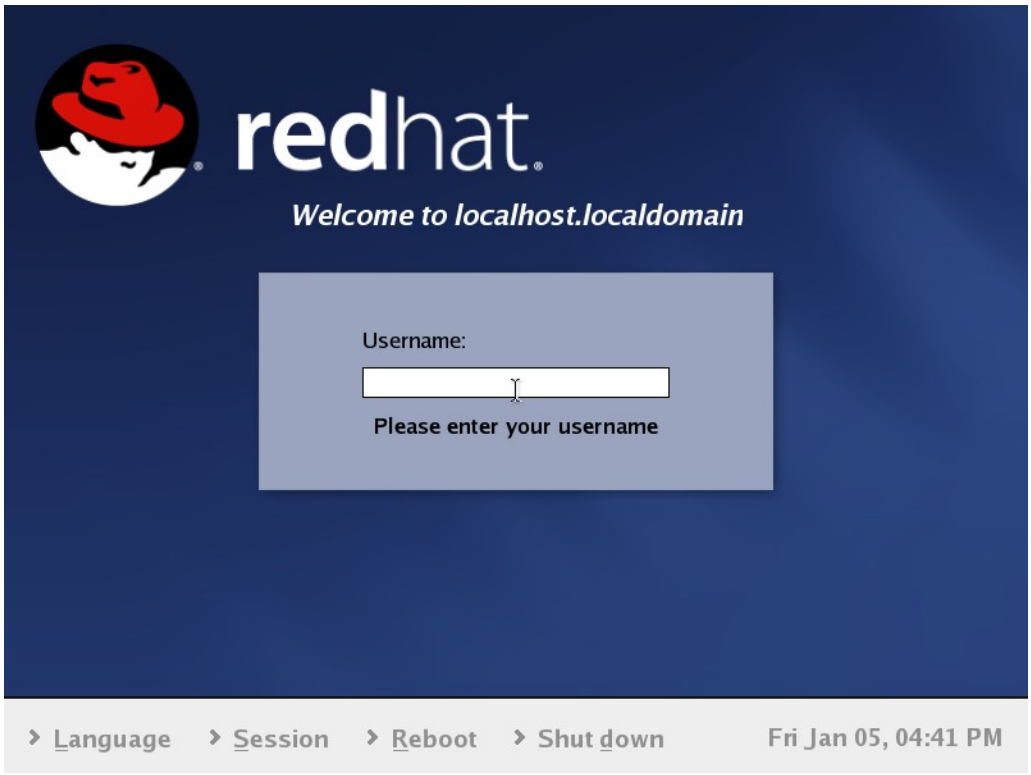
Hình 2.19

➢ **Màn hình sau khi cài đặt xong và khởi động lần đầu tiên**



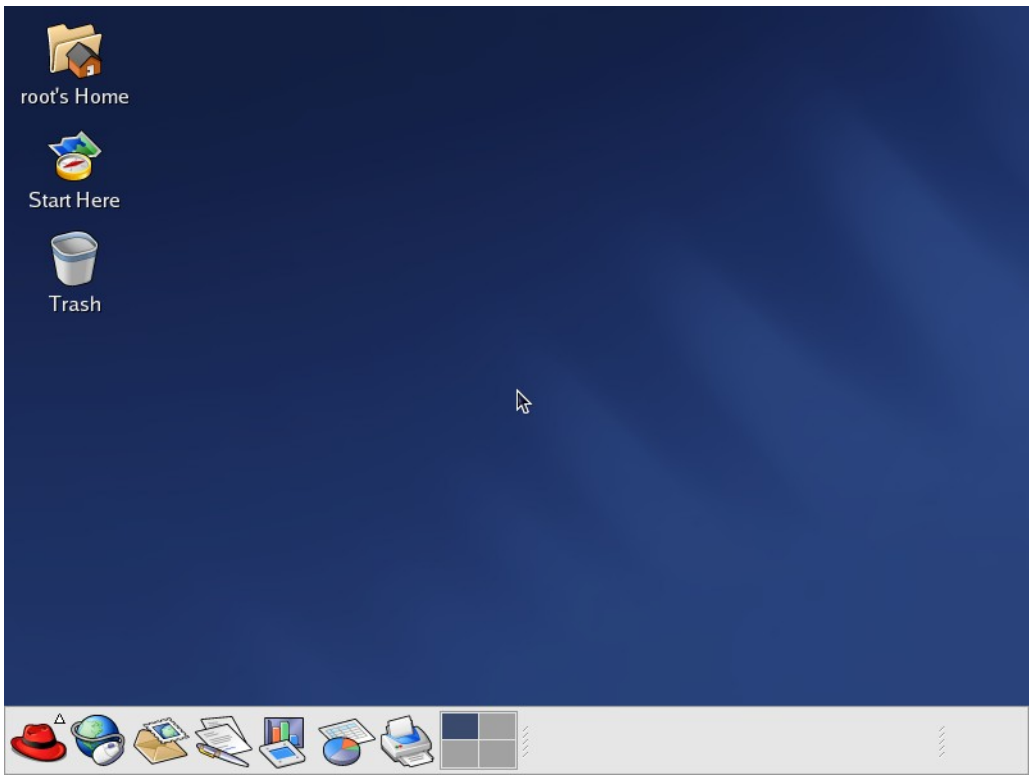
Hình 2.20

➢ **Màn hình login vào hệ thống**



Hình 2.21

➢ **Màn hình của hệ điều hành Red hat linux 9.0**

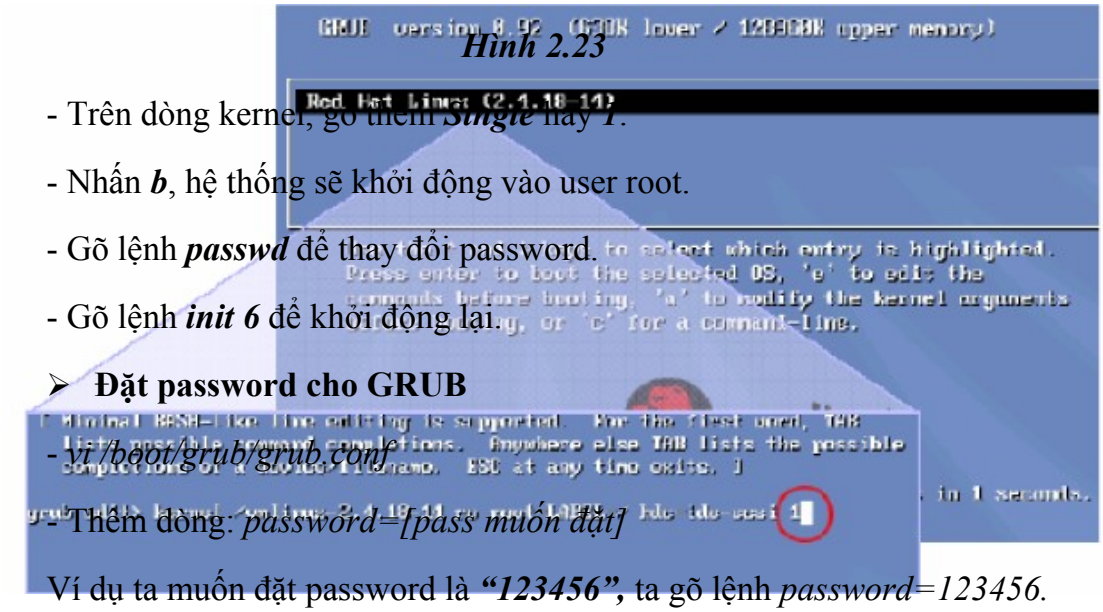


Hình 2.22

II.2.2 Thay đổi password cho root, GRUB

➢ **Thay đổi password cho root khi quên**

- Lúc khởi động vào GRUB nhấn e, để thay đổi tùy chọn khi khởi động.



Hình 2.23

- Trên dòng kernel, gõ thêm ***Single*** hay ***1***.

- Nhấn ***b***, hệ thống sẽ khởi động vào user root.

- Gõ lệnh ***passwd*** để thay đổi password.

- Gõ lệnh ***init 6*** để khởi động lại.

➢ **Đặt password cho GRUB**

- *vi /boot/grub/grub.conf*

- Thêm dòng: *password=[pass muốn đặt]*

Ví dụ ta muốn đặt password là ***"123456",*** ta gõ lệnh *password=123456.*

II.3 Câu hỏi và bài tập

Câu 1:

Hãy cài đặt hệ điều hành Red Hat Linux 9.0 trên máy tính cá nhân của mình. Trong quá trình cài đặt tạo account ***tttccn***.

Câu 2:

Hãy đăng nhập vào hệ thống Linux bạn vừa cài đặt.

- Làm quen với giao diện và môi trường làm việc của hệ điều hành Red Hat Linux 9.0.

- Thực hiện một số thao tác như chỉnh sửa giao diện desktop, tạo thêm một số thư mục mới, ...

- Thoát khỏi hệ thống.

Câu 3:

Hãy thay đổi password của người dùng ***root*** và đặt password cho ***GRUB***.

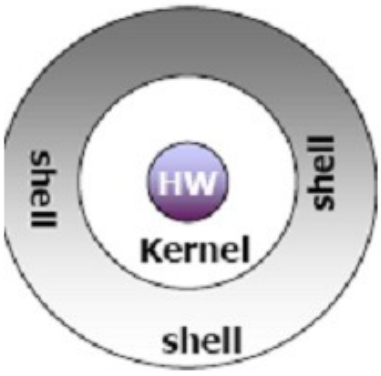
CHƯƠNG III. CHẾ ĐỘ DÒNG LỆNH TRÊN LINUX

III.1 Giới thiệu về sử dụng lệnh trong Linux

Như đã giới thiệu ở phần trên, Linux là một hệ điều hành đa người dùng, đa nhiệm, được phát triển bởi hàng nghìn chuyên gia tin học trên toàn thế giới nên hệ thống lệnh cũng ngày càng phong phú. Đến thời điểm hiện nay Linux có khoảng hơn một nghìn lệnh. Tuy nhiên chỉ có vài chục lệnh là thông dụng nhất đối với người dùng. Chúng ta đừng e ngại số lượng lệnh được giới thiệu chỉ chiếm một phần nhỏ trong tập hợp lệnh bởi vì đây là những lệnh thông dụng nhất và chúng cung cấp một phạm vi ứng dụng rộng lớn, đủ thỏa mãn yêu cầu của chúng ta.

III.1.1. Giới thiệu Shell

Cơ chế dòng lệnh là cách cơ bản nhất để tương tác với hệ thống máy tính. Shell cung cấp một giao diện giữa nhân và người sử dụng.



Hình 3.1

Shell nhận lệnh từ người sử dụng, phân tích lệnh và gửi lệnh tới nhân để thực thi.

Shell không chỉ có chức năng nhận và thông dịch lệnh, nó còn cung cấp một môi trường để có thể cấu hình hệ thống và lập trình.

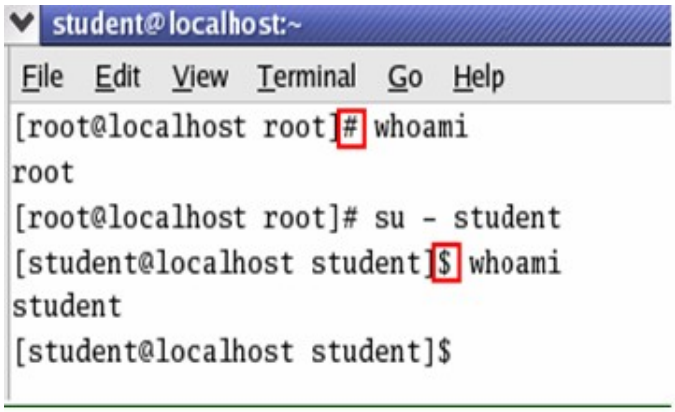
Giao diện của shell thường có một dấu nhắc mà tại đó bạn sẽ nhập lệnh vào. Giao diện này được gọi là giao diện dòng lệnh (*command line interface*).

Có nhiều loại shell. Shell mặc định trên Linux là *BASH*.

C shell	/bin/csh
Bourne shell	/bin/sh
Korn shell	/bin/ksh
Tom's C shell	/bin/tcsh
Bourne Again shell	/bin/bash
...	

Shell thường kết thúc bằng:

- $: là người sử dụng thông thường
- #: người dùng là Root (Administrator)



Hình 3.2

III.1.2 Sử dụng lệnh

Cú pháp lệnh: trong một lệnh thường có 3 thành phần chính:

Command <Options> {Arguments}

- ***Command**-lệnh:* cho biết hệ thống cần làm gì.
- ***Options**-tùy chọn:* làm như thế nào.
- ***Arguments**-tham số lệnh:* nơi lệnh được áp dụng.

Đôi khi không cần đến <*Options*> và <*Arguments*>, điều này phụ thuộc vào từng lệnh.

Ví dụ về cú pháp lệnh:

Xem lệnh trước: dùng phím *UP* và *DOWN* (các lệnh trước được lưu trong *~/.bash_history*).

Auto Complete: khả năng hoàn chỉnh lệnh hay tên file.

Trên dòng lệnh, khi nhập tên lệnh hoặc tên file nhưng chưa đầy đủ bạn có thể ấn phím ***TAB*** và shell sẽ tự điền nốt phần tên còn lại.

Ví dụ: Giả sử đã có file *document.*

$ cat doc (gõ phím TAB)

*$ cat doc**ument** (shell tự điền nốt phần còn lại)*

Dừng một lệnh đang thực hiện: *Ctrl+C, Ctrl+Z*

Kết thúc phiên làm việc shell:

- *^D (Ctrl+D)*
- *exit*
- *Logout*

Lưu ý: Linux phân biệt ký tự hoa - thường.

Wildcard Character-Kí tự đại diện:

- *: đại diện cho không hoặc nhiều kí tự bất kì.
- ?: đại diện cho một kí tự duy nhất.
- [...] tương ứng một trong các ký tự bên trong ngoặc.
- [!/^] không tương ứng với một trong các ký tự bên trong ngoặc.
- \ loại bỏ ý nghĩa đặc biệt của các ký tự *,?,).

Ví dụ mở rộng ký tự thay thế:

- *ls a** : liệt kê tất cả các tên bắt đầu bằng "a".
- *ls a?.txt* : liệt kê tất cả tên dạng a?.txt với ? là ký tự bất kỳ.
- *ls [aei]** : liệt kê tất cả các tên bắt đầu bằng a, e, hoặc i.
- *ls [a-d]*[0-9]* : liệt kê tất cả tên bắt đầu từ a đến d và kết thúc từ 0 đến 9.
- *ls [!L-T]** : liệt kê tất cả các tên không bắt đầu từ L đến T.

III.2 Các lệnh cơ bản

➢ **Tìm trợ giúp về lệnh:**

✓ Sử dụng lệnh *info* để xem các thông tin về lệnh.

- $ info command

Ví dụ ta muốn xem thông tin của lệnh ***date, mount*** tại cửa sổ lệnh ta gõ:

- *$ info date*
- *$ info mount*

✓ Sử dụng lệnh *man* để xem các hướng dẫn về lệnh.

- *$ man command*
- *$ man –k keyword*

✓ Duyệt các man page:

- ***spacebar***: chuyển sang trang kế.
- ***b***: chuyển về trang trước.
- ***q***: (quit) thoát ra khỏi lệnh.
- ***/keyword*** tìm trong nội dung man page.

Ví dụ: Tra cứu lệnh *ls*

- *$ man ls*

Để thoát khỏi chế độ tra cứu bấm phím ***q***.

➢ **Lệnh hiển thị runlevel đang chạy**

#runlevel

Thay đổi runlevel

#init <newrunlevel>

➢ **Shutdown và reboot lại hệ thống**

➢ **Lịch sử lệnh**

– BASH có thể lưu trữ một số lượng lớn các câu lệnh đã được thực hiện gần nhất.

– Các lệnh này được đánh số từ 1 tới tối đa là 500.

– Để hiển thị danh sách lệnh đó, ta dùng lệnh *history*.

– Nếu muốn dùng lại một lệnh trong danh sách, có thể gõ số hiệu của lệnh đó ngay sau dấu **!**.

Ví dụ:

$!25

➢ ***alias*: Đặt bí danh cho câu lệnh**

– Đôi khi một lệnh dài hoặc khó nhớ lại cần được dùng thường xuyên, trong trường hợp này bạn có thể đặt bí danh cho lệnh đó thông qua lệnh *alias*.

– Lệnh *alias* không làm thay đổi tên lệnh mà chỉ đặt một tên khác cho lệnh. Có thể dùng *alias* để đặt bí danh cho một lệnh có kèm tùy chọn của câu lệnh đó.

$ alias 'list=ls'

$ list

mydata today

– Để xem tất cả các bí danh lệnh hiện có, gõ lệnh *alias* không có đối số

➢ ***pwd***: Cho biết thư mục đang làm việc

Ví dụ:

- [student]$ pwd

- /home/student

➢ **Lệnh xem và thiết lập ngày giờ**

Xem ngày giờ:

date [tùy chọn] … [+định dạng]

Thiết lập thời gian:

date [tùy chọn] [MMDDhhmm[[CC]YY][.ss]]

Tùy chọn như sau:

-d, --date = xâu-văn-bản: hiển thị thời gian dưới dạng xâu-văn-bản, mà không lấy thời gian hiện tại của hệ thống, xâu văn bản được đặt trong 2 dấu nháy kép hoặc nháy đơn.

Ví dụ: hiển thị chuỗi *'20/10/2006'* dưới dạng thời gian như sau

#date –d '20/10/2006'

Fri Aug 10 00:00:00 EDT 2007

***-f, --file = Tên-File*:** giống như tham số *-d* nhưng sẽ hiển thị nhiều thời gian, ứng với mỗi dòng trong file được xem như một xâu văn bản.

Ví dụ: giả sử file *stringdate* có nội dung như sau:

10/10/2006

20/4/2007

Dùng lệnh sau để kiểm tra kết quả:

#date –f 'stringdate'

-r, --reference= tập-tin: Hiển thị thời gian sửa đổi tập-tin gần nhất.

Ví dụ: xem thời gian sửa đổi tập tin *stringdate*:

#date –r 'stringdate'

-s, --set=xâu-văn-bản: thiết lập lại thời gian theo kiểu xâu văn bản.

Ví dụ: thiết lập thời gian của hệ thống là *'09:00:00 1/1/2007'*

- *#date –s '9:00:00 1/1/2007'*
- *Mon Jan 1 9:00:00 EST 2007*

- **Lệnh *cal* cho phép xem lịch trên hệ thống**

cal [tùy-chọn] [<tháng>] [<năm>]

Nếu không có tham số, hiển thị lịch của tháng hiện tại.

Tùy-chọn:

- ***m:*** chọn ngày Thứ 2 là ngày đầu tiên trong tuần (mặc định là ngày Chủ nhật).

- ***j*:** hiển thị ngày trong tháng dưới dạng số ngày trong năm.

- ***y:*** hiển thị lịch của năm hiện thời.

➢ **Điều hướng dòng ra chuẩn STDOUT: >**

Bằng cách dùng toán tử >>, dữ liệu từ STDOUT sẽ được ghi thêm vào cuối file điều hướng thay vì ghi đè lên nó.

Ví dụ:

$ls > ketqua

Kết quả của lệnh *ls* được ghi đè lên nội dung file *ketqua*

$ date >> ketqua

Kết quả của lệnh *ls* được ghi thêm vào file *ketqua*

➢ ***Piping***

- Cú pháp: 

- Sử dụng ký tự '|'.

- Kết quả của **command1** là đầu vào của **command2**.

Ví dụ:

- *ls –l|tail -3*

Hiển thị 3 dòng cuối kết quả do lệnh *ls –l* trả về.

- *ls|head -2*

Hiển thị 2 dòng đầu của kết quả do lệnh *ls* trả về.

Ví dụ: kết quả của lệnh "*ls –l*" là đầu vào cho lệnh "*grep samba*"

III.3 Câu hỏi và bài tập

Câu 1:

- Hãy cho 10 ví dụ về câu lệnh trong Linux.

- Với mỗi câu lệnh hãy chỉ rõ từng thành phần trong cú pháp của một câu lệnh mà bạn đã được học.

- Cho biết các câu lệnh đó dùng để làm gì?

Câu 2:

- Mở cửa sổ gõ lệnh (terminal window) và dùng lệnh để xem thông tin về các lệnh sau: ***ls, echo, whoami, cat, sort***.

- Sau khi xem thông tin hãy cho biết các lệnh trên được dùng để làm gì?

Câu 3:

Ở chế độ dòng lệnh bạn hãy cho biết thư mục bạn đang làm việc? Lịch sử các lệnh đã sử dụng và thử gọi lại 1 lệnh trước đó đã dùng.

Câu 4:

Dùng lệnh để xem ngày giờ của hệ thống. So sánh với ngày giờ hiện tại, nếu thấy sai hãy viết lệnh thiết lập lại ngày giờ cho đúng.

Câu 5:

Hãy viết các câu lệnh để:

- Xem lịch của tháng *12/2010*.

- Xem lịch của năm hiện thời rồi lưu kết quả vào file *lich2008*.

- Ghi tiếp vào nội dung của file *lich2008* kết quả của câu lệnh *echo* "*Bao gio thi duoc nghi tet?"*.

Câu 6:

- Bạn thấy việc học cách sử dụng chế độ dòng lệnh để giao tiếp với hệ thống có cần thiết không khi các phiên bản hiện nay của hệ điều hành Linux đã có tiện ích đồ họa? Giải thích vì sao?

Câu 7:

- Hãy so sánh cơ chế dòng lệnh của hệ điều hành Linux và hệ điều hành MS-DOS.

CHƯƠNG IV: QUẢN LÝ THIẾT BỊ VÀ HỆ THỐNG TẬP TIN/THƯ MỤC

IV.1 Quản lý thiết bị

IV.1.1 Quy tắc quản lý thiết bị

Linux xây dựng cơ chế truy xuất đến tất cả các loại đĩa và thiết bị đều ở dạng tập tin (tập tin thiết bị) và lưu trong thư mục */dev*.

Linux quy ước đặt tên như sau:

- Ổ đĩa mềm: *fd*
- Ổ đĩa cứng vật lý thứ nhất: *hda*
- Ổ đĩa cứng vật lý tứ hai: *hdb*
- …

Nếu đĩa cứng theo tiêu chuẩn SCSI thì gọi là: *sda, sdb,…*

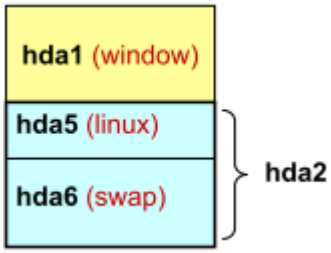
Các thiết bị USB, Linux xem như là thiết bị SCSI (ví dụ nếu máy có một đĩa cứng SCSI thì USB sẽ là *sdb1*).

Các phân vùng (partitions) được đánh số sau tên đĩa. Ví dụ: *hda1, hda2, sda1, sdb1 (ổ A), fd1 (ổ B) …*

Các phân vùng chính (*primary*) hoặc phân vùng mở rộng (*extended*) được đánh từ 1 -> 4.

Các phân vùng logic (nằm trong phân vùng mở rộng) đánh số từ 5 trở đi.

Ví dụ phân vùng đĩa cứng IDE:



Giải thích:

- *hda1*: phân vùng chính.

- *hda2*: phân vùng mở rộng.

- *hda5*: phân vùng lôgic.

- *hda6*: phân vùng lôgic.

Chú ý: nếu khi cài đặt Linux mà trước đó đã cài Window, thì Linux sẽ tự động cài đặt vào các phân vùng mở rộng.

IV.1.2 Cách truy xuất đĩa

Cũng tương tự như Window, trong Linux cũng có khái niệm đường dẫn (*path*). Tuy nhiên, có 2 điểm cần lưu ý:

- Thứ nhất, sử dụng ký tự sổ trái (/) làm ký tự phân cách thư mục và tập tin.

- Thứ hai, không sử dụng ký tự ổ đĩa, mà dùng ký tự / ở đầu đường dẫn (thư mục gốc).

Ví dụ:

- */usr/local/dev*

- */dev/hda*

Khi khởi động hệ điều hành, Linux chỉ kết gắn cho phân vùng chính (nơi chứa nhân Linux) bằng ký tự "**/**" (thư mục gốc).

Các thông tin của phân vùng khác được Linux đặt trong thư mục */dev* của phân vùng chính.

Như vậy mặc dù tất cả các file trong Linux đều được đặt trong cùng một cây thư mục, song chúng có thể được lưu trữ trên các bộ nhớ ngoài khác nhau như đĩa cứng hay CD-ROM. Mỗi thiết bị nhớ cũng có thể có hệ thống file (*file system*) khác nhau như *FAT, NTFS, ext2*, …

Để gắn một hệ thống file trên thiết bị lưu trữ vào cây thư mục chính ta dùng lệnh ***mount***.

IV.1.3 Các lệnh quản lý thiết bị ngoại vi

- **Lệnh *mount*: Ghép nối thiết bị vào cây thư mục**

Chú ý là thao tác *mount* chỉ thực hiện được nếu bạn là người dùng có quyền cao nhất (**root)**.

Cú pháp của lệnh *mount* như sau:

#mount thiết_bị_cần_mount điểm_nối_vào_hệ_thống_file

Ví dụ 1: Mount và sử dụng đĩa mềm:

 # mount /dev/fd0 /mnt/floppy

Trong lệnh trên, hệ thống sẽ kết nối đĩa mềm *fd0* vào cây thư mục tại điểm nối là */mnt/floppy*. Từ đó bạn có thể vào */mnt/floppy* để truy nhập nội dung ổ đĩa A.

Ví dụ 2: Mount và sử dụng ổ CD:

 # mount /dev/cdrom /mnt/cdrom

Ví dụ 3: Mount và sử dụng USB:

 # mount /dev/sdb1 /mnt/usb

- **Lệnh *umount*: Gỡ bỏ kết nối**

Để gỡ bỏ kết nối với một hệ thống file, ta dùng lệnh *umount* như sau:

 # umount thiết_bị_đã_mount điểm_nối_vào_hệ_thống_file

Ví dụ: Gỡ kết nối với đĩa mềm

 # umount /dev/fd0 /mnt/floppy

Tất cả các hệ thống file cần phải được *mount* trước khi truy nhập và phải được *umount* khi đóng hệ thống.

Tuy nhiên Linux sẽ tự động *mount* một số thiết bị cho bạn khi khởi động và các thiết bị này cũng sẽ tự động được *umount* khi đóng hệ thống.

- **Lệnh *du:* xem dụng lượng đĩa đã dùng:**

du <tùy-chọn> tênthưmục-Hoặc-têntậptin

Ví dụ: để xem thông tin về dung lượng đĩa đã dùng trong thư mục '*laptrinhc_linux*' ta gõ lệnh:

#du laptrinhc_linux

Các tùy-chọn:

***-a*:** liệt kê kích thước của tất cả các tập tin, thư mục trong thư mục cần coi.

***-b, --bytes*:** hiển thị kích thước theo byte.

***-c, --total*:** hiển thị cả tổng dung lượng được sử dụng trong hệ thống tập tin.

***-h, --human-readable*:** hiển thị kích thước các tập tin kèm theo đơn vị tính (ví dụ: *1K, 234M, 2G ...*).

***-k, --kilobytes*:** hiển thị kích thước tính theo kilobytes.

***-m, --megabytes*:** tính kích thước theo megabytes.

***-s*:** đưa ra kích thước của hệ thống tập tin/thư mục mà không hiển thị kích thước của thư mục con.

- **Lệnh *df*: kiểm tra dung lượng đĩa trống**

df <tùy-chọn> tênthưmục-Hoặc-têntậptin

Ví dụ:

 # df /mnt/floppy

Filesystem	**1k-blocks**	**Used**	**Available**	**Use%**	**Mounted on**
/dev/fd0	1423	249	1174	18%	/mnt/floppy

IV.2 Hệ thống tập tin/ thư mục

IV.2.1 Cấu trúc hệ thống tập tin/ thư mục

Hệ thống tập tin dùng để lưu trữ các tập tin theo một cấu trúc có tổ chức.

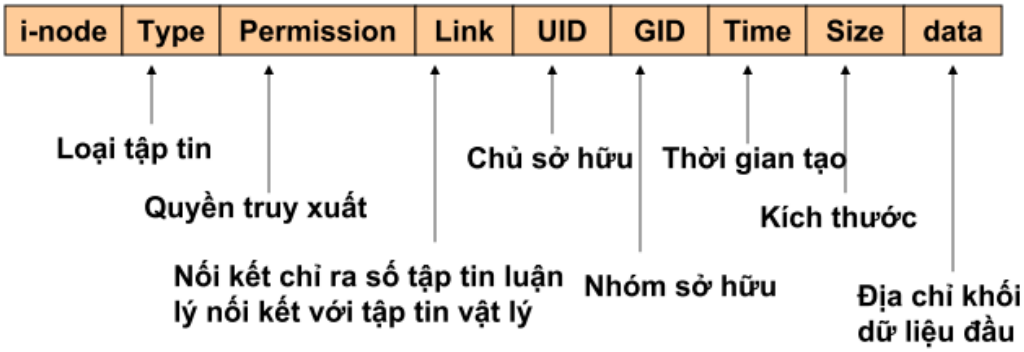
Hệ thống tập tin/thư mục được tạo trên phân vùng của Linux.

Linux hỗ trợ nhiều files system: *ext2, ext3, vfat, NTFS,…*

Trong mỗi file system, việc ánh xạ từ tên qua các khối dữ liệu được thực hiện thông qua cấu trúc gọi là *i-node*.

Cấu trúc thư mục ánh xạ tên và số *i-node*. Các phần tử của thư mục có dạng:

Mỗi *i-node* mô tả một file. Mỗi *i-node* chứa một danh sách các khối (block) của tập tin mà nó mô tả.



i-node không chứa tên tập tin. Tên tập tin ở trong cấu trúc thư mục (*directory structure*).

Người sử dụng khi truy xuất các tập tin trong Linux bị kiểm soát bởi "*quyền truy cập*" (trường **permission** trong *i-node*).

Linux có 4 kiểu file cơ bản:

- File thông thường (*program, text, library*, ...).
- Thư mục (*container*).
- File đặc biệt (*device, socket, pipe,* ...).
- Liên kết symbolic links (*symlinks*).

Trong Linux một thư mục là một tập tin chứa danh sách của tất cả các tập tin và thư mục con của thư mục đó).

Quy ước đặt tên file:

- Tối đa 225 ký tự.
- Có thể sử dụng bất kỳ ký tự nào (kể cả các ký tự đặc biệt).

"very ? long - file + name.test"

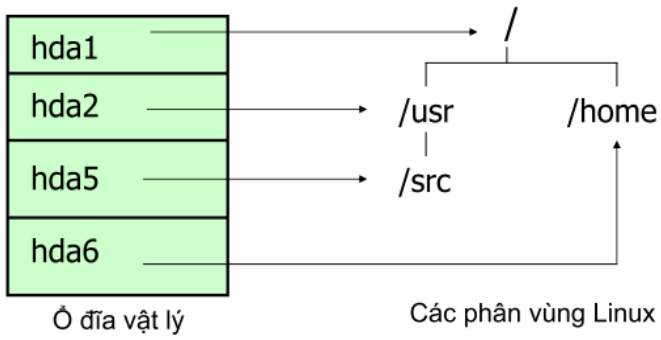
- File/thư mục ẩn được bắt đầu bằng một dấu chấm ".".

.bash_history .bash_profile .bashrc

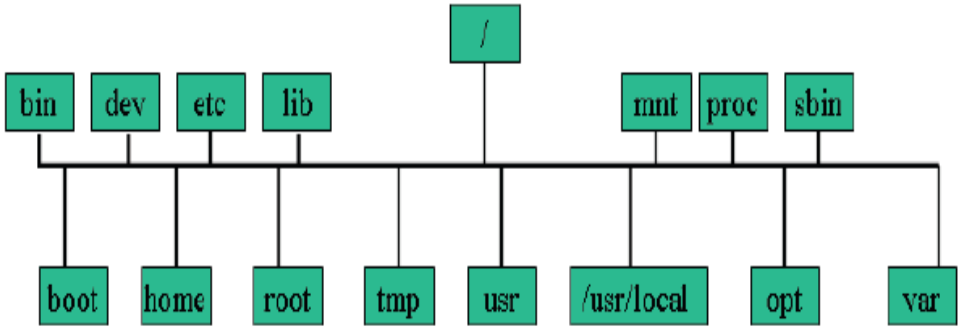
.desktop/ .kde/ .mozilla/

Cấu trúc cây thư mục:

Trong Linux không có khái niệm ổ đĩa. Sau quá trình khởi động, toàn bộ các thư mục và tập tin được kết gắn và tạo thành một hệ thống tập tin thống nhất, bắt đầu từ gốc "/".



Các thư mục hệ thống:



❖ **Thư mục *home*:** thư mục dữ liệu người dùng.

- Trong thư mục này có các thư mục con dành cho từng người dùng.

- Mỗi người dùng được phép tạo, cập nhật, xoá dữ liệu trong thư mục của mình.

- Khi bạn đăng nhập vào hệ thống thì bạn sẽ được đặt vào thư mục của bạn, thư mục này có tên chính là tên người dùng mà bạn đăng nhập.

❖ **Thư mục */bin*:** các lệnh cơ bản.

- Chứa các lệnh và các chương trình tiện ích chuẩn, chẳng hạn như *ls, cat* ...

❖ **Thư mục */usr*:** ứng dụng và thư viện.

Chứa các file và lệnh chuẩn sử dụng bởi hệ thống. Nó được chia thành nhiều thư mục con:

- */usr/bin*: chứa các lệnh và tiện ích hướng người dùng.

- /usr/sbin: chứa các lệnh quản trị hệ thống.

- */usr/lib*: chứa các thư viện dành cho lập trình.

- */usr/doc*: chứa tài liệu Linux.

- */usr/man*: chứa các file tra cứu lệnh.

- */usr/spool*: chứa các file được sinh ra bởi lệnh in hoặc truyền tin qua mạng.

❖ **Thư mục */sbin*:** các lệnh quản trị

- Chứa các lệnh quản trị dùng khi khởi động hệ thống.

❖ **Thư mục */var*:** dữ liệu biến động

- Chứa nhiều file khác nhau, chẳng hạn như các file thư (mailbox).

❖ **Thư mục */dev*:** khai báo thiết bị

- Chứa các giao diện cho thiết bị như terminal hay máy in.

❖ **Thư mục */etc*:** cấu hình hệ thống và ứng dụng

- Chứa các file cấu hình hệ thống và các file hệ thống khác.

❖ **Thư mục */boot*:** kernel và cấu hình boot.

❖ **Thư mục */lib*:** thư viện dùng chung (shared lib).

❖ **Thư mục */mnt*:** thư mục để mount floppy, cdrom, ...

❖ **Thư mục */proc*:** thông tin process (pseudo-filesystem).

❖ **Thư mục */tmp*:** dữ liệu tạm.

IV.2.2 Một số lệnh thao tác trên tập tin/ thư mục

Đường dẫn (*path*) là một dãy kí tự để xác định vị trí của tập tin hoặc thư mục.

Khi đăng nhập vào hệ thống Linux, mặc định được đặt trong thư mục có tên là tên tài khoản truy nhập của bạn (thư mục đăng nhập) nằm trong thư mục /*home*.

Để biết đường dẫn tới thư mục hiện thời, ta dùng lệnh ***pwd.***

Đường dẫn tuyệt đối (*Absolute Path Name*): chỉ rõ file và thư mục trong mối liên hệ với toàn bộ cây thư mục. Đường dẫn tuyệt đối luôn luôn bắt đầu với thư mục gốc (/).

Ví dụ:

```
# pwd
/home/cv_user
```

 # cd /etc/rc1.d ← ***Absolute Path***

 # pwd

/etc/rc1.d

Đường dẫn tương đối (*Relative path name*): mô tả vị trí của file và thư mục trong mối liên hệ với thư mục hiện tại. Đường dẫn tương đối không bao giờ bắt đầu với dấu "/".

Ví dụ:

 # pwd

/home/lpiuser

 # cd /etc

 # pwd

/etc

 # cd rc1.d ← ***Relative path***

Một số đường dẫn đặc biệt:

.	Thư mục hiện tại
..	Thư mục cha thư mục hiện tại
~	Thư mục chủ của người dùng (home directory)
~-	Đường dẫn đầy đủ của thư mục làm việc trước
~logname	Thư mục chủ của người dùng có tên là logname

- **Lệnh *mkdir:* tạo thư mục.**

Đối số của lệnh là tên thư mục cần tạo.

Ví dụ:

- Tạo thư mục *hoctap* trong thư mục *quynh* của thư mục home nằm trong thư mục gốc.

$ mkdir /home/quynh/hoctap/

- Tạo thư mục *week1* trong thư mục đăng nhập.

$ mkdir ~/week1/

- Tạo thư mục *week2* trong thư mục hiện thời.

$ mkdir week2

Tùy chọn: *-p* tạo thư mục cha nếu chưa tồn tại.

Ví dụ:

$ mkdir –p /home/sinhvien/thuchanh/

- **Lệnh *ls:* hiển thị nội dung thư mục**

Lệnh ***ls*** dùng để hiển thị các file và thư mục con trong một thư mục nào đó. Để có sự phân biệt giữa file và thư mục, ta dùng tùy chọn -**F**, khi đó những file là kiểu thư mục sẽ có thêm dấu / ở ngay sau tên đó ***$ ls –F***.

Có nhiều tuỳ chọn với ***ls*** để xem các thông tin khác nhau:

-a: hiển thị tất cả các file kể cả file ẩn (file hệ thống).

-l: hiển thị tất cả thông tin về file.

-i: hiển thị chỉ số inode của file.

Ví dụ:

$ ls -l datafile

Kết quả của lệnh trên như sau:

- r w - r - - r - -1 tinhoc 248 Jun 2 9:30 datafile

- **Lệnh *cd:* chuyển thư mục**

Lệnh ***cd*** cho phép thay đổi thư mục làm việc từ thư mục này sang thư mục khác. Đường dẫn có thể là tuyệt đối hoặc tương đối.

Ví dụ:

$ cd /home/sinhvien/thuchanh/

Sau câu lệnh ***cd*** ở trên, thư mục làm việc sẽ chuyển sang thư mục mới là */home/sinhvien/thuchanh*.

Lưu ý:

- *cd* chuyển đến thư mục home (thư mục của user).

- *cd* ~ chuyển đến thư mục home (thư mục của user).

- *cd ..* chuyển đến thư mục cha.

- ***cd ~user*** chuyển đến thư mục home của "user".

- ***cd path*** chuyển đến thư mục path.

- **Sao chép file: lệnh *cp***

Để copy file ta dùng lệnh ***cp*** với đối số thứ nhất là file gốc cần copy (file nguồn), đối số thứ hai là file bản sao (file đích). Copy một hoặc nhiều file đến file hoặc thư mục khác.

 # cp [options] file1 file2

 # cp [options] files directory

Để tránh việc ghi đè lên tên file đích đã tồn tại, thêm tuỳ chọn ***–i***. Khi đó shell sẽ hỏi để xác nhận sự ghi đè trước khi copy.

Ví dụ:

$ cp -i newdata datafile

Overwrite datafile? **n**

- **chuyển file: lệnh *mv***

Lệnh ***mv*** dùng giống như ***cp***. Song thay vì tạo ra một bản sao mới, ***mv*** sẽ di chuyển hẳn file gốc và do vậy không còn file gốc tại nơi ban đầu. Hoặc ta có thể dùng lệnh ***mv*** để đổi tên file.

Cú pháp: ***mv [options] source target***

Tùy chọn:

-f: ép buộc di chuyển nếu đường dẫn đích đã tồn tại.

-i: xác nhận trước khi di chuyển.

Ví dụ:

$ mv week1 tam

Câu lệnh trên sẽ đổi tên thư mục *week1* thành *tam*.

$ mv rootdate week2 tam

Câu lệnh trên di chuyển *rootdate* và *week2* vào thư mục *tam*.

- **Xoá file: lệnh *rm***

Lệnh ***rm*** cho phép xoá file, đối số của nó là tên các file cần xoá.

Cú pháp: ***rm [options] files***

Tùy chọn:

-i: xác nhận trước khi xóa.

-r, -R: xóa đệ qui.

-d: xóa thư mục nếu không rỗng.

Ví dụ:

Xoá file *rootdate* trong thư mục *tam*.

$ rm tam/rootdate

Xoá thư mục *moi* và toàn bộ nội dung bên trong.

$ rm -r moi

- **Lệnh *rmdir*:** xóa thư mục rỗng.

Cú pháp: ***rmdir [option] directories***

Tùy chọn: -p: xóa thư mục cha.

- **Lệnh *touch***: tạo tập tin không có nội dung.

Cú pháp: ***touch files***

- **Lệnh *wc***: đếm số dòng, từ, kí tự,… của một tập tin

Cú pháp: ***wc [options] files***

Tùy chọn:

-c: đếm số kí tự.

-l: đếm số dòng.

-w: đếm số từ.

- **Lệnh *cat*:** Hiển thị toàn bộ nội dung của một file

Cú pháp: ***cat [option] file***

Ví dụ:

Hiển thị nội dung file *mydata*.

$ cat mydata

- **Lệnh *more*:**

Cú pháp: ***more [option] file***

Shell cung cấp lệnh ***more*** cho phép kiểm soát, giới hạn nội dung hiển thị ra màn hình từng phần một.

Xem tiếp hoặc xem lại phần nội dung phía trước một cách dễ dàng. Dùng phím ***Space bar*** để xem trang tiếp theo, phím ***Enter*** để xem dòng tiếp theo, phím ***b*** để xem lại trang trước và phím ***q*** để thoát.

IV.2.3 Đặt quyền trên tập tin/ thư mục

Mỗi một file và thư mục trong Linux đều chứa tập các quyền xác định tài khoản nào có thể truy nhập.

Có ba kiểu quyền truy nhập trên file:

- Đọc (*read - r*).

- Ghi (*write - w*).

- Thực thi (*executable - x*).

Khi một file được tạo ra thì tự động người tạo có quyền đọc và ghi cho phép xem và sửa file.

Có ba kiểu người dùng trên file:

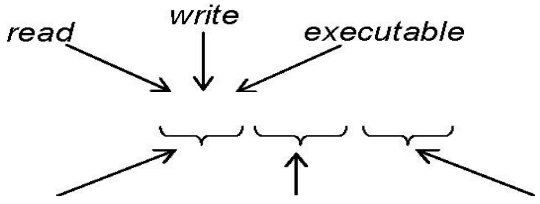
- Người sở hữu file (*owner - u*) là người tạo ra file.

- Nhóm người dùng file (*group - g*) (thường là những người cùng nhóm với người sở hữu).

- Những người dùng khác (*other - o*).

Như vậy bạn có thể thiết lập quyền truy nhập file cho từng đối tượng cụ thể.

Khi liệt kê chi tiết file bằng lệnh ***ls –l***, ta thấy quyền trên file gồm 9 ký tự như sau:



Định danh quyền truy cập:

-u user: chủ sở hữu file.

-g group: nhóm có user là thành viên.

-o others: các user khác trên hệ thống.

-a all: tất cả user (***u, g*** và ***o***).

Tác vụ trên quyền truy cập:

– **"+"** thêm quyền.

– **"-"** loại bỏ quyền.

– **"="** gán quyền.

❖ **Đặt quyền bằng ký hiệu quyền**

Với ký hiệu quyền và ký hiệu người dùng ở trên, ta có thể thiết đặt quyền bằng ký hiệu quyền như sau:

Cú pháp:

$ chmod ***kiểu_người_dùng+quyền_thêm_vào*** **tên_file**

$ chmod ***kiểu_người_dùng-quyền_bớt_đi*** **tên_file**

Ví dụ:

$ chmod u+x filename

$ chmod o+r-wx filename

❖ **Đặt quyền tuyệt đối bằng mã nhị phân**

Thay vì dùng ký tự biểu thị quyền, ta có thể thể hiện quyền bằng mã quyền tuyệt đối.

Cách đặt quyền tuyệt đối cho phép thay đổi tất cả các quyền cùng một lúc thay vì phải phân quyền cho từng kiểu người dùng.

Quyền tuyệt đối dùng mã nhị phân để tham chiếu tới các quyền của tất cả người dùng. Do mỗi kiểu người dùng có 3 quyền lần lượt là ***r, w*** và ***x*** nên quyền tuyệt đối của một người dùng gồm 3 bit. Có thể thể hiện giá trị này ở hệ cơ số 8 như:

0 -> 000	1-> 001	2-> 010	3-> 011
4-> 100	5-> 101	6-> 110	7-> 111

Với dãy 3 bit ở trên, nếu tại vị trí có giá trị 0 thì quyền tại đó bị hạn chế, nếu có giá trị 1 thì quyền tại đó là được phép. Dãy liên tiếp gồm 9 bit hay 3 số ở hệ bát phân chính là tập quyền phân cho cả ba kiểu người dùng.

Ví dụ:

$ chmod 544 filename

Lệnh thiết lập quyền trên *filename* như sau:

- Người sở hữu quyền **r**ead và e**x**ec (101 = 5).
- Nhóm người dùng quyền **r**ead (100 = 4).
- Những người khác quyền **r**ead (100 = 4).

Quyền trên thư mục:

Đặt quyền cho thư mục giống như đặt quyền cho file.

- Quyền ***read*** sẽ cho phép hiển thị nội dung thư mục.
- Quyền ***executable*** cho phép di chuyển vào thư mục.
- Quyền ***write*** cho phép tạo hay xoá các file trong thư mục.

Khi bạn tạo một thư mục thì người sở hữu có tất cả các quyền trên thư mục đó.

Thông thường bạn muốn cho những người dùng khác có thể hiển thị và di chuyển vào thư mục của bạn nhưng không được thay đổi nội dung thư mục, khi đó bạn đặt quyền cho những người dùng đó là ***read*** và ***executable***.

Lệnh *ls -l* sẽ hiển thị thông tin về tất cả các file có trong thư mục. Nhưng nếu bạn muốn chỉ hiển thị thông tin về bản thân thư mục thì dùng tùy chọn là *-ld*.

Ví dụ:

$ ls -ld thankyou

drwxr-x---2 nga tinhoc 512 Feb 10 04:30 thankyou

Thay đổi quyền sở hữu file:

Lệnh ***chown*** cho phép chuyển quyền sở hữu một file sang cho người khác.

Ví dụ:

Câu lệnh sau chuyển quyền sở hữu file *mydata* sang cho người dùng *tuan*

$ ls -l

mydata -rw-r--r-- 1 nga tinhoc 207 Feb 15 11:53 mydata

$ chown tuan mydata

$ ls -l

mydata -rw-r--r-- 1 tuan tinhoc 207 Feb 15 11:53 mydata

IV.2.4 Lưu trữ và nén tập tin/ thư mục

Tiện ích *tar* cho phép tạo ra các lưu trữ cho file và thư mục, dùng để tạo các bản backup dữ liệu. Với ***tar***, ta có thể lưu trữ file, cập nhật lưu trữ và thêm vào các file mới. Thậm chí có thể lưu trữ cả một thư mục và tất cả các thư mục con trong nó vào một file lưu trữ, sau đó bạn có thể lấy lại chúng từ file này. Lệnh ***tar*** có nhiều tùy chọn, chẳng hạn ***c** (create), **x** (extract), **u** (update)...*

Cú pháp: **tar *[OPTIONS] [DIRECTORY/FILE]***

OPTIONS:

c: tạo mới một archive.

x: trích file từ một archive.

z: nén/giải nén archive bằng gzip.

j: nén/giải nén archive bằng bzip2.

f: sử dụng archive được chỉ định bởi file.

Ví dụ:

Tạo file lưu trữ tên là *myarch.tar* cho thư mục *mydir*.

$ tar –cf myarch.tar mydir/

Sau lệnh trên, ta sẽ có file *myarch.tar* lưu trữ toàn bộ nội dung của *mydir*.

Ví dụ:

Lấy lại nội dung lưu trữ trong *myarch.tar*.

$ tar -xf myarch.tar

Sau lệnh trên, tất cả nội dung đã lưu trữ trong *myarch.tar* sẽ được lấy ra.

Ví dụ:

Đưa thêm *letters* vào file lưu trữ.

$ tar -rf myarch.tar letters

Ví dụ:

Cập nhật thư mục *mydir* trong *myarch.tar*.

$ tar -uf myarch.tar mydir

Ví dụ:

Liệt kê nội dung chứa trong file lưu trữ *myarch.tar*.

$ tar -tf myarch.tar

Nếu muốn nén file trước khi đưa vào lưu trữ, ta đưa thêm tùy chọn ***z***.

Ví dụ:

$ tar -cfz myarch.tar mydir/

Bạn cũng có thể thêm tùy chọn ***v*** để hiển thị quá trình lấy lại dữ liệu từ file lưu trữ.

Nén file: *gzip*

Để nén file, ta dùng lệnh ***gzip***.

Ví dụ: Nén file *mydata*.

$ gzip mydata

$ ls

mydata.gz

Để giải nén một file nén ta dùng lệnh ***gunzip***.

Ví dụ:

$ gunzip mydata.gz

Để xem nội dung file nén ta dùng lệnh ***zcat***

Ví dụ:

$ zcat mydata.gz

Ta cũng có thể nén một file lưu trữ.

Ví dụ:

$ gzip myarch.tar

$ ls

myarch.tar.gz

Tuy nhiên bạn cần phân biệt việc nén file lưu trữ bằng ***gzip*** và việc nén file trước khi đưa vào lưu trữ dùng tùy chọn ***z*** trong lệnh ***tar***. Hai cách này sẽ đưa lại hiệu quả nén khác nhau.

IV.3 Câu hỏi và bài tập

Câu 1:

- Nêu qui tắc đặt tên partition trong Linux.

- So sánh với qui tắc đặt tên của hệ điều hành Windows.

Câu 2:

Thực hiện thao tác ***mount*** và sử dụng ổ CD, USB trên hệ thống đã cài hệ điều hành Linux.

Câu 3:

Trong giao diện dòng lệnh, hãy gõ lệnh để xem dung lượng đã dùng của các thư mục */home, /dev, /lib, /home/tttccn* (hoặc thư mục người dùng bạn đang sử dụng), */root*, ...

(Sử dụng thêm các tùy chọn trong câu lệnh để có kết quả hiển thị những thông tin mong muốn.)

Câu 4:

- ***inode*** được dùng để làm gì?

- Trong cấu trúc của *inode* có trường nào chứa tên tập tin/thư mục không?

- Hãy kể tên các trường trong cấu trúc của 1 *inode*.

Câu 5:

Hãy lấy 5 ví dụ về:

- Đường dẫn tuyệt đối.

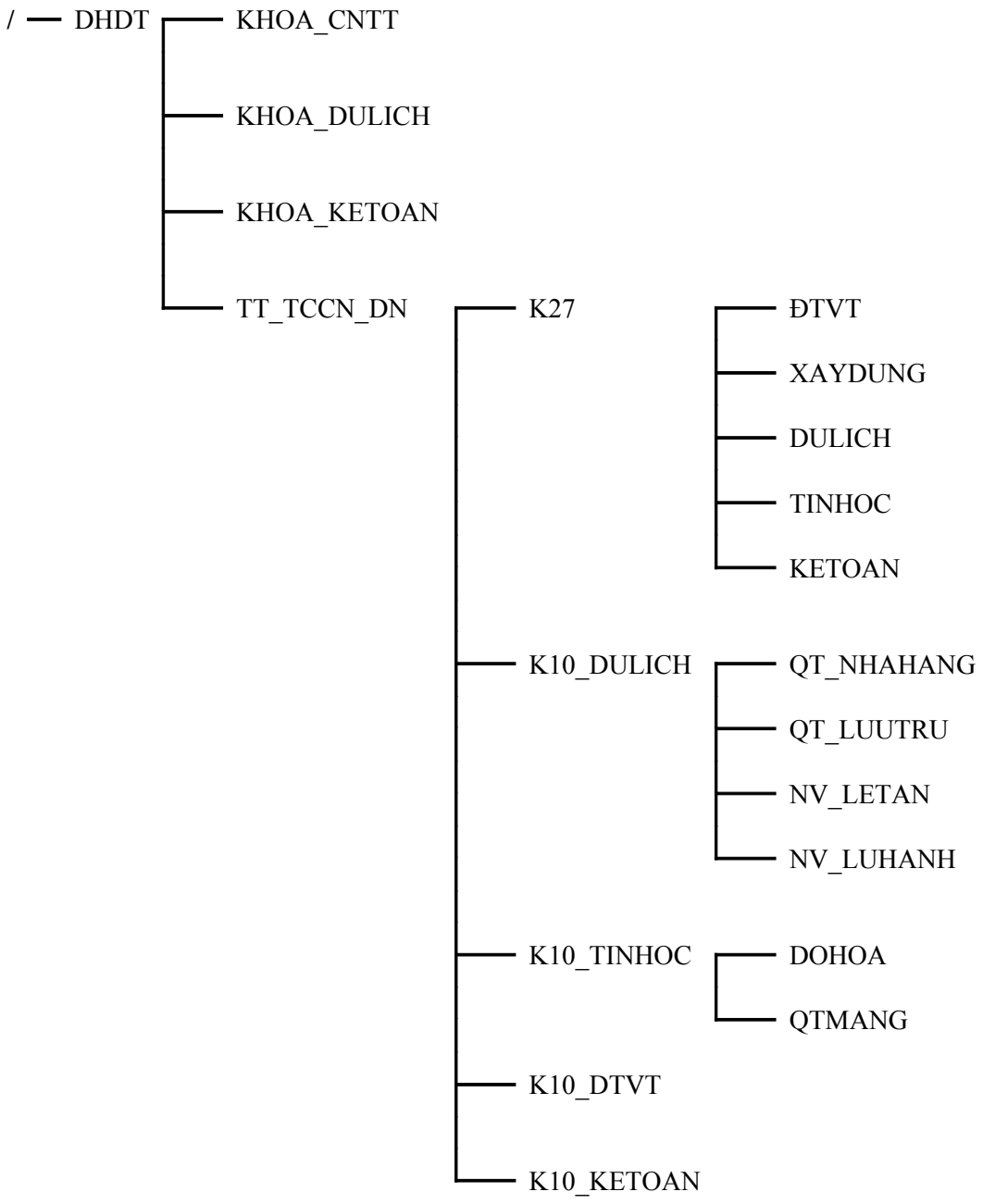
- Đường dẫn tương đối.

- Thư mục hệ thống.

- Tập tin/thư mục ẩn.

Câu 6:

Đang ở thư mục *root*, thực hiện tạo các câu lệnh để tạo cây thư mục sau:



6.1 *Chuyển thư mục làm việc tới /*DHDT. Tạo tập tin */DHDT/kq.txt* có nội dung là kết quả điều hướng của lệnh ***ls –l/DHDT.***

6.2 Tạo thư mục *HOSO* trong thư mục *TT_TCCN_DN.*

6.3 Dùng lệnh ***cat*** hoặc ***vi*** tạo tập tin *ds_lop.txt* trong thư mục *HOSO* ở trên, có nội dung là danh sách lớp (nhập khoảng 10 tên sinh viên).

6.4 Đổi tên tập tin vừa tạo thành *dslopk10.txt*.

6.5 Chuyển thư mục làm việc đến thư mục *HOSO*. Thêm nội dung của lệnh ***date*** vào cuối tập tin *dslopk10.txt*.

6.6 Di chuyển tập tin *kq.txt* vào thư mục *K27* và đổi tên thành *ds.list*.

6.7 Thực hiện xóa 2 thư mục *KHOA_CNTT* và *KHOA_KETOAN*.

6.8 Thực hiện di chuyển thư mục *DULICH* vào thư mục *KHOA_DULICH*.

6.9 Cho biết số dòng, số từ, số ký tự của file *dslopk10.txt*.

6.10 Tạo thư mục *LUUTRU* trong thư mục *TT_TCCN_DN*.

6.11 Copy 2 file *dslopk10.txt* và *ds.list* vào thư mục *LUUTRU* bằng một lệnh.

6.12 Nối lịch của tháng hiện tại vào tập tin *ds.list* ở trên.

6.13 Thay đổi quyền cho tập tin *dslopk10.txt* có quyền như sau ***rwer--r0--***.

6.14 Thay đổi quyền sở hữu tập tin *dslopk10.txt* cho người dùng ***tttccn***.

6.15 Thực hiện nén file *dslopk10.txt*, nén thư mục *DHDT*.

Câu 7.

Đăng nhập bằng người dùng root và thực hiện các công việc sau bằng câu lệnh:

7.1 Tạo thư mục *Baitap* và chuyển thư mục làm việc tới thư mục này.

7.2 Tạo các tập tin sau trong thư mục mới tạo trên: *taptin1.txt* là kết quả của lệnh ***ls –l /dev***, *taptin2.txt* là kết quả của lệnh ***ls –l /etc***, *taptin3.txt* là kết quả của lệnh ***cat /etc/passd***.

7.3 Nén thư mục *Baitap* thành tập tin *Baitap_bk.tar* lưu vào thư mục chủ của root.

7.4 Xem danh sách các tập tin bên trong *Baitap_bk.tar*.

7.5 Giải nén tập tin này vòa thư mục *NOI_DUNG* tạo trong */home*.

7.6 Copy tập tin *Baitap_bk.tar* vào thư mục */home* và đổi tên thành *lab_bk.tar*.

7.7 Nén tập tin *lab_bk.tar* thành dạng ***.gz***.

7.9 Tạo thư mục **LUU_TRU1** trong thư mục */home*. Copy tập tin *lab_bk.tar.gz* vào đây.

7.10 Giải nén hai tập tin này thành dạng ***.tar***.

7.12 Dùng lệnh ***tar*** để giải nén thành nội dung ban đầu.

CHƯƠNG V. QUẢN TRỊ NGƯỜI DÙNG

V.1 Thông tin của user

Như đã biết, trong hệ điều hành đa người dùng, cần phân biệt những người dùng khác nhau do quyền sở hữu các tài nguyên trong hệ thống, chẳng hạn như, mỗi người dùng có quyền hạn với file, quá trình riêng của họ. Điều này vẫn rất quan trọng thậm chí cả khi máy tính chỉ có một người sử dụng tại một thời điểm. Mọi truy cập hệ thống Linux đều thông qua tài khoản người dùng. Vì thế, mỗi người sử dụng được gắn với tên duy nhất (đã được đăng ký) và tên đó được sử dụng để đăng nhập. Việc đăng nhập sẽ giúp hệ thống biết được bạn là ai và có quyền gì. Tuy nhiên một người dùng thực sự có thể có nhiều tên đăng nhập khác nhau. Tài khoản người dùng có thể hiểu là tất cả các file, các tài nguyên, và các thông tin thuộc về người dùng đó.

Mỗi người sử dụng trên hệ thống được mô tả qua các thông tin sau:

- *username:* tên người sử dụng.
- *password:* mật khẩu (nếu có).
- *uid:* số nhận dạng (user identify number).
- *gid:* số của nhóm (group identify number).
- *comment:* chú thích.
- Thư mục chủ của tài khoản (home directory).
- Shell đăng nhập (chương trình chạy lúc bắt đầu phiên làm việc).
- Các thông tin trên được chứa trong tập tin ***/etc/passwd.***

Username và UserID

Tên người dùng là chuỗi ký tự xác định duy nhất một người dùng, người dùng sử dụng tên này khi đăng nhập cũng như truy xuất tài nguyên, trong Linux tên người dùng có sự phân biệt giữa chữ hoa và thường. Thông thường, tên người dùng thường sử dụng chữ thường. Để dễ dàng trong việc quản lý người dùng, ngoài tên

người dùng Linux còn sử dụng khái niệm định danh người dùng (user_ID). Mỗi người dùng có một con số định danh riêng.

Linux sử dụng số định danh để kiểm soát hoạt động của người dùng. Theo qui định chung, những người dùng có định danh là 0 là người dùng quản trị (root). Các số định danh từ 1- 99 sử dụng cho các tài khoản hệ thống, định danh của người dùng bình thường sử dụng giá trị bắt đầu từ 100.

Mật khẩu người dùng

Mỗi người dùng có một mật khẩu riêng để sử dụng tài khoản của mình. Mọi người đều có quyền đổi mật khẩu của chính mình. Người quản trị thì có thể đổi mật khẩu của những người khác.

Unix truyền thống lưu các thông tin liên quan tới mật khẩu người dùng trong tập tin */etc/passwd*. Tuy nhiên, mọi người dùng đều đọc được tập tin này do một số yêu cầu cho hoạt động bình thường của hệ thống (như chuyển User ID thành tên khi hiển thị trong lệnh *ls* chẳng hạn) và nhìn chung các người dùng đặt mật khẩu "yếu" do đó hầu hết các phiên bản Unix mới đều lưu mật khẩu (được mã hóa) thực sự trong một tập tin khác */etc/shadow* và chỉ có root được quyền đọc tập tin này.

Tài khoản root

Trong quá trình cài đặt Linux, trình cài đặt sẽ tạo ra một tài khoản đặc biệt với tên là root cho hệ thống. Tài khoản root còn được gọi là tài khoản quản trị hay superuser có quyền không giới hạn.

Sử dụng quyền root chúng ta thấy rất thoải mái vì chúng ta có thể làm được các thao tác mà không phải lo lắng gì đết xét quyền truy cập này hay khác.

Tuy nhiên, khi hệ thống bị sự cố do một lỗi lầm nào đó, chúng ta mới thấy sự nguy hiểm khi làm việc như root. Lời khuyên là không nên sử dụng tài khoản root để đăng nhập và làm việc với hệ thống và chỉ nên dùng trong những trường hợp thật cần thiết.

Không phải tài khoản superuser nào cũng gọi là root, mặc dù nó được tạo mặc định là root khi cài đặt Linux. Superuser có thể có tên bất kỳ nhưng thường được dùng nhất dưới tên root. Tài khoản này được định nghĩa là tài khoản có UserID là 0, các userID được định nghĩa trong file */etc/passwd.*

Chú ý:

- Nếu bạn đang ở User thường thì dấu nhắc tại Shell là $.

- Nếu bạn đang ở Super User (root) thì dấu nhắc tại Shell là #.

Các tập tin liên quan:

/etc/passwd: lưu trữ thông tin của tất cả các user.

/etc/shadow: lưu trữ tham số điều khiển truy xuất người dùng, mật khẩu và thông tin thời hạn của mật khẩu.

/etc/group: thông tin về nhóm của người dùng.

/etc/gshadow: lưu trữ password được mã hóa của nhóm.

Tập tin etc/passwd:

Tập tin */etc/passwd* đóng một vai trò quan trọng với hệ thống Unix/Linux. Mọi người đều có thể đọc được tập tin này nhưng chỉ có root mới có quyền thay đổi nó.

Tập tin */etc/passwd* được lưu dưới dạng text như đại đa số các tập tin cấu hình khác của Linux. Mỗi dòng trong file này tương ứng với một user. Dòng đầu tiên của tập tin /etc/passwd mô tả thông tin cho user root (chú ý là tất cả những tài khoản có user_ID = 0 đều là root), tiếp theo là các tài khoản khác của hệ thống đây là các tài khoản không có thật và không thể login vào hệ thống), cuối cùng là các tài khoản người dùng thường.

Mỗi dòng trong file tương ứng với bảy trường thông tin của một người dùng, và các trường này được ngăn cách nhau bởi dấu '**:**'. Ý nghĩa của các trường thông tin đó lần lượt như sau:

Cột 1: tên người sử dụng dùng đăng nhập (thường trên 8 ký tự).

Cột 2: mã liên quan đến passwd. Linux lưu mã này trong tập tin */etc/shadow* chỉ có root mới có quyền đọc.

Cột 3:4: user ID:group ID

Cột 5: tên đầy đủ của người sử dụng.

Cột 6: thư mục cá nhân, thường là */home/username* (ví dụ: */home/smith)*. Tất cả những file cá nhân, web pages,…. sẽ được lưu trữ ở đây.

Cột 7: chương trình sẽ chạy đầu tiên sau khi user login (thường là shell, thường được thiết lập "*/bin/bash*").

Ví dụ:

[srv@cap home]$ cat /etc/passwd

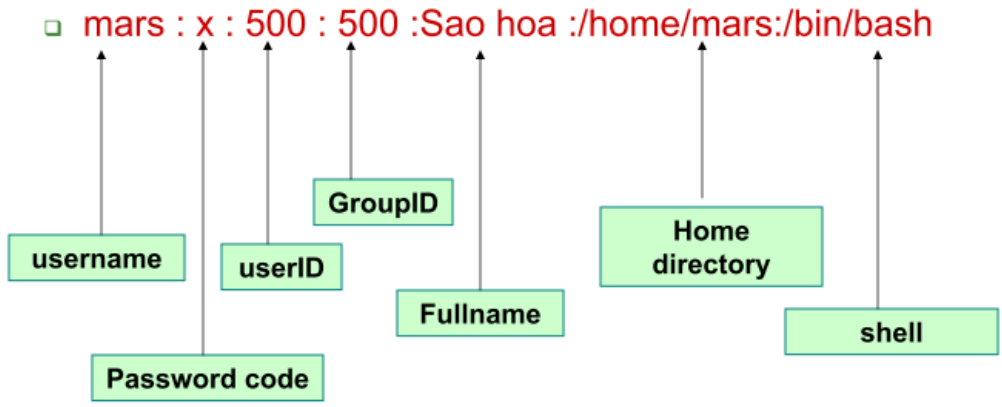
root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:

...

mars:x:500:500:Sao hoa :/home/mars:/bin/bash

Mỗi user được lưu trong một dòng gồm 7 cột



V.2 Các thao tác trên user

➢ **Tạo một user mới**

[redacted]

User sau khi tạo sẽ được lưu trữ thông tin vào file */etc/passwd* và file */etc/shadow*.

Tùy chọn:

***-u* UID:** một ID riêng của user mới (mặc định là ID tiếp theo cho user).

***-g* GID:** nhóm chính của User (mặc định là ID của nhóm other).

***-G* GID:** một số nhóm khác (danh sách của nhóm mà user là thành viên).

***-c* comment:** mô tả cho user (mặc định là trắng).

***-d* directory:** thư mục home (mặc định */home/username*).

***-m*:** tạo ra một thư mục home.

***-s* shell:** shell login (mặc định là */bin/bash*).

Ví dụ:

- **Lệnh thiết lập/thay đổi password:**

Ví dụ:

Thay đổi password cho người dùng *blobby*.

- **Lệnh thay đổi một số thuộc tính của user**

Tùy chọn

Ví dụ:

➢ **Lệnh xóa user đã tạo:**

Tùy chọn: *-r* sẽ xóa tất cả các thư mục home của username.

➢ **Lệnh su, who**

Cú pháp lệnh su: ***su username***

- Cho phép đăng nhập với tư cách là người dùng khác.

Cú pháp lệnh who: ***who [option]***

- Cho biết user đang sử dụng.

Tùy chọn:

-H, --heading: hiển thị tiêu đề của các cột trong nội dung lệnh.

-m: hiển thị tên máy và tên người dùng với thiết bị vào chuẩn.

-q, --count: hiển thị tên các người dùng đăng nhập và số người dùng đăng nhập.

➢ **Lệnh Chage**

Cú pháp:

Thay đổi số ngày thay đổi password và ngày cuối cùng phải thay đổi password.

V.3 Các thao tác trên nhóm

Mỗi người dùng trong hệ thống Linux đều thuộc vào một nhóm người dùng cụ thể. Tất cả những người dùng trong cùng một nhóm có thể cùng truy nhập một trình tiện ích, hoặc đều cần truy cập một thiết bị nào đó như máy in chẳng hạn.

Một người dùng cùng lúc có thể là thành viên của nhiều nhóm khác nhau, tuy nhiên tại một thời điểm, người dùng chỉ thuộc vào một nhóm cụ thể.

Nhóm có thể thiết lập các quyền truy nhập để các thành viên của nhóm đó có thể truy cập thiết bị, file, hệ thống file hoặc toàn bộ máy tính mà những người dùng khác không thuộc nhóm đó không thể truy cập được.

Thông tin về nhóm người dùng được lưu trong file */etc/group*, file này có cách bố trí tương tự như file */etc/passwd*.

Mỗi dòng trong file có bốn trường được phân cách bởi dấu ':'. Ý nghĩa của các trường theo thứ tự xuất hiện như sau:

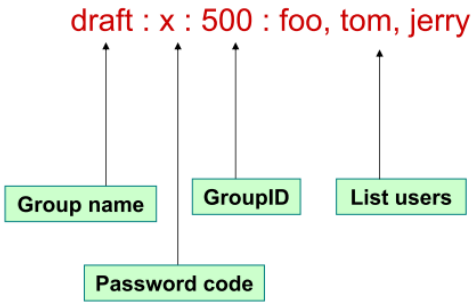
- Tên nhóm người dùng (*groupname*)

- Mật khẩu nhóm người dùng (*passwd* – được mã hóa), nếu trường này rỗng, tức là nhóm không yêu cầu mật khẩu.

- Chỉ số nhóm người dùng (*group id*).

- Danh sách các người dùng thuộc nhóm đó (*users*).

Ví dụ:



Các nhóm mặc định của hệ thống

- Mọi hệ Linux đều có một số các nhóm mặc định thuộc hệ điều hành. Các nhóm này thường là *bin, mail, root, uucp, sys, …*

- Các nhóm mặc định như: ***root, wheel, system:*** thường dùng để cho phép người dùng sử dụng lệnh su để chuyển lên quyền root.

- ***deamon:*** dùng để chỉ những người làm chủ thư mục *spool* (mail, squid, lpd, …).

- ***kmem:*** dùng cho các chương trình truy cập đến *kernel*, bộ nhớ trực tiếp.

- ***tty:*** làm chủ tất cả các file đặc biệt dùng làm việc với terminal.

➢ **Lệnh tạo nhóm mới**

Cú pháp lệnh:

[redacted]

Tùy chọn: *-g gid*: số ID của nhóm, có giá trị là một số nguyên dương >500, lớn hơn mọi ID của nhóm khác có trong hệ thống. 0<=ID<500 là số ID được dành cho các nhóm hệ thống.

Ví dụ:

➢ **Lệnh thay đổi nhóm**

Cú pháp:

Tùy chọn:

– ***n*** *NewName:* thay đổi tên nhóm thành một tên mới.

– *g* *gid*: thay đổi số ID của nhóm.

Ví dụ:

➢ **Lệnh xóa nhóm**

Cú pháp lệnh:

Xóa nhóm người dùng ra khỏi hệ thống.

Ví dụ:

➢ **Thay đổi Password của nhóm**

Cú pháp lệnh:

Ví dụ:

➢ **Lệnh tạo và xóa file */etc/gshadow***

Lệnh tạo file */etc/gshadow*

Lệnh này sẽ xóa tất cả Password trong file */etc/group* và lưu trữ vào file */etc/gshadow*.

Trường Password trong */etc/group* được thay thế bằng '**x**'.

Lệnh xóa file */etc/gshadow*

➢ **Lệnh id và groups**

Cú pháp của lệnh id:

- Liệt kê danh sách *id* của nhóm.

Cú pháp của lệnh groups:

- Liệt kê danh sách tên nhóm.

Ví dụ:

V.4 Câu hỏi và bài tập

Câu 1:

- Hãy kiểm tra xem hệ thống của bạn có bao nhiêu user, bao nhiêu group?

- Những user nào thuộc group nào?

- Kể tên các user và group của hệ thống, user và group được của người dùng bình thường?

Câu 2:

Đăng nhập vào người dùng ***root*** và thực hiện các công việc sau:

2.1 Xem nội dung của file */etc/passwd*. Nêu ý nghĩa của thông tin được hiển thị?

2.2 Tạo 2 user mới là ***user1***, ***user2***.

2.3 Thiết lập hoặc thay đổi password của *user1* là *"tttccn"*, *user2* là *"dhdt"*.

2.4 Đổi tên đăng nhập của *user1* là *usermoi*.

2.5 Dùng lệnh ***su*** để chuyển đổi người dùng sang *user2*. Xem nội dung của file */etc/shadow*.

2.6 Quay về người dùng *root*. Xóa người dùng *user2*.

2.7 Xem *usermoi* thuộc nhóm người dùng nào?

2.8 Xem nội dung của file */etc/group*. Nêu ý nghĩa của các thông tin được hiển thị.

2.9 Tạo thêm 2 nhóm: *group1*, *group2*.

2.10 Thay đổi tên nhóm *group1* thành *groupmoi*.

2.11 Thay đổi *gid* của *group2* bằng một số mới bất kỳ (>500).

2.12 Liệt kê danh sách các tên nhóm.

2.13 Chuyển *usermoi* làm thành viên của nhóm *groupmoi*.

CHƯƠNG VI. CÁC DỊCH VỤ VÀ TIỆN ÍCH TRÊN LINUX

VI.1 Trình soạn thảo văn bản VI

❖ **Giới thiệu trình soạn thảo VI**

- Có nhiều trình soạn thảo văn bản trong Linux: ***vi***, *emacs and xemacs, jed, joe, …*

- Các bản phân phối của Linux và Unix đều có ***vi***.

- VI-phát âm "*vee eye*" là trình soạn thảo văn bản có thể tìm thấy trên hầu hết các phiên bản hệ điều hành Unix.

- VI được phát triển đầu tiên ở đại học California và các phiên bản của nó được kèm theo trong hệ điều hành Unix.

- VI có thể hơi khó làm quen lúc đầu sử dụng nhưng chứa nhiều đặc tính mạnh mẽ.

- Khi soạn thảo với VI dữ liệu được đặt vào buffer và có thể lưu xuống đĩa hoặc bỏ qua.VI dùng rất ít tài nguyên hệ thống.

Cú pháp: ***vi file_name***

❖ **VI có 3 chế độ:**

♦ **Chế độ lệnh-Command mode:**

- Là chế độ mặc định của trình soạn thảo ***vi***.

- Trong chế độ này người dùng có thể thực hiệc các lệnh để:

  - Xóa, sao chép, di chuyển văn bản.
  - Định vị con trỏ, tìm kiếm và thoát khỏi trình soạn thảo này.

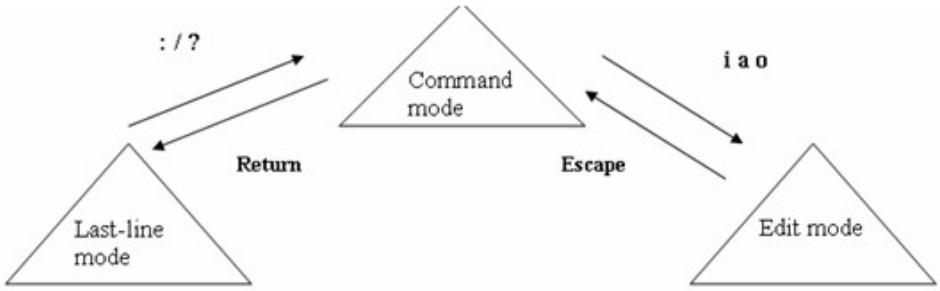
♦ **Chế độ soạn thảo-Edit mode:** cho phép soạn thảo file văn bản

- Để qua chế độ *edit* sử dụng một trong các lệnh sau:

  - ***i:*** chèn văn bản trước con trỏ.
  - ***o:*** mở một dòng mới dưới con trỏ.
  - ***a:*** nối văn bản sau con trỏ.

♦ **Last line mode:** có các lệnh để soạn thảo văn bản nâng cao. Để chuyển qua chế độ này nhấn dấu **":"** ở chế độ command.

Chuyển đổi giữa 3 chế độ:



Hình 6.1

❖ **Chèn và nối văn bản:**

❖ **Di chuyển vị trí con trỏ trong VI:**

h	Qua trái một kí tự
j	Di chuyển xuống một dòng.
k	Di chuyển lên một dòng.
l	Qua phải một kí tự
$	Về cuối dòng
^	Về đầu dòng(không phải khoảng trắng)
G	Về dòng cuối của file văn bản
1G	Về dòng đầu tiên của file văn bản
w	Qua phải một từ
b	Qua trái một từ
:n	Chuyển tới dòng n

❖ **Các lệnh về xóa văn bản:**

R	Ghi đè và thay thế ký tự từ cursor cho đến khi nhấn ESC
C	Ghi đè và thay thế ký tự từ cursor cho đến cuối dòng
X	Xoá ký tự tại cursor
dw	Xoá từ bên phải cursor
dd	Xoá dòng tại cursor
^	Tới đầu dòng
D	Xóa từ cursor đến cuối dòng
:n, n1 d	Xoá từ dòng n đến dòng n1

❖ **Các lệnh cắt dán văn bản**

Yy	Copy một dòng
P	Dán dòng đã copy vào dòng dưới cursor
P	Dán dòng đã copy vào dòng trên cursor
:n co n1	copy từ dòng n tới dòng n và đặt sau dong n1
:n1 m n2	Chuyển từ dòng n1 tới n2

❖ **Các lệnh thay đổi văn bản:**

r	Thay thế kí tự tại vị trí con trỏ bằng một kí tự khác.
~	Chuyển chữ hoa thành chữ thường và ngược lại.
u	Khôi phục lại văn bản trước khi lệnh trước thực hiện.
U	Khôi phục tất cả thay đổi của dòng hiện tại.
.	Lập lại lệnh trước.

❖ **Các lệnh tìm kiếm và thay thế văn bản:**

/string	Tìm chuỗi về phía trước văn bản.
?string	Tìm chuỗi về phía sau văn bản.
n	Tìm chuỗi xuất hiện tiếp theo. Sử dụng sau khi tìm một chuỗi.
N	Tìm chuỗi xuất hiện phía trước.
:%s/old/new/g	Tìm chuỗi old và thay thế bằng chuỗi new cho toàn bộ văn bản.

❖ **Các lệnh thoát và lưu văn bản:**

:w	Lưu sự thay đổi
:w new_file	Lưu trữ tới new_file
:wq[!] hoặc :x[!] hoặc :zz	Lưu và thoát
:q[!]	Thoát mà không lưu

VI.2 Sử dụng e-mail

Thư điện tử hiện nay đang trở thành phương tiện chính để liên lạc trên mạng. Thư điện tử dễ sử dụng, tiện lợi và nhanh chóng. Trong phần này ta sử dụng dịch vụ *sendmail* của hệ thống Linux.

VI.2.1 Gửi thư bằng sendmail

Cú pháp: ***mail <address1> <address2> <address3> ...***

$mail user01 root

- Tiếp theo, trên màn hình xuất hiện.

Subject:

- Bạn gõ vào chủ đề bức thư. Nhấn *Enter*, bắt đầu nhập vào nội dung thư.

- Sau khi nhập vào nội dung thư, nhấn *CTRL-D* để gởi thư đi.

- Trên màn hình xuất hiện:

CC:

- Nhập vào tên những người cùng nhận thư hoặc nhấn *Enter* để bỏ qua.

VI.2.2 Nhận thư

- Khi có thư đến, trên màn hình xuất hiện thông báo:

You have mail

- Để đọc thư, gõ vào lệnh: ***$mail***

- Trên màn hình sẽ liệt kê các bức thư theo thứ tự 1, 2, 3, ... Để đọc nội dung thư nào, gõ vào số thứ tự của bức thư đó.

- Dấu & nhắc rằng bạn đang ở chương trình đọc thư.

- Để xóa thư đang đọc, tại dấu nhắc bạn gõ: ***&d***

- Để thoát chương trình đọc thư, tại dấu nhắc gõ: ***&q***

Ví dụ một phiên gởi mail của *user12*:

[user12@linux user12]$ mail user15 root

Subject: Chao ban

Thuc hanh LINUX

Cc:

[user12@linux user12]$

VI.2.3 Các thao tác hỗ trợ

- Để hủy bỏ thư trước khi gởi, bạn nhấn *CTRL-C* hai lần.

- Đọc nội dung một tập tin trên thư mục hiện hành vào mail: ***-r filename***

- Thay đổi chủ đề của thư: ***~s***

- Xem tất cả các thư lưu trong hộp thư: ***$more mbox***

VI.3 Tiện ích tạo đĩa mềm boot

Ta có thể sử dụng lệnh ***mkbootdisk*** để tạo đĩa mềm khởi động hệ thống. Các bước thực hiện như sau:

- Đăng nhập vào hệ thống bằng user ***root***.

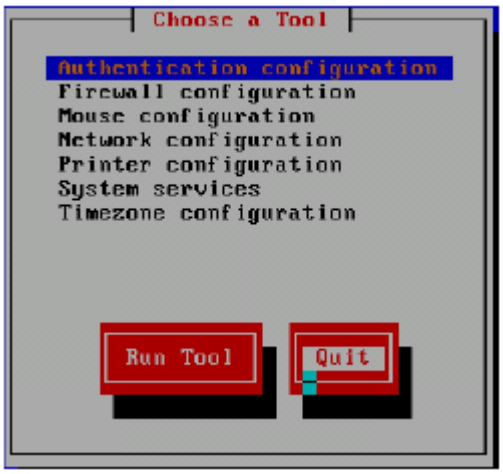
- Xem phiên bản kernel của Linux dùng lệnh ***ls /lib/modules/*** hoặc lệnh ***uname –r*** (trong ví dụ này Linux kernel là *2.2.12-20*).

- Sử dụng lệnh ***/sbin/mkbootdisk 2.2.12-20*** từ dấu nhắc shell.

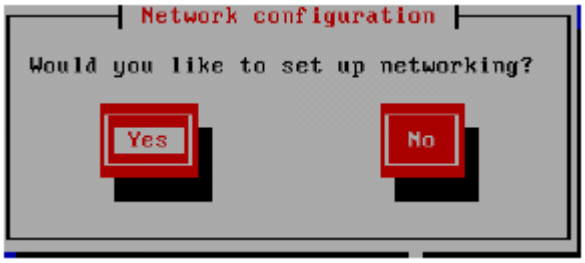
- Đưa đĩa mềm vào ổ đĩa khi được hệ thống yêu cầu (*Insert a disk in /dev/fdo. Any information on the disk will be lost*).

VI.4 Trình tiện ích setup

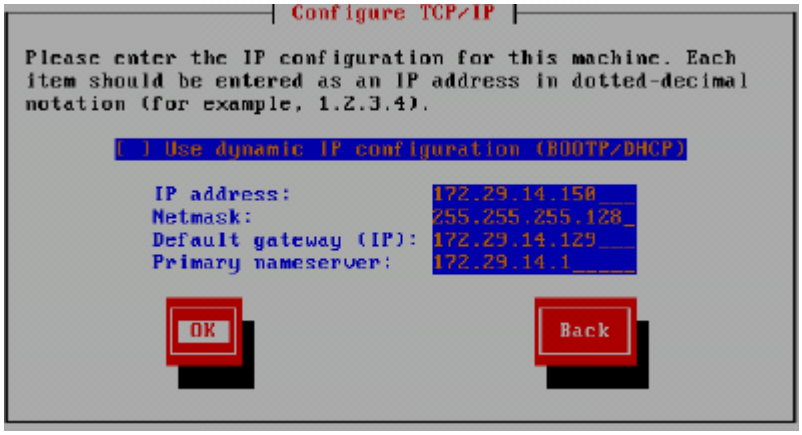
Là trình tiện ích hỗ trợ cài đặt thiết bị, filesystem, thiết lập cấu hình mạng, dịch vụ hệ thống, từ dấu nhắc lệnh ta enter vào lệnh ***setup***, dialog chọn công cụ sẽ được hiển thị.



Ta có thể dùng chương trình này để cài đặt cấu hình TCP/IP cho hệ thống, từ giao diện trên ta chọn item ***Network Configuration -> Run Tool.***



Sau khi ta chọn ***Yes*** để thực hiện quá trình cấu hình thích hợp.



Sau đó ta chọn ***Ok -> Exit***. Có thể dùng lệnh ***/etc/init.d/network restart*** để cập nhật lại các thông số mạng.

VI.5 Trình tiện ích fdisk

Là tiện ích cho phép quản lý ổ đĩa cứng như: tạo mới, xem thông tin và xóa các parttion trong hệ thống. Cú pháp lệnh:

#fdisk <device_name>

Trong đó *<device_name>* có thể là */dev/hda* hoặc */dev/sha*. Sau đây là một số lệnh **fdisk** cơ bản:

Lệnh	Giải thích
P	Liệt kê danh sách các parttition table
N	Tạo mới 1 parttition
D	Xóa parttition
Q	Thoát khỏi trình tiện ích
W	Tạo mới parttition
A	Thiết lập boot parttition
T	Thay đổi system parttition ID
L	Liệt kê loại partition (bao gồm ID)

Sau đây là một số bước để tạo mới một parttition với dung lượng 384M

Bước thực hiện	Giải thích
#fdisk /dev/hdb	Khởi tạo tiện ích fdisk để thao tác lên Parttition */dev/hdb*.
Command (m for help): p Disk /dev/hdb: 64heads, 63sectors, 621 cylinders Units = cylinders ò 4032 * 512 bytes	Liệt kê danh sách các partition trong hệ thống.
Command (m for help):n	Tạo mới một primary

Command action e extended p primary partition (1-4) p Partition number (1-4):1 First cylinder (1-621, default 1): <RETURN> Using default value 1 Last cylinder or +size or + sizeM or +sizeK (1-621, default 621): +384M	partition với kích thước 384MB.
Command (m for help): p Device Boot Start End Blocks Id System /dev/hdb1 1 196 395104 83 Linux	Xem thông tin partition mới vừa tạo.

Lưu ý: Sau khi ta dùng fdisk để tạo một partition mới thì ta phải reboot lại hệ thống và dùng lệnh: ***mkfs –t ext3 <filesystem>*** để định dạng lại partition đó trước khi sử dụng.

VI.6 Câu hỏi và bài tập

Câu 1:

1.1 Dùng lệnh ***vi*** tạo ra tập tin *vanban.txt.*

1.2 Thêm nội dung sau vào văn bản. Dùng các lệnh **a, o, i** để chèn thêm văn bản và lệnh xóa văn bản **X, D,** ... để xóa từ khi đánh sai:

```
Mot so lenh co ban trong trinh soan thao van ban trong VI
- Chen va noi van ban:
        a : chen van ban vao vi tri dau nhac hien thoi.
        A : chen van ban vao cuoi mot dong
        i : chen van ban vao ben trai dau nhac
        o : chen 1 dong trong vao duoi dong hien tai
        O : chen 1 dong trong vao tren dong hien tai
        :r tenfile : chen vao vi tri con tro noi dung cua file
- Cac lenh ve xoa van ban
        R : ghi de va thay the ky tu tu con tro cho den khi nhan nut Esc
        X : Xoa ky tu tai vi tri con tro
        dw : Xoa tu tai con tro.
        dd : Xoa dong tai con tro.
        D : Xoa tu con tro den cuoi dong.
        :n1, n2 d : xoa dong n1 den n2
- Cac lenh cat dan van ban
        yy: copy 1 dong.
        p : dan dong da copy vao dong duoi con tro.
        P : dan dong da copy vao dong tren con tro.
        :n1 co n2- copy dong n1 va dat sau dong n2.
        :n1 m n2 - chuyen dong n1 xuong dat ta dong n2
- Cac lenh thoat va luu van ban
        :w : luu su thay doi file
        :w Ten_file_moi: luu tru toi Ten_file_moi
        :wq[!] hoac :x[!] hoac :zz : luu tru va thoat
        :q[!] : thoat ma khong luu
```

1.3 Dùng lệnh ***:wq*** để lưu và thoát.

1.4 Dùng lệnh ***vi*** để tạo tập tin *noivb.txt* có nội dung sau:

```
Truong Dai hoc Duy Tan
Trung tam THCN&DN
Lop: Tin hoc
```

1.5 Dùng lệnh ***vi*** để mở file *vanban.txt*, sau đó dùng lệnh ***:r noivanban.txt*** để nối nội dung của file *noivb.txt* vào sau nội dung của *vanban.txt*.

1.6 Dùng lệnh *:set nu* để hiển thị thêm số thứ tự ở đầu mỗi dòng.

Câu 2:

Soạn và gửi thư từ người dùng *root* tới một người dùng khác trong hệ thống.

Câu 3:

Nêu mục đích, cách thực hiện các trình tiện tích **tạo đĩa mềm boot**, **tiện ích setup, tiện ích fdisk.**

TÀI LIỆU THAM KHẢO

- ***Tài liệu chính***

1. Tập bài giảng: **Hệ điều hành Linux,** do GVBM soạn.

2. Sách: **Nhập môn Hệ điều hành Linux**, NXB KH&KT.

- ***Tài liệu tham khảo***

Sách:

1. **Red Hat Linux 9.0**, NXB Thống Kê.

2. **Linux toàn tập**, NXB Thống Kê.

3. **100 thủ thuật cao cấp với Linux**, NXB GTVT.

4. **Giáo trình lý thuyết và thực hành Linux**, NXB Thống Kê.

5. Linux Junior Level Administration - LPI 101

6. Linux Junior Level Administration - LPI 102

7. Mastering™ Red Hat® Enterprise Linux 3, Michael Jang.

Địa chỉ trang web:

http://www.redhat.com/

http://www.linuxhomenetworking.com/

GIÁO VIÊN HƯỚNG DẪN **TRỢ GIẢNG**

GIÁM ĐỐC TT TCCN&DN

QUẢN TRỊ HỆ ĐIỀU HÀNH LINUX

MỤC LỤC

Thay đổi UID
Thay đổi nhóm mặc định
Thay đổi thời hạn kết thúc của một tài khoản
Sửa đổi một nhóm đang tồn tại
Xóa hoặc hủy bỏ một tài khoản người sử dụng

Cài đặt máy in

Cấu hình máy in
Cài đặt máy in cục bộ
Cài đặt máy in trên hệ thống Unix ở xa
Cài đặt máy in Samba (SMB)
Chọn trình điều khiển Print Driver và kết thúc
Thay đổi thông số cấu hình các máy in có sẵn
Backup các thông số cấu hình máy in
Quản lý công việc in ấn

5. **Trình diễn thiết lập mạng và cài đặt Diul-up trên Linux**

Thiết lập mạng

HĐH Linux và card mạng
Cấu hình card mạng
Các tiện ích mạng: Telnet và ftp

Cài đặt Diul-up

Cài đặt
Quay số từ xa

6. **Lập trình shell**

Tạo và chạy chương trình shell
Sử dụng các biến

Gán một giá trị cho một biến
Tham số và các biến Shell có sẵn

Sử dụng dấu trích dẫn
Làm việc với câu lệnh test
Sử dụng các câu lệnh rẽ nhánh

Lệnh if
Lệnh case

Sử dụng các câu lệnh vòng lặp

Lệnh for
Lệnh while
Lệnh until
Lệnh shift
Lệnh select
Lệnh repeat

Sử dụng các hàm
Tổng kết

7. **Cài đặt và Quản trị WebServer**

Hướng dẫn cài đặt trên môi trường Linux
Quản trị WebServer

Phần mềm Apache
Biên dịch và cài đặt
Khởi động và tắt WebServer
Cấu hình Apache
Xác thực người dùng

8. **Quản lý tiến trình**

Tiến trình

Tiến trình tiền cảnh

Tiến trình hậu cảnh

Điều khiển và giám sát tiến trình

Sử dụng lệnh ps để lấy thông tin trạng thái của tiến trình

Phát tín hiệu cho một chương trình đang chạy

Giao tiếp giữa các tiến trình

Lập kế hoạch các tiến trình

Sử dụng lệnh at

Sử dụng lệnh crontab

9. **Bảo mật hệ thống**

Những nguy cơ an ninh trên Linux

Xem xét chính sách an ninh của bạn

Tăng cường an ninh cho KERNEL

An toàn các giao dịch trên mạng

Linux firewall

Dùng công cụ dò tìm để khảo sát hệ thống

Phát hiện sự xâm nhập qua mạng

Kiểm tra khả năng bị xâm nhập

Đối phó khi hệ thống bị tấn công

1. Giới thiệu hệ điều hành Linux

1.1. Lịch sử

Linux là hệ điều hành mô phỏng Unix, được xây dựng trên phần nhân (kernel) và các gói phần mềm mã nguồn mở. Linux được công bố dưới bản quyền của GPL (General Public Licence).

Unix ra đời giữa những năm 1960, ban đầu được phát triển bởi AT&T, sau đó được đăng ký thương mại và phát triển theo nhiều dòng dưới các tên khác nhau. Năm 1990 xu hướng phát triển phần mềm mã nguồn mở xuất hiện và được thúc đẩy bởi tổ chức GNU. Một số licence về mã nguồn mở ra đời ví dụ BSD, GPL. Năm 1991, Linus Torvald viêt thêm phiên bản nhân v0.01 (kernel) đầu tiên của Linux đưa lên các BBS, nhóm người dùng để mọi người cùng sử dụng và phát triển. Năm 1996, nhân v1.0 chính thức công bố và ngày càng nhận được sự quan tâm của người dùng. Năm 1999, phiên bản nhân v2.2 mang nhiều đặc tính ưu việt và giúp cho linux bắt đầu trở thành đối thủ cạnh tranh đáng kể của MSwindows trên môi trường server. Năm 2000 phiên bản nhân v2.4 hỗ trợ nhiều thiết bị mới (đa xử lý tới 32 chip, USB, RAM trên 2GB...) bắt đầu đặt chân vào thị trường máy chủ cao cấp. Quá trình phát triển của linux như sau:

- Năm 1991: 100 người dùng.
- Năm 1997: 7.000.000 người dùng.
- Năm 2000: hàng trăm triệu người dùng, hơn 15.000 người tham gia phát triển Linux. Hàng năm thị trường cho Linux tăng trưởng trên 100%.

Các phiên bản Linux là sản phẩm đóng gói Kernel và các gói phần mềm miễn phí khác. Các phiên bản này được công bố dưới licence GPL. Một số phiên bản nổi bật là: Redhat, Caldera, Suse, Debian, TurboLinux, Mandrake.

Giống như Unix, Linux gồm 3 thành phần chính: kernel, shell và cấu trúc file.

Kernel là chương trình nhân, chạy các chương trình và quản lý các thiết bị phần cứng như đĩa và máy in.

Shell (môi trường) cung cấp giao diện cho người sử dụng, còn được mô tả như một bộ biên dịch. Shell nhận các câu lệnh từ người sử dụng và gửi các câu lệnh đó cho nhân thực hiện. Nhiều shell được phát triển. Linux cung cấp một số shell như: desktops, windows manager, và môi trường dòng lệnh. Hiện nay chủ yếu tồn tại 3 shell: Bourne, Korn và C shell. Bourne được phát triển tại phòng thí nghiệm Bell, C shell được phát triển cho phiên bản BSD của UNIX, Korn shell là phiên bản cải tiến của Bourne shell. Những phiên bản hiện nay của Unix, bao gồm cả Linux, tích hợp cả 3 shell trên.

Cấu trúc file quy định cách lưu trữ các file trên đĩa. File được nhóm trong các thư mục. Mỗi thư mục có thể chứa file và các thư mục con khác. Một số thư mục là các thư mục chuẩn do hệ thống sử dụng. Người dùng có thể tạo các file/thư mục của riêng mình cũng như dịch chuyển các file giữa các thư mục đó. Hơn nữa, với Linux người dùng có thể thiết lập quyền truy nhập file/thư mục, cho phép hay hạn chế một người dùng hoặc một nhóm truy nhập file. Các thư mục trong Linux được tổ chức theo cấu trúc cây, bắt đầu bằng một thư mục gốc (root). Các thư mục khác được phân nhánh từ thư mục này.

Kernel, shell và cấu trúc file cấu thành nên cấu trúc hệ điều hành. Với những thành phần trên người dùng có thể chạy chương trình, quản lý file, và tương tác với hệ thống.

1.2. Cài đặt máy chủ Linux

Lưu ý: trước khi cài đặt, cần tìm hiểu các thông tin về phần cứng của hệ thống, bao gồm

- Thông tin về ổ đĩa cứng
- Thông tin về card mạng
- Thông tin về card đồ hoạ
- Thông tin về màn hình
- Thông tin về giao thức và cấu hình mạng nếu kết nối mạng
- Thông tin về các thiết bị ngoài.

Có thể chọn nhiều phương án cài đặt như cài đặt từ đĩa mềm, từ đĩa cứng, từ đĩa CD Rom hoặc qua mạng. Tài liệu này chọn hướng dẫn quá trình cài đặt phiên bản 7.0 từ đĩa CDRom. Yêu cầu máy cài đặt có khả năng khởi động (boot) từ ổ đĩa CD-Rom (được hỗ trợ hầu hết trong các máy tính hiện nay).

Sau đây là các bước cài đặt cụ thể. Khi kết thúc bước trước chương trình cài đặt tự động chuyển sang bước sau. Một số bước cài đặt cho phép quay lại bước trước bằng cách chọn Back.

1. Đưa đĩa CD Rom Redhat vào ổ đĩa. Khởi động lại máy (lưu ý phải đảm bảo máy có khả năng khởi động từ đĩa CD-Rom. Chọn chế độ cài ***text***
2. Chọn chế độ cài ***text***

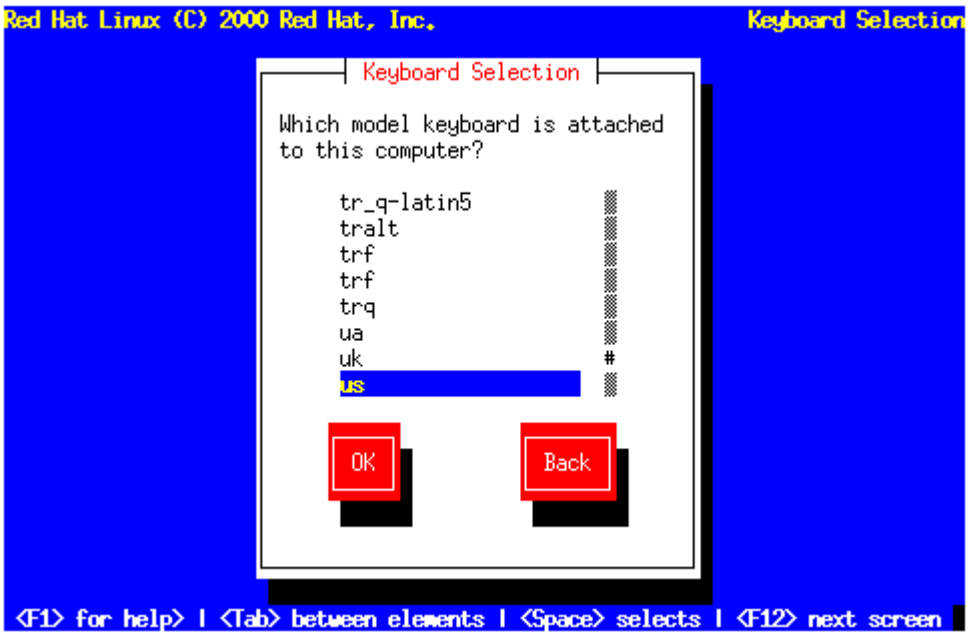
```
boot: text
```

3. Lựa chọn ngôn ngữ

Chọn ngôn ngữ mặc định là English

4. Lựa chọn kiểu bàn phím

Lựa chọn kiểu thể hiện bàn phím là **us**.



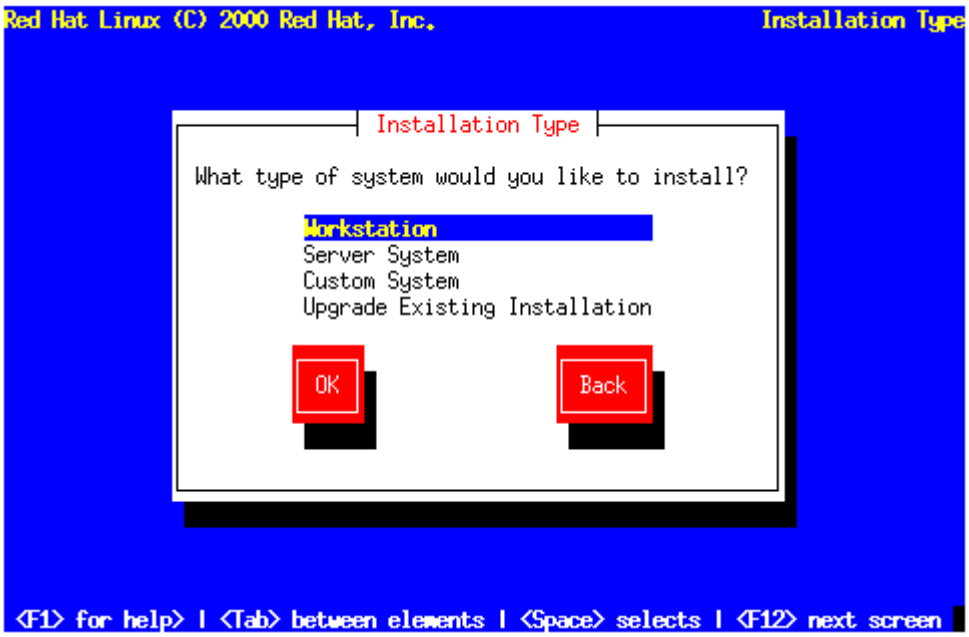
5. Màn hình chào mừng

Sau khi đã lựa chọn xong ngôn ngữ cài đặt, bàn phím và phương pháp cài đặt, màn hình chào mừng xuất hiện. Bấm OK để tiếp tục.

6. Chọn kiểu cài đặt

Hộp hội thoại cho phép bạn chọn lựa kiểu cài đặt hệ điều hành Linux RedHat như một Workstation, Server, Custom hay chỉ là nâng cấp phiên bản đã cài đặt.

Chọn kiểu cài đặt là Custom System. Chọn OK để tiếp tục.

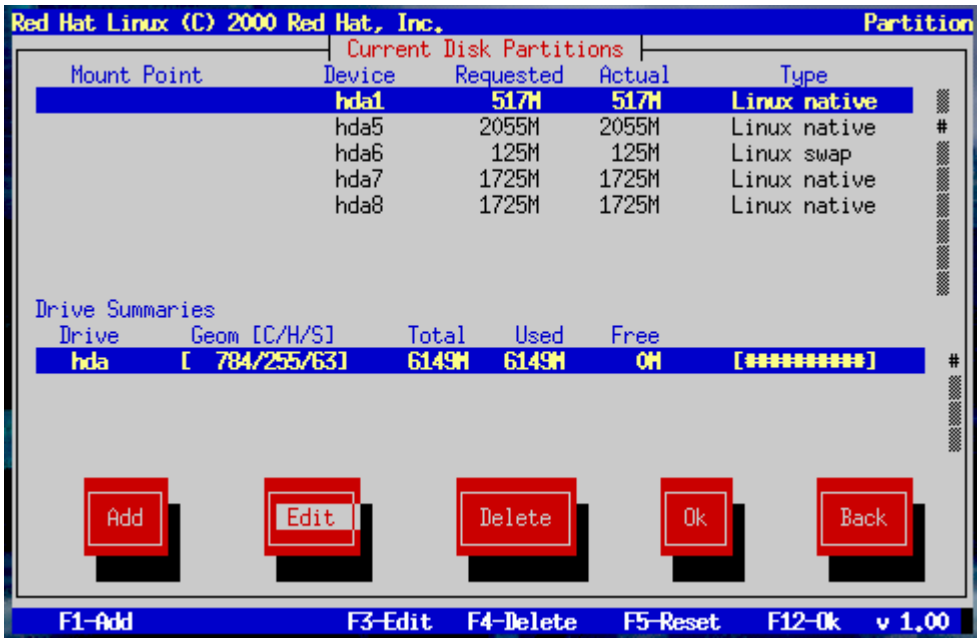


7. Lựa chọn phần mềm phân chia ổ đĩa

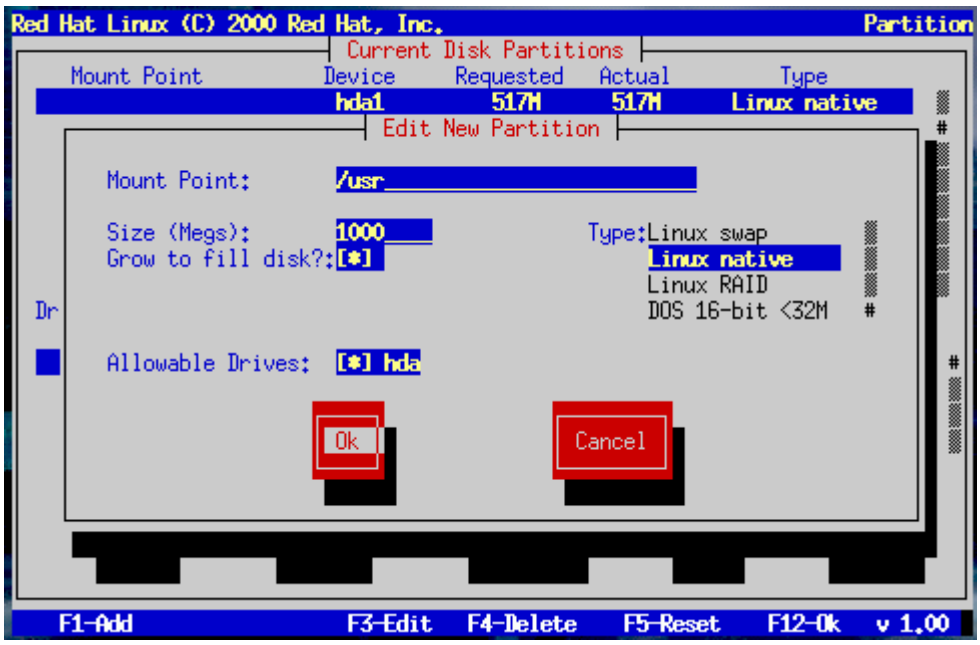
Linux đưa ra cho bạn hai phần mềm để phân chia ổ đĩa dành cho Linux: đó là `Disk Druid và fdisk. Chọn Disk Druid để tiếp tục.`

Bạn cần tạo 2 partition để install RedHat, nhớ đừng delete những partition có sẵn trong máy bạn (nếu không thì dữ liệu có sẵn sẽ mất, tốt nhất là bạn nên sao lưu dữ liệu trước cho bảo đảm!). Dùng các chức năng **add**, **edit**, **delete** tạo 1 partition với type là

Linux swap, dung lượng bằng dung lượng RAM của máy. Tiếp theo tạo một partion tên "/" với loại **Linux native**, dung lượng ít nhất là 500Mb (tuỳ theo dung lượng còn trống của đĩa bạn, nếu bạn muốn install trọn gói RedHat thì cần đến khoảng 2288MB). Hãy yên chí là nếu bạn tạo sai (partition kích thước quá lớn, lớn hơn dung lượng còn trống của đĩa) thì RedHat sẽ không cho bạn đi tiếp. Chỉ cần tạo 2 partition này là đủ rồi. Khi nào bạn click được Next thì *coi như* là thành công!



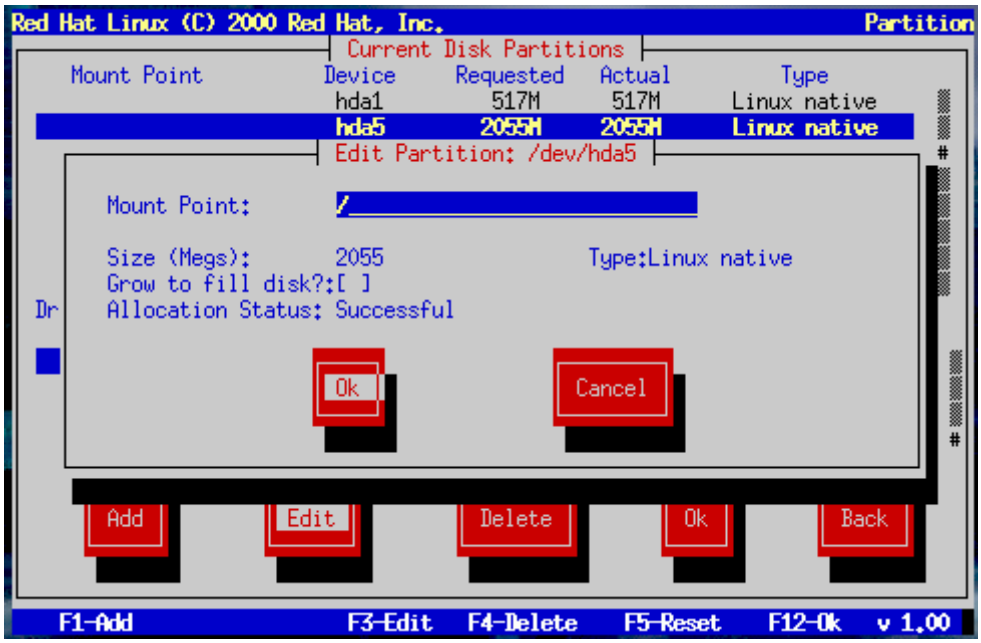
Để tạo một partition mới, chọn Add. Màn hình **`Edit New Partition`** xuất hiện



Một số vấn đề có thể xảy ra khi thêm một partition

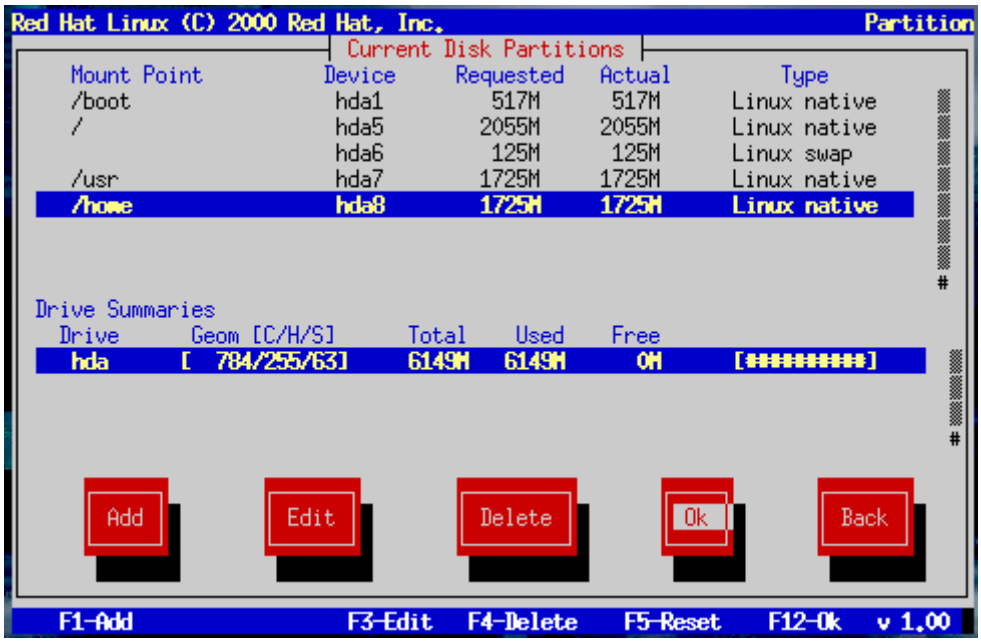
8. Hiệu chỉnh một partition

Chọn một partition cần hiệu chỉnh, nhấn Edit, màn hình mới sẽ cho phép bạn thay đổi các thông số của partition đã chọn như kích thước, kiểu, ...



9. Hoàn thành việc phân chia đĩa

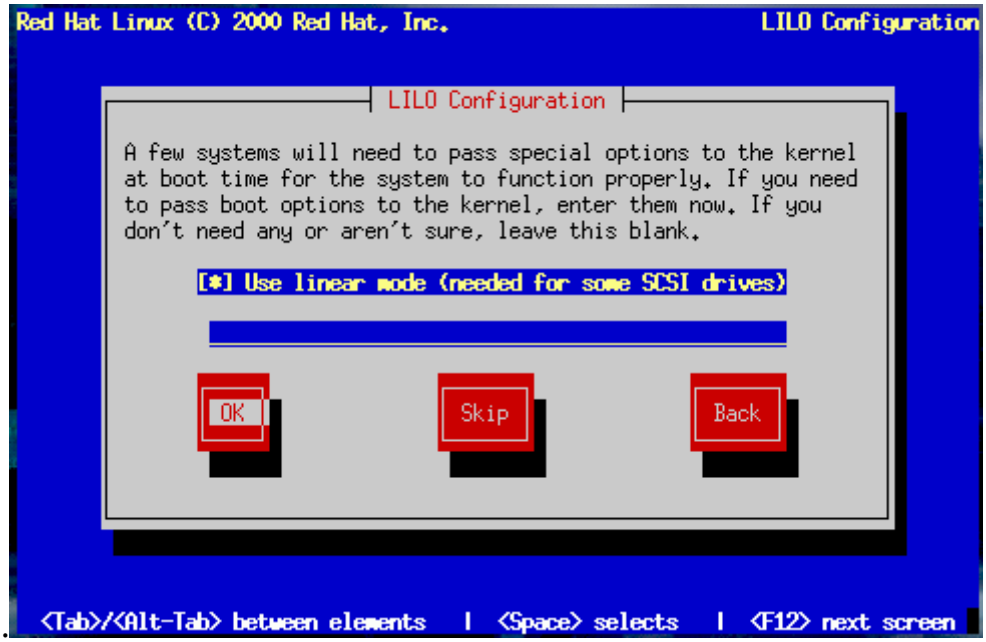
Chương trình cài đặt sẽ yêu cầu bạn format lại phân vùng vừa tạo, chú ý không chọn những phân vùng dữ liệu quan trọng đối với bạn.



10. Khởi tạo LILO

LInux **LO**ader (LILO) cho phép bạn xác định thời gian để khởi tạo Linux hay một hệ điều hành nào khác. Khi khởi tạo cho server, LILO được cấu hình tự động trên Master Boot Record [MBR]. If you are performing a custom-class installation, the **LILO Installation** dialogs let you indicate how or whether to install LILO.

Việc chọn LILO trong cửa sổ **LILO Configuration** cho phép bạn thêm các tuỳ chọn mặc định vào lệnh boot LILO và các tùy chọn này được chuyển cho Linux kernel tại thời điểm boot.



Chú ý rằng nếu bạn chọn **Skip**, bạn sẽ không thể boot hệ thống Red Hat Linux một các trực tiếp mà sẽ phải sử dụng phương pháp boot khác (boot disk chẳng hạn) Bạn chỉ nên lựa chọn cách này khi bạn chắc chắn đã có cách khác để boot hệ thống Red Hat Linux của bạn.

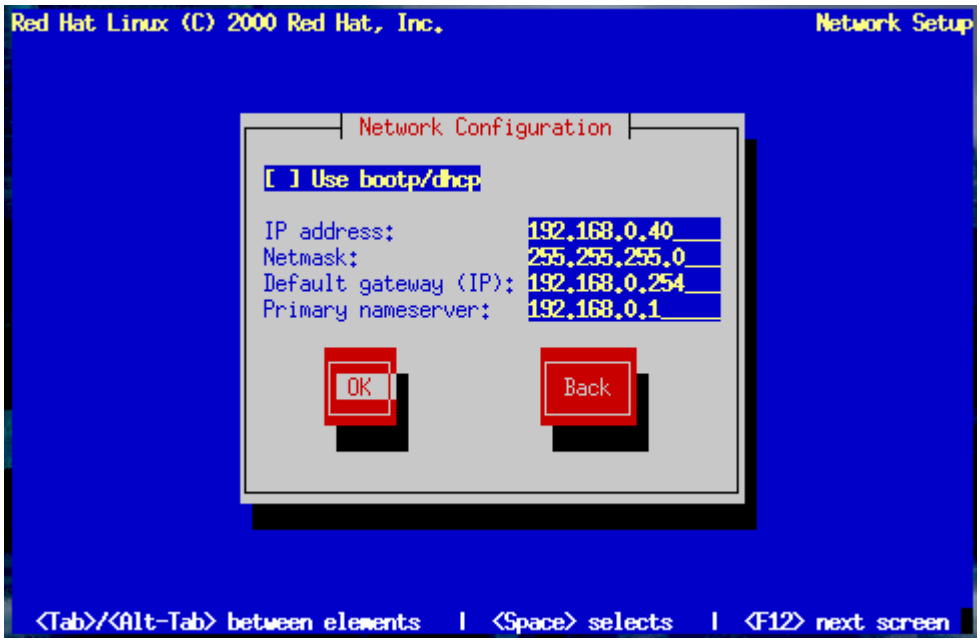
Dùng lựa chọn đặt boot loader tại Master Boot Record để khởi tạo ngay hệ điều hành Linux khi bật máy.

Màn hình này cho phép bạn đặt tên cho máy tính của mình. Bạn có thể thay đổi hostname sau khi đã cài đặt xong bằng lệnh **hostname *newname*,** trong đó newname là tên mà bạn muốn đặt.

11. Cấu hình kết nối mạng

Nếu máy không có card mạng, sẽ không nhận được màn hình này. Thực hiện cấu hình mạng cho máy như sau

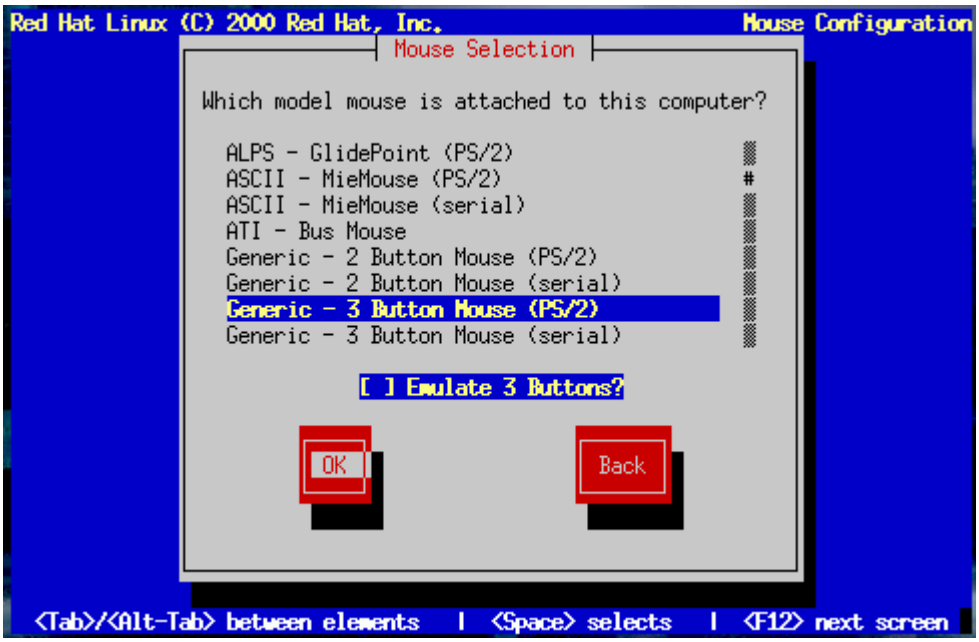
Bỏ lựa chọn ***config using DHCP*** *(chế độ cấp phát địa chỉ IP động)*, nhập địa chỉ IP, subnetmask theo hướng dẫn của giáo viên hướng dẫn thực hành.



12. Cấu hình firewall: chọn ***Medium***

13. Cấu hình chuột

Thông thường thì chương trình cài đặt sẽ tự phát hiện loại chuột của máy bạn. Nếu không, bạn hãy chọn loại chuột phù hợp trong danh sách, và nếu bạn không biết chuột của mình loại gì thì cứ để yên, click **Next** để tiếp tục.



Lựa chọn **Emulate 3 Buttons** cho phép bạn sử dụng chuột của bạn như chuột có 2 nút trong đó dùng nút giữa bằng cách bấm hai nút cùng một lúc.Nếu bạn có chuột hai nút, bạn hãy sử dụng chức năng này vì XWindow trở nên dễ dùng nhất với khi chuột có ba nút.

14. Cấu hình Time Zone

Nếu bạn muốn thiết lập đồng hồ cho CMOS theo giờ GMT (Greenwich Mean Time), chọn **Hardware clock set to GMT**. Tuy nhiên, nếu máy tính của bạn sử dụng một hệ

điều hành khác thì việc thiết đặt đồng hồ theo giờ GMT sẽ khiến cho hệ điều hành khác đó hiển thị sai thời gian.

Để đặt giờ VN, chọn Asia/Saigon

Để thay đổi cấu hình về thời gian sau khi bạn đã cài đặt, bạn có thể dùng lệnh **/usr/sbin/timeconfig**

15. Thiết lập mật khẩu root

Hộp thoại **Root Password** buộc bạn phải thiết lập một mật khẩu root cho hệ thống của bạn. Bạn sẽ sử dụng mật khẩu này để log vào hệ thống và thực hiện các chức năng quản trị hệ thống của mình.

16. Tạo user

Bạn có thể tạo tài khoản user cho chính mình để sử dụng hàng ngày. User root (*superuser*) có đủ quyền truy nhập vào hệ thống nhưng rất nguy hiểm, chỉ nên sử dụng để bảo dưỡng hay quản trị hệ thống.

Mật khẩu của user có phân biệt chữ hoa chữ thường và ít nhất là 6 ký tự.

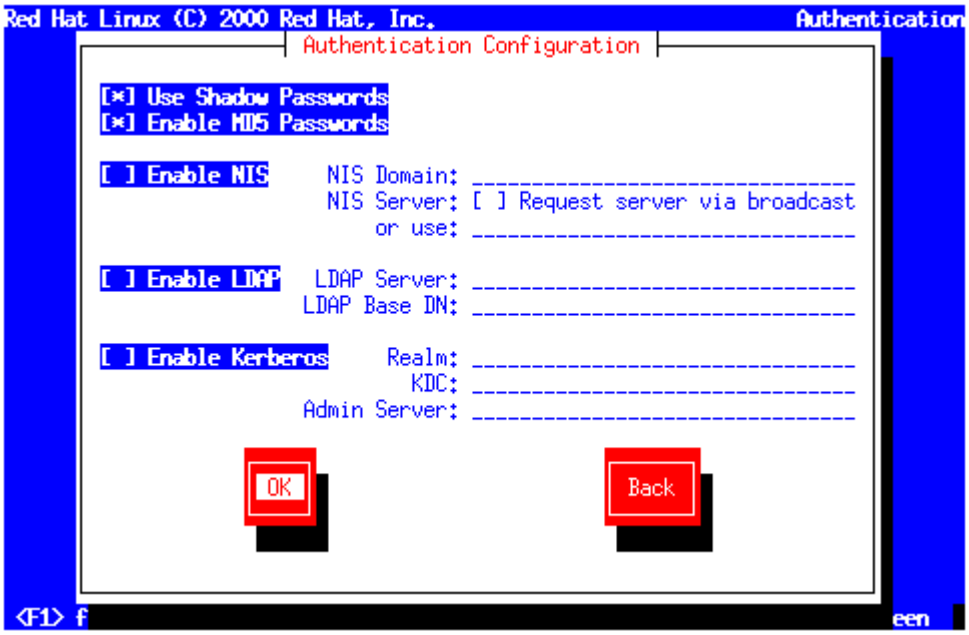
15. Bạn có thể tạo tiếp nhiều user theo cửa sổ sau:

16. Cấu hình xác thực người dùng

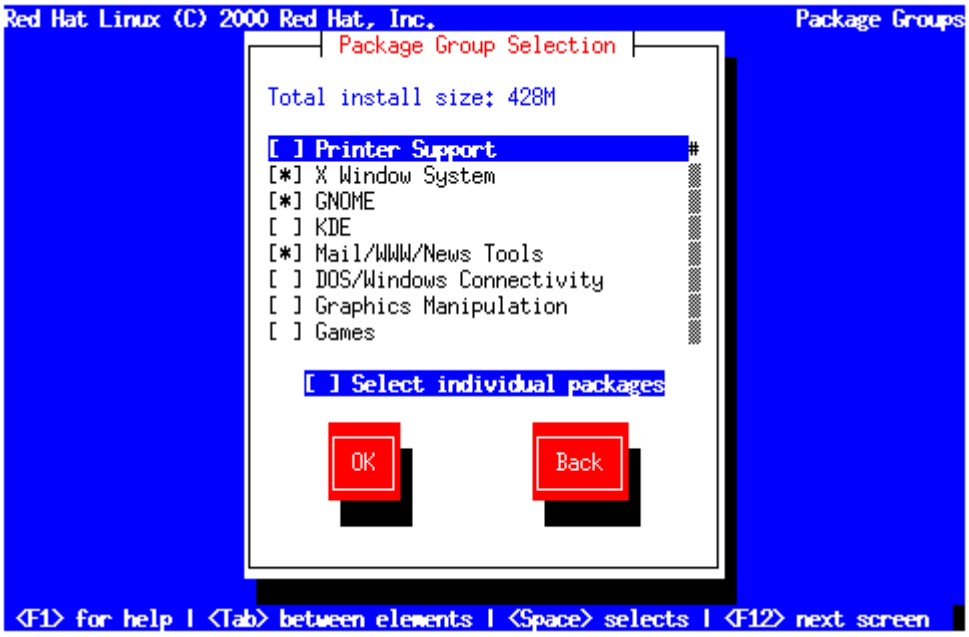
Do bạn khởi tạo theo chế độ custom, bước này cho phép bạn cấu hình cách mà hệ điều hành linux của bạn sử dụng để xác thực mật khẩu.

Lựa chọn **Use Shadow Passwords**: mật khẩu của bạn đáng nhẽ nằm trong tệp /etc/passwd sẽ được thay thế bằng thư mục /etc/shadow và chỉ được truy nhập bởi superuser (root)

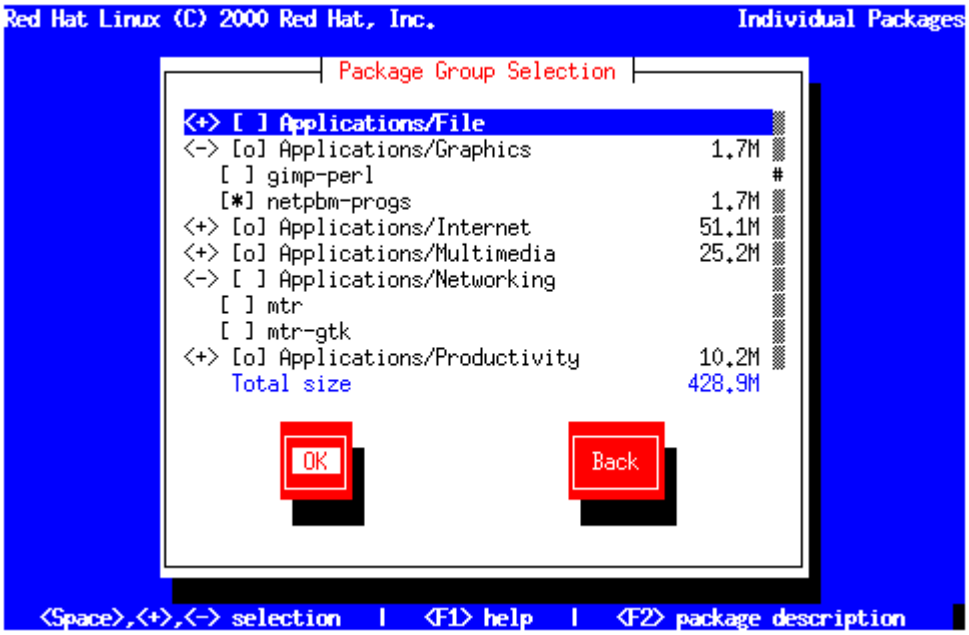
Tuỳ chọn **Enable MD5 Passwords** -- cho phép mã hóa mật khẩu theo chuẩn MD5.



17. Tiếp theo, bạn có thể chọn lựa các gói tin để cài đặt. Bạn nên chọn các phần mềm, dịch vụ hay sử dụng nhất để cài đặt sẵn trên máy khi khởi động. Tuy nhiên, tuy nhiên, bạn cũng có thể cài đặt sau này tuỳ theo nhu cầu sử dụng. Các gói tin này nếu được cài đặt sẽ được ghi lại trong tệp /tmp/install.log sau khi khởi tạo lại hệ thống của bạn.



Có thể cài đặt từng gói tin nhỏ hơn bằng cách chọn **Select individual packages** và nhấn OK.

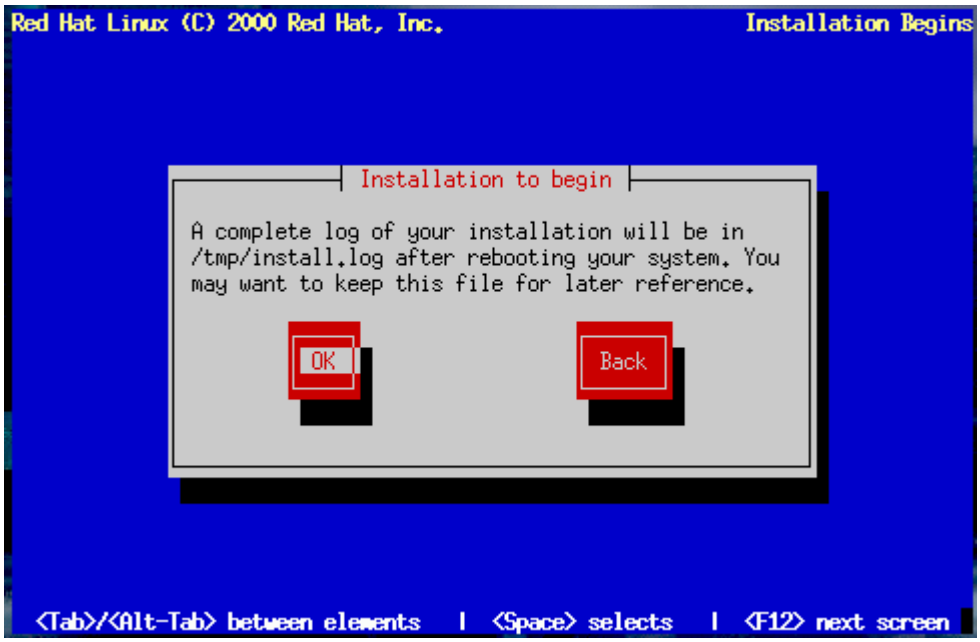


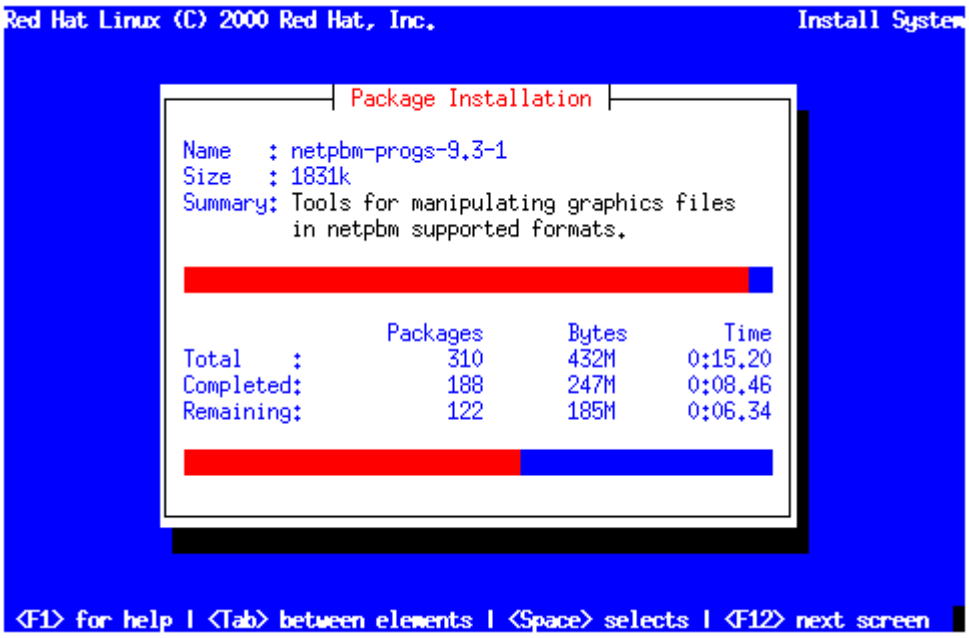
18. Cấu hình Video Adapter

Chương trình cài đặt sẽ tự phát hiện video card khởi tạo. Nhấn OK để tiếp tục.

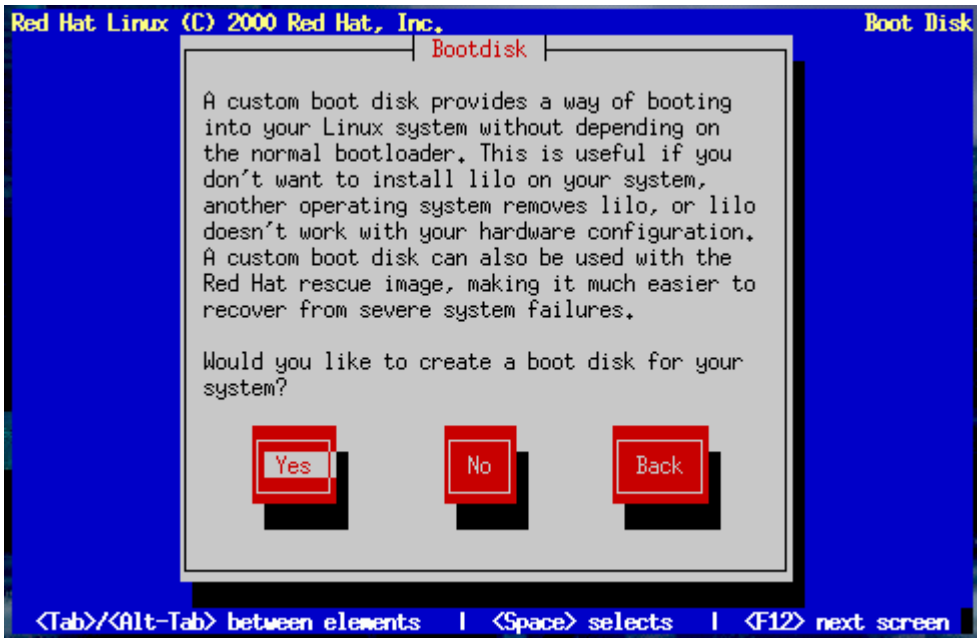
19. Bắt đầu khởi tạo các gói tin:

Quá trình khởi tạo sẽ được ghi vào tệp /tmp/install.log. Nhấn OK để tiếp tục.





20. Tạo đĩa khởi tạo cho hệ thống (boot disk): Chọn No và tiếp tục.



21. Hoàn thành cài đặt

Như vậy là bạn đã hoàn thành xong công việc cài đặt hệ điều hành RedHat 7.0. Bạn hãy rút đĩa ra khỏi ổ CD và nhấn OK để khởi động lại hệ thống.

2. Giao tiếp trên môi trường Linux

2.1. Trình soạn thảo vi

Chương trình vi là một chương trình soạn thảo mạnh mà gần như chắc chắn được tìm thấy trên tất cả các hệ điều hành họ UNIX bởi kích thước và khả năng của nó.vi không đòi hỏi nhiều tài nguyên, thêm vào đó là các chức năng soạn thảo cơ bản. vi có thể tìm kiếm, thay thế, và kết nối các file,và nó có ngôn ngữ macro của chính nó, cúng như một số các đặc điểm bổ sung.Có hai chế độ trong vi:

Chế độ thứ nhất là chế độ input. Trong chế độ này, văn bản được đưa vào trong tài liệu, bạn có thể chèn hoặc bổ sung văn bản.

Chế độ thứ hai là chê độ dòng lệnh. Khi ở chế độ này, bạn có thể dịch chuyển trên tài liệu, trộn các dòng, tìm kiếm, …Bạn có thể thực hiện tất cả các chức năng của vi từ chế độ dòng lệnh ngoại trừ việc nhập vào văn bản. Văn bản chỉ có thể được vào trong chế độ input.

Khi vi khởi động, nó ở chế độ dòng lệnh. bạn có thể chuyển đổi từ chế độ dòng lệnh sang chế độ input bằng cách sử dụng một trong các câu lệnh sau: [aAiIoOcCsSR]. Để trở lại chế độ dòng lệnh bạn chọn phím ESC. Hãy xem các câu lệnh và tác dung của các câu lệnh trong chế độ dòng lệnh.

Câu lệnh	Tác dụng
Ctrl + D	Chuyển cửa sổ xuống bằng một nửa màn hình
Ctrl + U	Chuyển cửa sổ lên bằng một nửa màn hình
Ctrl + F	Dịch chuyển cửa sổ lên phía trước bằng một màn hình
Ctrl + B	Dịch chuyển cửa sổ về phía sau một màn hình
k hoặc up arrow	Dịch chuyển con trỏ lên một dòng
j hoặc down arrow	Dịch chuyển con trỏ xuống một dòng
l hoặc right arrow	Dịch chuyển con trỏ sang phải một ký tự
h hoặc left arrow	Dịch chuyển con trỏ sang trái một kí tự
Return	Dịch chuyển con trỏ đến vị trí bắt đầu dòng tiếp theo
-	Dịch chuyển con trỏ đến vị trí bắt đầu của dòng trước
w	dịch chuyển con trỏ đến vị trí bắt đầu của từ tiếp theo
b	dịch chuyển con trỏ đến vị trí bắt đầu của từ trước
^ hoặc 0	dịch chuyển con trỏ đến vị trí bắt đầu của dòng hiện tại
$	dịch chuyển con trỏ đến vị trí kết thúc của dòng hiện tại

i,a	Chèn văn bản ngay trước/sau vị trí con trỏ
o	Mở một dòng mới ngay sau dòng hiện tại
O	Mở một dòng mới ngay trước dòng hiện tại
x	Xóa ký tự sau con trỏ
dw	Xoá một từ (bao gồm cả ký tự trống ngay sau nó)
D	Xoá từ vị trí con trỏ đến kết thúc dòng
d^	Xoá từ vị trí bắt đầu dòng đến vị trí ký tự trống hay ký tự bên trái con trỏ
u	Huỷ bỏ thay đổi trước đó
/pattern	Tìm xâu pattern. Theo hướng tiến.
?pattern	Tìm xâu pattern, theo hướng lùi về đầu văn bản.
n,N	Lặp lại việc tìm kiếm theo cùng hướng / ngược hướng
p, P	Dán đoạn văn bản vừa xoá vào trước / sau con chạy
.	Lặp lại câu lệnh cuối.
dd	Xóa dòng có con trỏ chạy
:w	Ghi lại tất cả các thay đổi của file hiện tại và tiếp tục soạn thảo
:q!	Kết thúc, không lưu trữ bất kỳ thay đổi
:ZZ	Lưu thay đổi của file hiện tại và kết thúc.

2.2. Tiện ích mc.

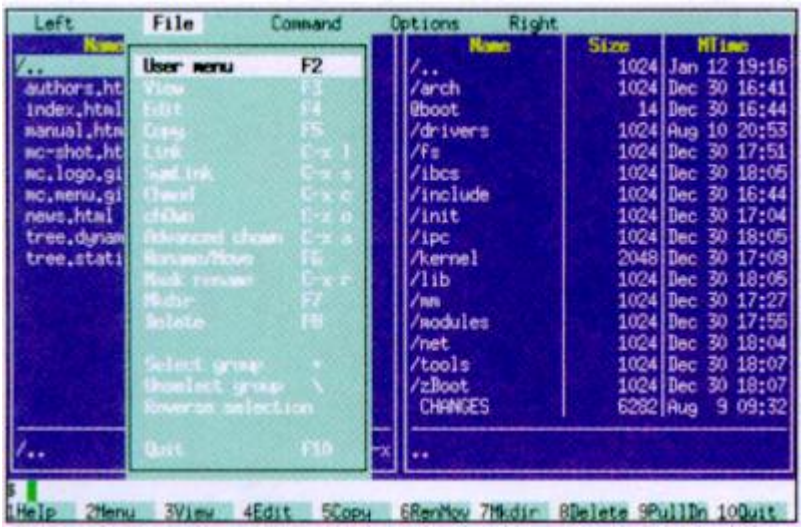
Một khi người dùng có ác cảm với giao diện dòng lệnh của DOS, họ cho rằng các lệnh của Linux cũng khó học. Trong thời kỳ của DOS trước Windows, việc định hướng các tập tin thông qua hệ thống menu và các chương trình quản lý bắt đầu phát triển mạnh, cho dù chúng chỉ dựa trên chế độ text. Một trong số chương trình thông dụng như vậy là Norton Commander.

Linux cũng có một chương trình tiện ích với chức năng tương tự như vậy gọi là Midnight Commander (MC). Bạn không phải mất công tìm kiếm MC, phần lớn các nhà phân phối Linux đều cung cấp kèm theo HĐH và nó được cài trong /usr/bin/mc. Chương trình chạy ở cả hai chế độ: text mode và đồ họa (Xterm dưới X Windows).

Sau khi nhập lệnh "mc" để chạy chương trình, bạn sẽ nhìn thấy một cửa sổ được chia đôi như trong hình 1. Midnight Commander hầu như là bản sao của Norton

Commander. Phần lớn cách trình bày, phím tắt và các đặc tính đều giống NC. Sử dụng mouse cũng được hỗ trợ ở chế độ text.

Nếu driver mouse được tải khi khởi động (phần lớn các nhà cung cấp Linux đều làm như vậy), bạn có thể dùng mouse để truy cập menu và các tập tin. Nhấn vào file thực thi để chạy, nhấn vào thư mục để chuyển vào đó, hoặc nhấn vào tập tin với phần đuôi mở rộng để mở nó với chương trình tương ứng. Bằng cách nhấn nút phải chuột vào một tập tin, bạn chọn hoặc bỏ chọn tập tin đó. Bạn có thể thực hiện tìm tên file bằng nhấn tổ hợp phím Ctrl-S và trên file với Alt. Sau đây là những phím lệnh cơ bản:



F1: Trợ giúp

F2: Menu người dùng

F3: Xem các tập tin được chọn

F4: Hiệu đính tập tin

F5: Copy tập tin

F6: Đổi tên, chuyển tập tin

F7: Tạo thư mục

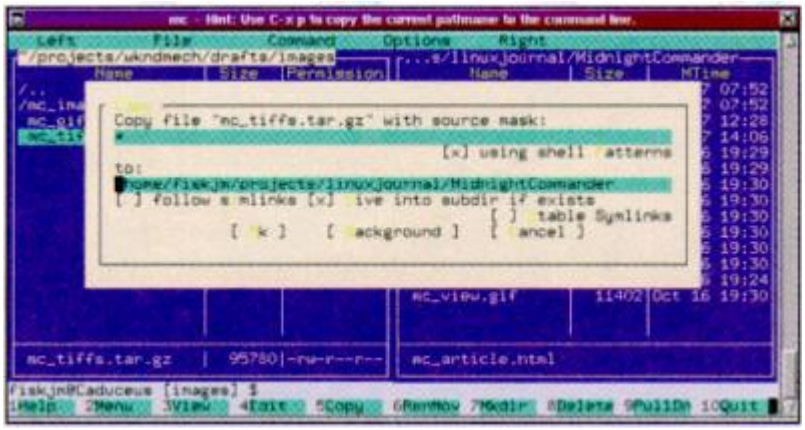
F8: Xoá tập tin

F9: Gọi menu thả xuống (pull-down)

F10: Thoát khỏi Midnight Commander

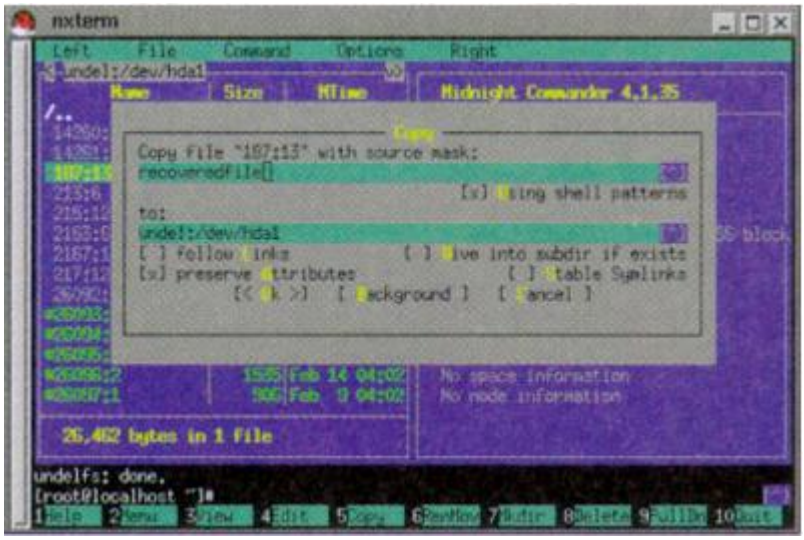
Midnight Commander hỗ trợ một số hệ thống tập tin ảo, nghĩa là bạn có thể xem file không chỉ trên các đĩa cứng cục bộ. Bạn cũng có thể xem các kiểu tập tin nén khác nhau, như .tar, .tgz, .zip, .lha, .rar, .zoo và thậm chi cả .rpm và .deb (các dạng thức tập tin nén của Red Hat và Debian. Việc xem các tập tin được thực hiện thông qua hệ

thống tập tin mạng của UNIX (UNIX Network File System - NFS), Midnigh Commander có thể hoạt động như một máy khách ftp bằng cách đưa liên kết FTP vào menu.



Có thể hồi phục các tập tin đã xóa trong Linux?

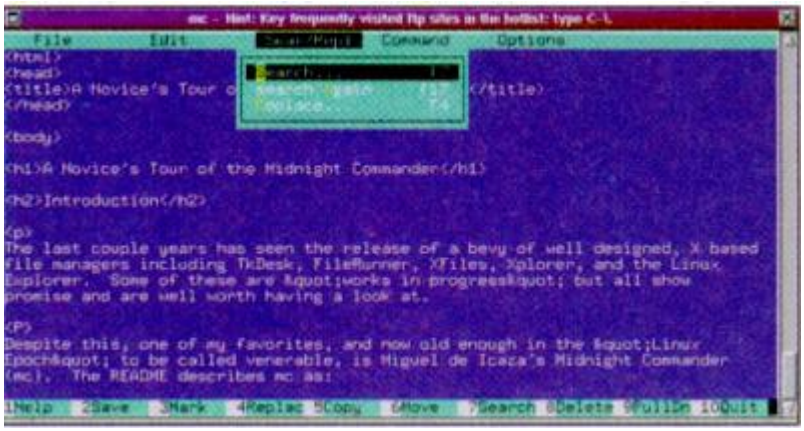
Midnight Commander cho thấy rằng vấn đề chúng ta nói đến trong phần trước (PC World VN số 7/1999 trang 95) - không có cách nào hồi phục được các tập tin bị xoá trong Linux - là không hoàn toàn chính xác. Nếu bạn sử dụng phần mở rộng ext2, hệ thống tập tin cơ bản trong Linux và cấu hình hệ thống để cho phép hồi phục tập tin bị xóa thì trên thực tế bạn có thể truy cập vào các file đã xóa.



Với Midnight Commander, bạn nhập dòng "undel:/" trước tên tập tin, ví dụ "undel:/dev/hda1". Sau đó bạn có thể xem các tập tin bị xóa (hình 3). Chọn tập tin bạn muốn hồi phục bằng chuột hay bàn phím và dùng F5 để copy chúng vào thư mục đích nào đó. Trở ngại duy nhất ở đây là thông tin về tên file bị mất, bởi vậy bạn phải cố xác định được tập tin nào bạn muốn hồi phục.

Midnight Commander bao gồm cả chương trình xem và soạn thảo tập tin. Cả hai đều có thể làm việc với file văn bản và file nhị phân (text và binary) vả hiển thị các ký tự 8-bit ngoài 128 ký tự mã ASCII.

Trình soạn thảo có giao diện menu và giống Windows ở nhiều phím soạn thảo cơ bản: nhấn Shift và phím mũi tên để chọn text, nhấn Ctrl-Ins để copy text và Shift-Ins để dán text. Bạn có thể ghi macro với Ctrl-R cũng như thực hiện những tìm kiếm theo từ thông thường.



Midnight Commander có một số tính năng mà DOS không có. Bạn có thể thay đổi quyền sở hữu tập tin và xem chi tiết về quyền truy cập tập tin. MC còn có khả năng quản lý quy trình, cho phép bạn xem những quá trình đang được thực hiện ở chế độ nền, và bạn có thể dừng chúng, khởi động lại hoặc tắt chúng hoàn toàn.

Midnight Commander có rất nhiều tính năng mà không thể liệt kê hết trong bài này. Trên Internet có nhiều Web site dành riêng cho Midnight Commander, chẳng hạn như www.gnome.org/mc, bạn có thể tham khảo chi tiết hơn.

2.3. Các câu lệnh cơ bản trên Linux

2.3.1. Hiểu biết về các câu lệnh trong Linux

2.3.1.1. Sử dụng các ký tự đại diện

Khi bạn sử dụng các câu lệnh về file và thư mục, bạn có thể sử dụng các ký tự đặc biệt được gọi là các ký tự đại diện để xác định tên file, tên thư mục. Ví dụ, để đưa ra danh sách tất cả các file có tên kết thúc bằng .c, bạn sử dụng câu lệnh sau:

*ls *.c*

Kí tự * là một ký tự đại diện, khi shell thông dịch, nó sẽ thay * bằng tất cả cac tên file có kết thúc bằng .c. Bảng bên dưới chỉ ra một số các ký tự đại diện thường được sử dụng:

*	Tương ứng với thứ tự bất kỳ của một hay nhiều ký tự
?	Tương ứng với một ký tự bất kỳ
[]	Tương ứng với một trong những ký tự trong ngoặc hoặc giới hạn

Ví dụ:

Jo* : Các file bắt đầu với Jo

Jo*y : Các file bắt đầu với Jo và kết thúc với y

Ut*l*s.c : Các file bắt đầu với Ut, chứa một ký tự l và kết thúc với s.c

?.h : Các file bắt đầu với một ký tự đơn, theo sau bởi .h

Doc[0-9].txt : Các file có tên Doc0.txt, Doc1.txt ….Doc9.txt

Doc0[A-Z].txt : Các file có tên Doc0A.txt, Doc0B.txt …Doc0Z.txt

2.3.1.2. Cơ bản về các biểu thức chính quy

Các biểu thức chính quy được sử dụng bởi phần lớn các câu lệnh. Chúng cung cấp một cách thuận tiện và đồng nhất để xác định các mẫu phù hợp . Chúng tương tự với các ký tự đại diện, nhưng chúng mạnh hơn rất nhiều. Chúng cung cấp một phạm vi rộng các mẫu lựa chọn. các ký tự đặc biệt được đưa ra ở dưới đây là các biểu thức chính quy thường được sử dụng:

Ký tự	Ý nghĩa
.	Tương ứng với một ký tự đơn bất kỳ ngoại trừ dòng mới
*	Tương ứng với không hoặc nhiều hơn các ký tự đứng trước
^	Tương ứng với bắt đầu của một dòng
$	Tương ứng với kết thúc một dòng
\<	Tương ứng với bắt đầu một từ
\>	Tương ứng với kết thúc một từ
[]	Tương ứng với một trong các ký tự bên trong hoặc một dãy các ký tự
[^]	Tương ứng với các ký tự bất kỳ không nằm trong ngoặc
\	Lấy ký hiệu theo sau dấu gạch ngược

Trước tiên, trong một biểu thức chính quy, một ký tự bất kỳ không có ý nghĩa riêng cho chính nó. Ví dụ để tìm kiếm các dòng chứa chữ “foo” trong file data.txt sử dụng câu lệnh sau:

grep foo data.txt

Để tìm kiếm các dòng bắt đầu bằng từ "foo", ta sử dụng câu lệnh:

grep '^foo' data.txt

Việc sử dụng dấu trích dẫn đơn nói cho shell để nguyên các ký tự và bỏ qua chúng trong chương trình. Việc sử dụng dấu trích dẫn đơn là cần thiết khi sử dụng các ký tự đặc biệt.

grep 'hello$' data.txt

Các dòng bất kỳ kết thúc với chuỗi "hello" được trả lại. Để tìm kiếm một mẫu bắt đầu bằng một từ, sử dụng \<. Ví dụ:

grep '\<ki' data.txt

biểu thức ở bên trên sẽ cho phép tìm kiếm các từ bắt đầu bằng 'ki' trong file data.txt. Để tìm kiếm mẫu 'wee' kết thúc của một từ, sử dụng:

grep 'wee\>' data.txt

Ở bảng bên trên, chú ý rằng dấu chấm sẽ phù hợp với một ký tự bất kỳ trừ dòng mới. Điều này có thể được thao tác, nếu chúng ta tìm kiếm tất cả các dòng chứa ký tự 'C' được theo sau bởi hai ký tự và kết thức bởi ký tự 's', biểu thức chính quy có thể là:

grep 'C..s' data.txt

Biểu thức này có thể có các mẫu phù hợp như 'Cats', 'Cars' và 'Cris' nếu chúng được chứa trong file data.txt. Nếu bạn muốn xác định một dãy các ký tự, sử dụng một dấu gạch nối phân biệt ký tự bắt đầu và ký tự kết thúc của dãy. Khi bạn xác định một dãy, thứ tự phải giống như mã ASCII. Ví dụ, để tìm kiếm tất cả các dòng chứa một ký tự "B" theo sau bởi một ký tự thường sử dụng:

grep 'B[a-z]' data.txt

Cũng có thể xaca định nhiều giới hạn trong cùng một mẫu:

grep 'B[A-Za-z]' data.txt

2.3.2. Các câu lệnh về thư mục và file

Lệnh cat

Cú pháp: cat file [>|>] [destination file]

Lệnh cat sẽ hiển thị nội dung của một file ra thiết bị ra chuẩn. Nó thường hữu ích để kiểm tra nội dung của một file bằng sử dụng câu lệnh cat. Đối số mà bạn đưa vào lệnh cat là file bạn muốn xem. Để xem toàn bộ nội dung của một file:

cat name

Lệnh cat cũng có thể trộn nhiều file đang tồn tại vào một file:

cat name1 name2 name3 > allnames

Ví dụ này sẽ kết hợp các file : name1, name2 và name3 cho file cuối cùng allnames. Thứ tự của việc trộn được thiết lập bởi thứ tự của các file được đưa vào trên dòng lệnh. Sử dụng lệnh cat, chúng ta có thể bổ sung một file vào một file khác đang tồn tại. Trong trường hợp bạn quên thêm name4 vào câu lệnh trước, chúng ta vẫn có thể nhận được kết quả mong muốn bằng cách thực hiện lệnh:

cat name4 > allnames

Lệnh này sẽ bổ sung nội dung của file name4 vào allnames

Lệnh chmod

Cú pháp: chmod [-R] permission-mode file hoặc thư mục

Lệnh chmod dùng để thay đổi quyền truy cập file hoặc thư mục. Ví dụ:

chmod myscript.pl

Để thay đổi quyền của một thư mục và tất cả các file, các thư mục con của thư mục đó sử dụng câu lệnh:

chmod –R 744 public_html

Lệnh chown

Cú pháp: chown [-fhR] Owner [:Group] { file …| thư mục… }

Lệnh chown thay đổi quyền sở hữu file hay thư mục. Giá trị của khai báo Group có thể la một ID của nhóm người sử dụng hoặc tên của nhóm người sử dụng được tìm thấy trong file /etc/group. Chỉ người sử dụng root mới có quyền thay đổi quyền sở hữu đối với file. Chi tiết về các tuỳ chọn được chỉ ra ở bên dưới:

-f : ngăn chặn tất cả các thong báo lỗi trừ các thong báo sử dụng

-h: thay đổi quyền sở hữu của lien kết tượng trưng nhưng không thay đổi quyền sở hữu của file mà được chỉ đến bởi lien kết tượng trưng đó.

-R: thay đổi quyền sở hữu của thư mục, các file và các thư mục con bên trong thư mục hiện tại được chỉ ra

Lệnh clear

Xoá màn hình, trả lại dấu chắc dòng lệnh ở phía trên của màn hình

clear

lệnh cmp

Cú pháp: cmp [-ls] file1 file2

Lệnh này so sánh nội dung của hai file. Nếu không có sự khác nhau nào, lệnh cmp sẽ kết thúc một cách yên lặng, tuỳ chọn –l sẽ n ra số byte và các giá trị khác nhau giữa hai file. Tuỳ chọn –s không hiển thị cài gì cả, nó chỉ trả lại trạng thái chỉ ra rằng sự tương đương giữa hai file. Giá trị 0 được trả lại nếu các file giống hệt nhau, giá trị bằng 1 nếu hai file khác nhau và lớn hơn 1 nếu lỗi xuất hiện khi thực hiện câu lệnh.

Lệnh cp

Cú pháp: cp [-R] file_hoặc_thư_mục file_hoặc_thư_mục

Lệnh cp sẽ sao chép một file từ thư mục nguồn đến thư mục đích được đưa vào. Để sao chép toàn bộ các file và các thư mục con bên trong thư mục mong muốn, bạn sử dụng câu lệnh cp với tuỳ chọn –R

Lệnh du

Lệnh này tổng kết việc sử dụng đĩa. Nếu bạn xác định một thư mục, lệnh du sẽ báo cáo việc sử dụng đĩa cho chính các thư mục đó.

Cú pháp: du [-ask] tên_file

Tuỳ chọn –a sẽ đưa ra màn hình kích thước của mỗi thư mục và file

Tuỳ chọn –s sẽ chỉ in ra tổng cộng

Tuỳ chọn –k sẽ in ra tất cả các kích thước file theo kilobytes

Lệnh file

Cú pháp: file filename

Câu lệnh xác định kiểu của file. Nếu file không phải là file thông thường, kiểu của file được xác định.

Lệnh find

Câu lệnh find tìm các file và các thư mục.

Cú pháp : find [path] [-type fd] [-name mẫu] [-atime [+-] số_ngày] [-exec câu_lệnh {} \;] [-empty].

Ví dụ:

find . –type d

Câu lệnh trả lại tất cả các thư mục con trong thư mục hiện tại. Tuỳ chọn –type xác định kiểu, d cho các thư mục, f cho các file hay l cho các lien kết.

find . –type f –name ".txt"*

Lệnh này sẽ tìm tất cả các file văn bản có phần mở rộng ".txt" trong thư mục hiện tại và cả trong các thư mục con.

find . –type f –name ".txt" –exec grep –l 'magic' {} \;*

Câu lệnh này sẽ tìm kiếm tất cả các file văn bản (kết thúc với phần mở rộng .txt) trong thư mục hiện tại và các thư mục con có chứa từ "magic".

find . –type f empty

Hiển thị tất cả các file rỗng trong thư mục hiện tại.

Lệnh grep

Cú pháp: grep [–viw] mẫu file

Lệnh grep cho phép bạn tìm kiếm một hoặc nhiều file có các mẫu ký tự đặc biệt. Mỗi dòng của mỗi file chứa các mẫu được hiển thị trên màn hình. Câu lệnh grep hữu ích khi bạn có nhiều file và bạn muốn tìm ra file chứa từ hoặc câu xác định. Sử dụng tuỳ chọn –v, bạn có thể hiển thị các file không chứa một mẫu. Ví dụ, để chọn các dòng trong data.txt không chứa từ "the" ta thực hiện:

grep –vw 'the' data.txt

nếu tuỳ chọn –w không được xác định thì bất kỳ các từ chứa "the" đều phù hợp như "together". Tuỳ chọn –w được xác định buộc mẫu phải là toàn bộ một từ. Cuối cùng , tuỳ chọn –i bỏ qua sự khác nhau giữa các ký tự chữ hoa và ký tự chữ thường khi tìm kiếm mẫu.

Lệnh head

Cú pháp: head [-count | -n number] filename

Câu lệnh này sẽ hiển thị vài dòng đầu tiên của một file. Bởi mặc định, 10 dòng đầu của một file được hiển thị. Tuy nhiên, bạn có thể sử dụng các tuỳ chọn để xác định số dòng hiển thị. Ví dụ:

head -2 doc.txt

sẽ hiển thị hai dòng đầu tiên.

Lệnh ln

Cú pháp: ln [-s] file_nguồn đích

Lệnh ln tạo các liên kết cứng và mềm. Các liên kết cứng được tạo sử dụng lệnh ln không có tuỳ chọn –s. Ví dụ:

ln ./www ./public_html

Một lien kết cứng có hạn chế, nó không thể tạo liên kết đến một thư mục khác, và một liên kết cứng không thể liên kết đến một file trên một hệ thống file khác. Sử dụng tuỳ chọn –s bạn có thể tạo một liên kết mềm, loại bỏ các giới hạn này.

ln –s /dev/fs02/jack/www /dev/fs01/foo/public_html

Ở đây chúng ta đã tạo một liên kết mềm giữa thư mục www trên hệ thống file 2 và một file mới được tạo trên hệ thống file 1.

Lệnh locate

Cú pháp : locate từ_khoá

Câu lệnh locate tìm đường dẫn đến một file đặc biệt hay một câu lệnh. Lệnh locate sẽ tìm kiếm chính xác hay một phần của chuỗi phù hợp. Ví dụ:

locate foo

kết quả tìm kiếm sẽ đưa ra các file có tên chứa từ khoá 'foo' theo đường dẫn tuyệt đối hoặc sẽ không đưa ra kết quả nếu không có tên file như vậy.

Lệnh ls

Lệnh ls cho phép bạn đưa ra danh sách các file và các thư mục con.

Cú pháp : ls [-1aRl] file_hoặc_thư_mục

Khi sử dụng tuỳ chọn -1 , nó chỉ hiển thị tên file và tên thư mục con của thư mục hiện tại. Khi chọn tuỳ chọn –l, một danh sách các file và thư mục con của thư mục hiện tại được hiển thị với đầy đủ các thông tin về file và thư mục. Tuỳ chọn –a cho phép bạn hiển thị tất cả các file và thư mục (kể cả các file ẩn, tên file bắt đầu bằng dấu chấm) trong thư mục hiện tại. Tuỳ chọn –R sẽ hiển thị tất cả các file và các thư mục con bên trong nó nếu có.

Lệnh mkdir

Cú pháp: mkdir thư_mục

Để tạo một thư mục, sử dụng câu lệnh mkdir. Chỉ có 2 giới hạn khi chọn tên thư mục, đó là tên của thư mục có thể lên tới 255 ký tự và tên thư mục có thể chứa bất kỳ ký tự nào trừ ký tự '/'.Ví dụ:

mkdir dir1 dir2 dir3

Lệnh trên tạo ra ba thư mục, nằm bên trong thư mục hiện tại.

Lệnh mv

Cú pháp : mv [-if] file_nguồn file_đích

Sử dụng lệnh mv để dịch chuyển hay đổi tên các file hay các thư mục. Câu lệnh thực hiện việc dịch chuyển hay đổi tên phụ thuộc vào file_đích có là một thư mục hay không. Để minh hoạ, chúng ta sẽ đổi tên một thư mục foo thành foobar:

mv foo foobar

Bởi vì foobar chưa tồn tại, foo sẽ được đổi tên thành foobar. Nếu câu lệnh sau được thực hiện:

mv doc.txt foobar

và foobar đã tồn tại, việc dịch chuyển file sẽ được thực hiện sau đó. Tuỳ chọn –f sẽ xoá các file đích đang tồn tại và không bao giờ nhắc người sử dụng. Tuỳ chọn –i sẽ nhắc người sử dụng có ghi đè hay không nếu file_đích đã tồn tại.

Lệnh pwd

Cú pháp: pwd

Câu lệnh này hiển thị tên thư mục hiện tại bao gồm cả đường dẫn tuyệt đối. Ví dụ:

pwd

Trên màn hình hiển thị :

/home/trantu

Lệnh rm

Cú pháp: rm [-rif] thư_mục/file

Để xoá thư mục hoặc file, sử dụng câu lệnh rm. bạn có thể xoá nhiều file sử dụng ký tự đại diện hoặc gõ vào tên các file. Ví dụ:

rm doc1.txt doc2.txt doc3.txt

Tương ứng với:

rm doc[1-3].txt

rm là câu lệnh rất mạnh, hãy cẩn thận khi sử dụng lệnh này vì bạn có thể nhầm và xoá đi các file quan trọng. Nếu chưa chắc chắn, bạn có thể sử dụng tuỳ chọn –i, hệ thống sẽ nhắc lại cho bạn xác thực mỗi lần xoá một file. Nếu như đã chắc chắn file cần xoá, bạn có thể chọn tuỳ chọn –f để không phải nhận các thông tin nhắc bạn xác thực. Tuỳ chọn –r sẽ cho phép bạn xoá toàn bộ các thư mục con.

Lệnh tail

Cú pháp: tail [-count | -fr] tên_file

Câu lệnh tail hiển thị phần cuối của một file, mặc định nó sẽ hiển thị 10 dòng cuối cùng của file. Để hiển thị 50 dòng cuối cùng của file doc.txt, bạn có thể sử dụng câu lệnh:

tail -50 doc.txt

Tuỳ chọn –r sẽ thực hiện công việc ngược lại, mặc định nó sẽ hiển thị tất cả các dòng trừ 10 dòng cuối cùng. Tuỳ chọn –f hữu ích khi bạn đang giám sát một file. Với tuỳ chọn này, tail sẽ chờ cho dữ liệu mới được ghi vào file. Khi dữ liệu mới được thêm vào file, tail sẽ hiển thị dữ liệu lên màn hình. Để dừng lệnh tail khi đang giám sát file, chọn tổ hợp phím Ctrl + C bởi vì lệnh tail không tự dừng được.

2.3.3. Các câu lệnh nén dữ liệu

Lệnh compress

Cú pháp: compress [-v] file

Câu lệnh compress sẽ cố gắng giảm kích thước của một file sử dụng. Các file được nén sẽ được thay thế bởi một file có phần mở rộng .Z. Tùy chọn –v sẽ hiển thị phần trăm dung lượng giảm của một file được nén và sẽ nói cho bạn têncủa file mới:

compress –v inbox

trên màn hình sẽ hiển thị

inbox: Compression: 37.20% - replaced with inbox.Z

Lệnh gunzip

Cú pháp: gunzip [-v] files

Để giải nén các file về dạng nguyên bản , sử dụng lệnh gunzip, sẽ cố gắng giải nén các file có phần mở rộng: .gz, -gz, .z, -z, _z, .Z, hoặc tgz. Tùy chọn –v sẽ hiển thị kết quả đẹp khi giải nén các file. Ví dụ:

gunzip –v README.txt.gz

Lệnh gzip

Cú pháp: gzip [-rv9] file

Lệnh gzip là một chương trình nén khác. Nó được biết đến là chương trình nén có tỉ lệ nén tốt nhất. các file được nén bởi lệnh gzip sẽ được thay thế bởi các file có phần mở rộng .gz. Tùy chọn –9 có tốc độ nén tốt nhất. Tùy chọn –v cho phép hiển thị đẹp trên màn hình. Kích thước, tổng số và tỉ lệ nén được đưa ra danh sách cho mỗi file. Tùy chọn –r sẽ nén tất cả các file trong mỗi thư mục theo cùng một cách.

Lệnh tar

Cú pháp: tar [c] [x] [v] [z] [f tên_file] tên_file_hoặc_thư_mục

Lệnh tar cho phép bạn nén nhiều file và thư mục vào một file .tar. Nó cũng cho phép bạn giải nén các file và các thư mục từ một file nén. Ví dụ:

*tar cf source.tar *.c*

Câu lệnh này sẽ tạo một file source.tar, chứa tất cả các file mã nguồn C (có phần mở rộng .c) trong thư mục hiện tại.

*tar cvf source.tar *.c*

Tùy chọn –v ở đây cho phép bạn xem các file đã được nén

tar cvzf backup.tar.gz important_dir

Ở đây, tất cả các file và các thư mục con của thư mục important_dir được nén trong một file được gọi là backup.tar.gz. Chú ý rằng file này cũng được nén do có tùy chọn z , và do đó kết quả là file có phần mở rộng là .gz. Thông thường phần mở rộng .tar.gz được viết ngắn thành .tgz. Để giải nén các file , ví dụ như backup.tar , bạn sử dụng câu lệnh:

tar xf backup.tar

Để giải nén một file có phần mở rộng .tgz hay .tar.gz, bạn thực hiện câu lệnh sau:

tar xzf backup.tgz

Lệnh uncompress

Cú pháp: uncompress [-v] file

Khi một file được nén sử dụng câu lệnh compress, để giải nén bạn sử dụng câu lệnh uncompress. Lệnh uncompress giải nén các file có phần mở rộng .Z, vì vậy cú pháp của nó tương tự như lệnh compress

uncompress –v inbox.Z

Lệnh unzip

Cú pháp: unzip file

Lệnh này sẽ giải nén các file có phần mở rộng .zip. Các file này có thể được nén với lệnh zip.

Lệnh zip

Cú pháp : zip [-ACDe9] file

Đây là chương trình nén file theo định dạng nổi tiếng tương thích với nhiều hệ điều hành. Các file được nén với lệnh zip có phần mở rộng .zip.

Lệnh mount

Cú pháp: mount –a [-t fstype] [-o option] device directory

Lệnh mount được sử dụng để gán các thiết bị với hệ thống, các tùy chọn thông thường thường có trong file /etc/fstab. Ví dụ:

/dev/hda6 /intranet ext2 defaults 1 2

Nếu dòng bên trên được tìm thấy trong /etc/fstab, bạn có thể gắn hệ thống file được lưu trong phân vùng /dev/hda6 như sau:

mount /intranet

Cùng một hệ thống file, câu lệnh sau đây là tương tự:

mount –t ext2 /dev/hda6 /intranet

Tùy chọn –t được sử dụng để xác định kiểu file hệ thống. Để gắn tất cả các hệ thống file có trong /etc/fstab sử dụng tùy chọn –a. Ví dụ:

mount –a –t ext2

Thông thường người sử dụng chọn tùy chọn –o là ro (chỉ đọc) hoặc rw (đọc ghi). Ví dụ:

mount –t ext2 –o ro /dev/hda6 /secured

- **Lệnh umount**

Cú pháp : umount –a [-t fstype]

Lệnh umount ngược lại với lệnh mount. Ví dụ

umount /cdrom

2.3.4. Các câu lệnh quản lý tiến trình

- **Lệnh bg**

Cú pháp: bg

Đây là kịch bản shell được xây dựng sẵn. Đưa một tiến trình đang chạy về chạy ở sau hậu cảnh (tiến trình nền).

- **Lệnh fg**

Cú pháp: fg [%job-number]

Câu lệnh này cho phép bạn chuyển một tiến trình nền lên chạy ở trên tiền cảnh.

Nếu bạn chạy câu lệnh này không có bất kỳ đối số nào, nó sẽ đưa câu lệnh cuối cùng ở sau hậu cảnh lên hiển thị. Ví dụ, nếu có hai câu lệnh chạy ở sau hậu cảnh, bạn có thể chuyển câu lệnh thứ nhất lên chạy trên tiền cảnh bằng câu lệnh:

fg %1

Lệnh jobs

Cú pháp: jobs

Lệnh này cho phép bạn hiển thị các tiến trình nền đang chạy. Ngoài ra còn một số lệnh sẽ được trình bày trong các phần sau.

3. Giới Thiệu Hệ Thống Tập Tin, Thư Mục

3.1. Giới thiệu

Trong linux file được tổ chức thành các thư mục, theo mô hình phân cấp. Tham chiếu đến một file bằng tên và đường dẫn. Các câu lệnh thao tác file cho phép thực hiện các chức năng như dịch chuyển, sao chép toàn bộ thư mục cùng với các thư mục con chứa trong nó…

Có thể sử dụng các ký tự, dấu gạch dưới, chữ số, dấu chấm và dấu phảy để đặt tên file. Không được bắt đầu một tên file bằng dấu chấm hay chữ số. Những ký tự khác như '/', '?', '*', là ký tự đặc biệt được dành riêng cho hệ thống. Chiều dài của tên file có thể tới 256 ký tự.

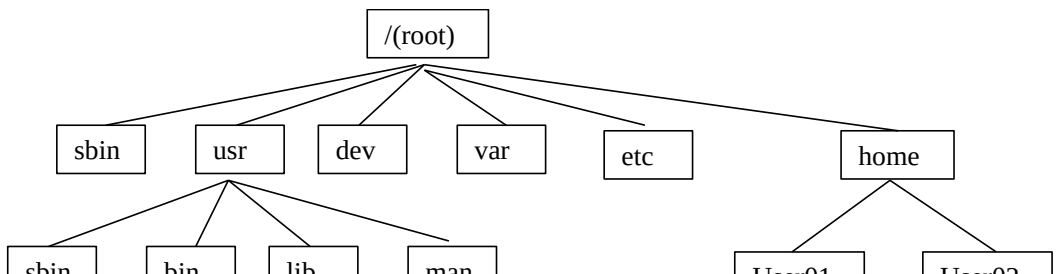
Tất cả các file trong linux có chung cấu trúc vật lý là chuỗi các byte (byte stream). Cấu trúc thống nhất này cho phép linux áp dụng khái niệm file cho mọi thành phần dữ liệu trong hệ thống. Thư mục cũng như các thiết bị được xem như file. Chính việc xem mọi thứ như các file cho phép linux quản lý và chuyển đổi dữ liệu một cách dễ dàng. Một thư mục chứa các thông tin về thư mục, được tổ chức theo một định dạng đặc biệt. Các thành phần được xem như các file, chúng được phân biệt dựa trên kiểu file: *ordinary file, directory file, character device file*, và *block device file.*

3.1.1. Thư mục chủ

Sau khi đăng nhập hệ thống, người dùng sẽ đứng ở thư mục chủ. Tên của thư mục này giống với tên tài khoản đăng nhập hệ thống. Các file được tạo khi người dùng đăng nhập được tổ chức trong thư mục chủ.

3.1.2. Các thư mục hệ thống

Thư mục root, là gốc của hệ thống file của Linux, chứa một vài thư mục hệ thống. Thư mục hệ thống chứa file và chương trình sử dụng để chạy và duy trì hệ thống. Biểu diễn các thư mục như sau:



Mô tả thư mục

Thư mục	Chức năng
/	Bắt đầu cấu trúc file, gọi là thư mục gốc *(root)*
/home	Chứa thư mục gốc (*home*) của người dùng
/bin	lưu chữ tất cả các câu lệnh chuẩn và các chương trình tiện ích
/usr	chứa các file, câu lệnh được hệ thống sử dụng, thư mục này được chia thành các thư mục con khác
/usr/bin	Chứa các câu lệnh hướng người dùng và các chương trình tiện ích
/usr/sbin	Chứa các câu lệnh quản trị hệ thống
/usr/lib	Chứa thư viện cho các ngôn ngữ lập trình
/usr/doc	Chứa tài liệu của linux
/usr/man	Chứa các file chỉ dẫn cho các câu lệnh (man)
/sbin	Chứa các file hệ thống để khởi động hệ thống
/dev	Chứa giao diện cho các thiết bị như đầu cuối và máy in
/etc	Chứa file cấu hình hệ thống và các file hệ thống khác

3.2. Các quyền truy cập file/thư mục

Trong Linux, mỗi file hay thư mục được kết hợp với một người sử dụng và một nhóm người sử dụng. Hãy xem một ví dụ:

```
-rwxr-x-r--   1 trantu   trantu        191 Apr 14 14:55 .bash_profile
```

Dòng bên trên được tạo bởi lệnh *ls –l .bash_profile* trên hệ điều hành Linux. Lệnh ls đưa ra danh sách các file và thư mục. Tùy chọn –l đưa ra danh sách đầy đủ các thông tin về file .bash_profile. Bảng bên dưới mô tả các kiểu thông tin đưa ra:

Kiểu thông tin	Thông tin kết xuất
Quyền truy cập file	-rw-rw-r--
Số liên kết	1
Người sử dụng (sở hữu file)	Trantu
Nhóm sử dụng	Trantu
Kích thước file (theo bytes)	191
Ngày sửa đổi sau cùng	Apr 14
Thời gian sửa đổi sau cùng	14:55
Tên file	.bash_profile

Ở đây, người sử dụng là trantu. Đây là người sử dụng thường xuyên, có quyền thay đổi các quyền trùy cập đối với file này. Chỉ có một người sử dụng khác có quyền thay đổi thuộc tính file này, đó là superuser. Nhóm sử dụng file này là trantu, bất kỳ những người sử dụng nào thuộc nhóm trantu cũng có quyền đọc, và thực thi dựa vào quyền của nhóm được đặt bởi người sở hữu. Khi bạn tạo một file trên hệ thống Linux, hệ thống sẽ mặc định người sở hữu file này có tên là tên đăng nhập của bạn và có tên nhóm giống như tên của người sở hữu. Một người sử dụng thông thường không thể gán lại quyền sở hữu một file hay thư mục cho người khác. Ví dụ, bạn không thể tạo một file với người sử dụng *kabid* rồi sau đó gán lại quyền sở hữu cho người khác có tên là *sheila* bởi lý do bảo mật. Nếu một người sử dụng thông thường có quyền gán quyền sở hữu file cho người khác, thi một ai đó cúng có thể tạo một chương trình xấu như xóa các file, và thay đổi quyền sở hữu cho superuser, và không biết điều gì sẽ xảy ra. Chỉ có người superuser mới có thể gán lại quyền sở hữu file hay thư mục cho người khác.

3.2.1. Thay đổi quyền sở hữu file, thư mục sử dụng lệnh chown

Người sử dụng superuser có thể thay đổi quyền sở hữu file, thư mục cho một người sử dụng khác. Để thay đổi quyền sở hữu sử dụng câu lệnh sau:

chown **newuser** **file hoặc thư mục**

Ví dụ:

chown trantu example.txt

Câu lệnh này làm cho người sử dụng trantu có quyền sở hữu file example.txt

Nếu superuser muốn thay đổi nhóm cho một file hoặc thư mục, người đó có thể sử dụng câu lệnh chown như sau:

chown **newuser.newgroup file hoặc thư mục**

Ví dụ

chown trantu.admin example.txt

Câu lệnh trên không chỉ thay đổi quyền sở hữu file cho trantu mà còn đặt lại nhóm sử dụng file là admin. Nếu superuser muốn thay đổi người sở hữu và nhóm sử dụng cho tất cả các file trong một thư mục, người đó có thể sử dụng câu lệnh chown với tùy chọn –R. Ví dụ

chown –R trantu.admin /home/trantu/

3.2.2. Thay đổi nhóm sử dụng file/thư mục với lệnh chgrp

Câu lệnh chgrp cho phép bạn thay đổi quyền sử dụng file hay thư mục của một nhóm, chỉ nếu bạn thuộc về cả hai nhóm (nhóm cũ và nhóm mới). Ví dụ:

*chgrp httpd *.html*

Lệnh trên sẽ thay đổi nhóm sử dụng cho tất cả các file có phần mở rộng html. Bạn chỉ có thể thay đổi được nếu bạn thuộc nhóm httpd. Giống như lệnh chown, lệnh chgrp cũng có tùy chọn –R để thay đổi quyền với nhiều file hay thư mục.

3.2.3. Sử dụng số theo hệ cơ số 8 tương ứng với thuộc tính truy cập

Hệ cơ số 8 sử dụng 8 số (0-7), và mỗi số tương ứng với 3 bit (theo hệ nhị phân). Bảng bên dưới chỉ cho bạn thấy sự tương ứng về quyền với số hệ cơ số 8.

		Số thứ 1	Số thứ 2	Số thứ 3	Số thứ 4
Giá trị cơ số 8	4	set-UID	R	r	r
	2	set-GID	W	w	w
	1	sticky-bit	X	x	x
		Special	User	Group	Others

Như ở trên bảng trên, số thứ nhất được sử dụng cho việc thiết lập các quyền đặc biệt, số thứ hai được sử dụng cho việc thiết lập người sở hữu file hay thư mục. Số thứ ba được sử dụng để thiết lập quyền cho nhóm người sử dụng và số thứ tư được sử dụng để thiết lập quyền cho tất cả mọi người. Khi bất kỳ một số nào bị bỏ qua, nó được xem như nhận giá trị 0. Bảng bên dưới chỉ ra một vài ví dụ về các giá trị tương ứng với quyền:

Giá Trị	Giải Thích
0400	Chỉ có quyền đọc cho người sở hữu, nó tương ứng với 400.
0440	Chỉ có quyền đọc với người sở hữu và nhóm người sử dụng. Nó tương ứng với giá trị 440.
0444	Quyền đọc cho tất cả mọi người. Nó tương ứng với giá trị 444
0644	Người sở hữu có quyền đọc và ghi, tất cả mọi người có quyền đọc, tương ứng với giá trị 644. (6 là tọa bởi 4:r và 2:w)
0755	Đọc ghi và thực thi đối với người sử dụng, đọc và thực thi đối với tất cả mọi người. (7 là tạo bởi 4:r , 2:w và 1:x)
4755	Nó tương ứng với giá trị 755 ngoại trừ file này được đặt giá trị set-UID = 4. Điều này có nghĩa là khi file được thực thi, nó có tất cả các quyền của người sở hữu để thực hiện công việc. Sẽ là một lỗ hổng lớn nếu người sở hữu ấy là root và những người khác có quyền thực thi file này. Hãy cẩn thận khi thiết lập giá trị của set-UID.
2755	Nó tương tự với giá trị 755 ngoại trừ, khi thực thi nó có tất cả các quyền của nhóm sử dụng file.

Để thiết lập quyền phù hợp, bạn nên chỉ ra kiểu truy cập của người sử dụng, nhóm người sử dụng và của những người khác.

3.2.4. Sử dụng ngôn ngữ tự nhiên tương ứng với quyền truy cập

Bây giờ chúng ta sẽ sử dụng xâu truy cập đơn giản hơn việc sử dụng số. Bảng bên dưới chỉ ra các xâu truy cập tương ứng với các quyền:

read (r)	read (r)	read (r)	read (r)
write (w)	write (w)	write (w)	write (w)
execute (x)	execute (x)	execute (x)	execute (x)
Special	**User**	**Group**	**Others**
	all (a)		

Mỗi kiểu quyền tương ứng với một ký tự đơn (trong dấu ngoặc).

3.2.5. Thay đổi quyền truy cập file thư mục sử dụng lệnh chmod

Tiện ích chmod cho phép bạn thay đổi các quyền. Bạn có thể sử dụng các chữ số hay các ký tự với tiện ích này để thay đổi quyền. Ví dụ

*chmod 755 *.pl*

Câu lệnh trên thay đổi quyền cho các file có phần mở đuôi là .pl. Mỗi một file .pl được đặt các quyền đọc, ghi và thực thi bởi người sở hữu, các file cúng có thể đọc và thực thi bởi nhóm người sử dụng và những người khác. Bạn có thể hoàn thành cùng một công việc như vậy với lệnh sau:

*chmod a+rx,u+w *.pl*

a+rx được sử dụng để cho phép tất cả mọi người đọc và thực thi đối với mỗi file .pl và u+w được sử dụng để cho phép người sở hữu có quyền ghi đỗi với mỗi file .pl.

Nếu bạn muốn thay đổi các quyền cho tất cả các file và các thư mục con trong một thư mục, bạn có thể sử dụng tùy chọn –R:

chmod –R 750 /www/mysite

3.2.6. Các chú ý đặc biệt trên các quyền thư mục

Các quyền thiết lập cho một thư mục cũng tương tự như các file thông thường, nhưng không giống hệt nhau. Dưới đây là một vài chú ý đặc biệt trên các quyền thư mục:

- Quyền chỉ đọc cho một thư mục sẽ không cho phép bạn chuyển vào bên trong thư mục, để chuyển vào bên trong bạn cần có quyền thực thi
- Quyền chỉ được thực thi sẽ cho phép bạn truy cập vào các file bểntong một thư mục khi bạn biết tên của chúng và bạn được phép đọc chúng.
- Để có thể đưa ra danh sách nội dung của một thư mục sư dụng câu lệnh tương tự như ls và cúng có thể chuyển vào bên trong thư mục bạn cần có cả quyền đọc và quyền thực thi đối với thư mục đó
- Nếu bạn có quyền ghi cho một thư mục, bạn có thể tạo, thay đổi, xóa các file bất kỳ hay các thư mục con bất kỳ bên trong thư mục đó ngay cả khi file và thư mục con được sở hữu bởi người khác

3.3. Tạo một chính sách quyền cho một server nhiều người sử dụng

3.3.1. Thiết lập cấu hình các quyền truy cập file của người sử dụng

Trong thư mục của mỗi người sử dụng có một vài file ẩn chung bắt đầu với dấu chấm (.). Các file này thường được sử dụng để thực thi các câu lệnh tại thời điểm người sử dụng đăng nhập. Ví dụ, tất cả các shell (csh, tcsh, bash, …) sẵn sàng cho một người sử dụng đọc các thiết lập của họ từ một file giống như .cshrc hay .bashrc. Nếu một người sử dụng không cẩn thận trong việc giữ quyền các file một cách hoàn hảo, một người sử dụng không thân thiện khác có thể gây ra các vấn đề không mong muốn.. Ví dụ, nếu một file .cshrc của người sử dụng có thể được viết bởi người khác, người su có thể chơi một trò tấn công ngu ngốc như đưa một câu lệnh logout ngay dòng đầu của file .cshrc, như vậy người sử dụng sẽ thoát ngay khi đăng nhập vào hệ thống. Nếu

bạn có quyền thao tác với những người sử dụng bạn có thể thực hiện nhanh chóng việc kiểm tra đơn giản sau:

*find /home -type f -name ".*rc" -exec ls -l {} \;*

Câu lệnh này sẽ hiển thị quyền của tất cả các file có ký tự đầu tiên là dấu chấm, kết thúc bằng "rc" nằm trong thư mục home

3.3.2. Thiết lập mặc định các quyền truy cập file cho người sử dụng

Là người quản trị bạn cần định nghĩa các quyền mặc định thiết lập cho tất cả các file của người sử dụng đưa vào hệ thống của bạn. Để thiết lập mặc định quyền cho các file mới, bạn có thể sử dụng cầu lệnh umask như sau:

*umask **mask***

Để hiểu từ umask như thế nào, hãy xem ví dụ sau. Khi nói rằng umask đặt là 022, file mới được tạo , thông thường một quyền 0666 được yêu cầu bởi hàm tạo file – open. Tuy nhiên, trong trường hợp này , quyền cuối cùng thiết lập cho các file được tạo bởi hệ thống như sau: 0666 được thực hiện phép toán AND với phần bù của 022 (phần bù của 022 là 755) do đó kết quả của phép AND thu được là 0644, nó cho phép người sở hữu đọc và ghi còn những người khác chỉ có quyền đọc. Để tạo một mask mặc định cho cacs quyền truy cập file, bạn có thể nhúng câu lệnh umask vào một shell tài nguyên chung trong /etc để khi một người sử dụng đăng nhập và chạy một shell, file tài nguyên shell chung sẽ được thực thi. Ví dụ, nếu người sử dụng của bạn sử dụng shell /bin/csh hay /bin/tcsh, bạn có thể đưa một câu lệnh umask mong muốn trong file /etc/csh.cshrc cho mục đích này.

3.3.3. Thiết lập các quyền có thể thực thi cho các file

Các file chương trình có thể được chạy bởi những người sử dụng thông thường không bao giờ nên đặt quyền được ghi cho bất kỳ ai khác ngoài người sở hữu. Ví dụ, các file chương trình trong /usr/bin nên thiết đặt các quyền như chỉ root có quyền đọc, ghi và thực thi và tất cả mọi người chỉ có quyền đọc và thực thi các file này. Việc cho phép người khác ghi có thể tạo ra một lỗ hổng nghiêm trọng cho hệ thống.

3.4. Làm việc với các file và các thư mục

3.4.1. Xem các file và các thư mục

Bạn có thể đã quen với lệnh ls, thông thường nó được sử dụng với các tùy chọn –l (long listing) hiển thị đầy đủ thông tin, -a hiển thị tất cả các file bao gồm cả các file bắt đầu bằng dấu chấm và –R hiển thị tất cả các file và các thư mục con bên trong thu mục mong muốn

3.4.2. Chuyển đến thư mục

Bạn gần như đã quen với câu lệnh cd, nó là một shell xây dựng sẵn. Nếu bạn không cung cấp một tên thu mục bất kỳ làm đối số cho nó, nó sẽ chuyển về thư mục chủ của

bạn mà hiện tại bạn đang sử dụng. Khi bạn đang đứng ở bất kỳ đâu trong hệ thống file, bạn có thể sử dụng lệnh pwd để hiển thị đường dẫn đến thư mục hiện tại.

3.4.3. Xác định kiểu file

Không giống như hệ điều hành Windows, Linux không dựa vào phần mở rộng của file để xác định kiểu file. Bạn có thể sử dụng tiện ích file để xác định kiểu file trong hệ thống. Ví dụ:

file todo.txt

Kết quả hiển thị như sau:

todo.txt: ASCII text

3.4.4. Xem thống kê các quyền của file hay thư mục

Bạn có thể sử dụng lệnh stat để lấy thống kê về các file và các thư mục:

stat ./exam

Kết quả hiển thị trên màn hình

File: "./exam"
Size: 4096 Blocks: 8 IO Block: -4611692478058196992 Directory
Device: 812h/2066d Inode: 157762 Links: 2
Access: (0755/drwxr-xr-x) Uid: (0/ root) Gid: (0/ root)
Access: Wed Jun 18 14:56:48 2003
Modify: Wed Jun 18 11:18:42 2003
Change: Wed Jun 18 11:18:42 2003

3.4.5. Sao chép file và thư mục

Sử dụng câu lệnh cp để sao chép từ một vị trí xác định đến vị trí khác:

cp /some/important /new/place

Bạn cũng có thể xác định một tên mới cho file sao chép. Thông thường lệnh cp được sử dụng với tùy chọn –f để sao chép file từ nguồn đến đích mà không quan tâm đến đến việc có một file cùng tên tồn tại ở đích. File mới sẽ được sao chép đè lên file cũ. Để sao chép một thư mục đến một thư mục khác bạn thực hiện lệnh cp với tùy chọn –r ví dụ:

cp –r /tmp/foo /zoo/foo

3.4.6. Dịch chuyển các file và thư mục

Để dịch chuyển các file hay thư mục sử dụng câu lệnh mv. Ví dụ, để chuyển /file1 vào /tmp/file2 ta sử dụng câu lệnh sau:

mv /file1 /tmp/file2

3.4.7. Xóa các file và thư mục

Để xóa cac file và thư mục sử dụng lệnh sau:

rm filename

Khi xóa hệ thống sẽ hỏi bạn có thự sự muốn xóa hay không. Nếu bạn đã chắc chắn file bạn muốn xóa bạn có thể thực hiện lệnh xóa rm với tùy chọn –f để không hiện ra thông tin yêu cầu xác nhận của hệ thống. Để xóa một thư mục, bạn cần thực hiện lệnh rm với tùy chọn –r

3.4.8. Tìm kiếm file

Để xác định vị trí chính xác của một filem, bạn có thể sử dụng lệnh which. Ví dụ:

which httpd

Câu lệnh này sẽ chỉ ra cho bạn đầy đủ đường dẫn của chương trình httpd nếu nó sẵn có. Bạn cũng có thể xác định một phần của tên file hay thư mục sử dụng lệnh locate

locate netpr.pl

4. Quản lý người dùng và tài nguyên

4.1. Khái niệm

Linux là hệ điều hành đa nhiệm và đa người dùng. Mỗi người dùng có tên truy nhập và mật khẩu riêng, tương ứng với những quyền hạn nhất định trong hệ thống file của Linux.

Để tạo điều kiện thuận lợi trong quản lý người dùng và quyền hạn đối với hệ thống file, Linux cho phép khai báo những nhóm người dùng, mỗi nhóm là một tập hợp những người dùng chung một mục đích khai thác tài nguyên nhất định. Mỗi người dùng có thể tham gia nhiều nhóm người dùng khác nhau. Mỗi người dùng cũng mặc

nhiên lập nên một nhóm người dùng là nhóm của chính họ (nhóm có thể chỉ có một thành viên).

Người dùng có toàn quyền trong Linux là người dùng root, mặc nhiên thuộc về nhóm root. Người dùng có quyền root ấn định một người dùng nào đó thuộc về nhóm root và có quyền tương đương với root.

4.2. Trở thành superuser

Bạn đã biết rằng tài khoản root là tài khỏan superuser trong hệ thống Linux. Thực ra nếu bạn tự cài đặt hệ thống, bạn đã sử dụng tài khoản này để đăng nhập hệ thống lần đầu tiên. Bạn cũng biết rằng root là tài khoản superuser, tài khoản này có quyền làm mọi thứ trên hệ thống. Người sử dụng root có thể khởi động hay dừng một chương trình bất kỳ cũng như tạo và xóa một file bất kỳ. Rất nhiều những người mới quản trị hệ thống Linux cho rằng chỉ có root là tài khoản superuser. Hãy nhìn xuống đoạn mã bên dưới có trong file /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
vietvq:x:0:0:root:/home/vietvq:/bin/bash
xanhhh:x:0:0:root:/root:/bin/bash
tuta:x:0:0:root:/var:/bin/bash
```

Bạn có thể thấy được ở trên có 4 tài khoản superuser. Để hiểu tại sao bạn hãy xem định dạng một dòng trong file /etc/passwd

```
username:passwd:UID:GID:fullname:home-dir:shell
```

Bạn hãy chú ý vào các trường UID (User ID) và GID (Group ID) của tài khoản root. Những tài khoản mà có các giá trị của các trường này là 0 là những superuser. Hay nói một cách khác những người có UID = 0 và GID = 0 có quyền tương đương với tài khoản root.

Như vậy nếu hệ thống của bạn phải có nhiều tài khoản superuser do một số lý do quản trị, bạn có thể dễ dàng tạo một tài khỏan superuser. Tuy nhiên, hãy nhớ rằng một tài khoản superuser (UID=0, GID=0) có thể làm mọi thứ.

4.3. Quản lý người dùng với các công cụ dòng lệnh

4.3.1. Tạo một tài khoản người sử dụng mới

Tạo một người sử dụng mới khá dễ dàng, để tạo người sử dụng từ dòng lệnh, bạn có thể sử dụng câu lệnh useradd. Ví dụ để tạo người sử dụng có tên là tutavn, bạn có thể chạy câu lệnh sau:

useradd tutavn

Trong file /etc/passwd sẽ bổ sung thêm dòng mới như sau:

tutavn:x:502:504::/home/tutavn:/bin/bash

Kí hiệu x có nghĩa là tài khoản chưa có mật khẩu. Vì vậy bạn cần tạo mật khẩu cho người sử dụng bằng câu lệnh sau:

paswd tutavn

Bạn sẽ được yêu cầu vào mật khẩu hai lần, và khi mật khẩu được tiếp nhận, nó sẽ được mã hóa và thêm vào dòng của người sử dụng trong file /etc/passwd. Các giá trị UID và GID sẽ được lựa chọn tự động bởi useradd, thông thường nó tăng giá trị UID và GID lên một so với người được thêm vào lần sau cùng trước đó. Bạn có thể tạo người sử dụng có thư mục chủ khác với mặc định (trong thư mục home) bằng thực hiện câu lệnh:

useradd newuser –d /www/newuser

Người sử dụng mới sẽ được tạo và có thư mục chủ là /www/user. Khi bạn tạo một người sử dụng mới, hệ thống cũng đồng thời mặc định tạo ra một nhóm mới có trong file /etc/group có tên giống như tên tài khoản của người sử dụng. Để tạo người sử dụng với tên nhóm mới hay tên nhóm đã tồn tại trong hệ thống, bạn sử dụng lệnh adduser với tùy chọn –g. Ví dụ:

useradd tutavn –g users

Nếu bạn muốn tạo người sử dụng mới là thành viên của một số nhóm, bạn có thể sử dụng tùy chọn –G. ví dụ

useradd tutavn –G users1,users2

4.3.2. Tạo một nhóm mới

Để tạo một nhóm mới bạn sử dụng câu lệnh groupadd. Ví dụ:

groupadd mygroup

Nếu bạn tạo một tên nhóm đã có trong hệ thống bạn sẽ nhận được một thông báo lỗi

4.3.3. Sửa đổi một tài khoản người sử dụng đang tồn tại

- *Thay đổi mật khẩu*

Để thay đổi mật khẩu của tài khoản đang tồn tại bạn sử dụng câu lệnh passwd.Ví dụ:

passwd tutavn

Câu lệnh này tương đối đơn giản vì nó không có các tùy chọn, và nó chỉ cho phép người sử dụng thông thường chỉ có thể thay đổi mật khẩu của chính họ. Hệ thống sẽ yêu cầu bạn nhập mật khẩu hai lần và khi mật khẩu được tiếp nhận, nó sẽ được mã hóa trước khi đưa vào file /etc/passwd

4.3.4. Thay đổi đường dẫn thư mục chủ

Để thay đổi đường dẫn thư mục chủ của người sử dụng đang tồn tại, sử dụng câu lệnh usermod như sau:

usermod –d **new_home_directory username**

Ví dụ, nếu một người sử dụng tutavn có thư mục chủ /home/tutavn và muốn chuyển thành /home2/tutavn, bạn có thể chạy câu lệnh sau:

usermod –d /home2/tutavn tutavn

Tuy nhiên, nếu bạn muốn nội dung thư mục chủ đến một vị trí mới, sử dụng tùy chọn –m như sau:

usermod –d –m /home2/tutavn tutavn

4.3.5. Thay đổi UID

Để thay đổi UID của một người sử dụng, sử dụng câu lệnh usermod như sau:

usermod –u UID username

Ví dụ:

usermod –u 500 myfrog

Câu lệnh này sẽ thay đổi UID của người sử dụng myfro là 500

4.3.6. Thay đổi nhóm mặc định

Để thay đổi nhóm mặc định cho người sử dụng, sử dụng câu lệnh usermod với tùy chọn –g

usermod –g 777 myfrog

Câu lệnh này sẽ thay đổi nhóm mặc định của myfrog thành 777.

4.3.7. Thay đổi thời hạn kết thúc của một tài khoản

Bạn có thể thay đổi thời hạn kết thúc của một tài khoản sử dụng câu lệnh usermod với tùy chọn –e. Cú pháp của câu lệnh như sau:

usermod –e MM/DD/YY username

Ví dụ:

usermod –e 12/31/99 kabir

4.3.8. Sửa đổi một nhóm đang tồn tại

Để sửa đổi tên một nhóm đang tồn tại, sử dụng câu lệnh groupmod. Cú pháp như sau:

groupmod –n **new_group current_group**

Ví dụ:

groupmod –n experts novices

Nhóm novices đang tồn tại được đổi tên thành experts. Để thay đổi GID của một nhóm sử dụng tùy chọn –g như sau:

groupmod –g 666 troublemaker

Câu lệnh này sẽ thay đổi GID của một nhóm troublemaker thành 666.

4.3.9 Xóa hoặc hủy bỏ một tài khoản người sử dụng

Để xóa một tài khoản đang tồn tại sử dụng câu lệnh userdel. Ví dụ:

userdel snake

Sẽ xóa bỏ tài khoản tài khoản snake khỏi hệ thống. Nếu bạn muốn xóa thư mục chủ của người sử dụng và tất cả các nội dung trong thư mục, sư dụng tùy chọn –r. Chú ý rằng userdel sẽ không xóa người sử dụng nếu người sử dụng hiện tại đang đăng nhập.

Nếu bạn muốn hủy bỏ tạm thời quyền truy cập của tất cả các tài khoản bạn có thể tạo một file tạm thời có tên là /etc/nologin với một thông tin giải thích lý do vì sao không được phép truy cập. Chương trình login sẽ không cho phép bất kỳ tài khoản nào khác tài khoản root có thể đăng nhập trong thời gian này.

4.4. Cài đặt máy in

4.4.1. Cấu hình máy in

Ứng dụng **printconf** cho phép người dùng cấu hình máy in trong Red Hat Linux. Nó cho phép sửa đổi tệp tin cấu hình `/etc/printcap`, các thư mục bộ đệm in và bộ lọc in.**printconf** cấu hình hệ thống in ấn của bạn, được gọi là LPRng. LPRng cũng là một hệ thống in ấn ngầm định. Phần này tập trung vào việc sử dụng **printconf** để cấu hình LPRng.

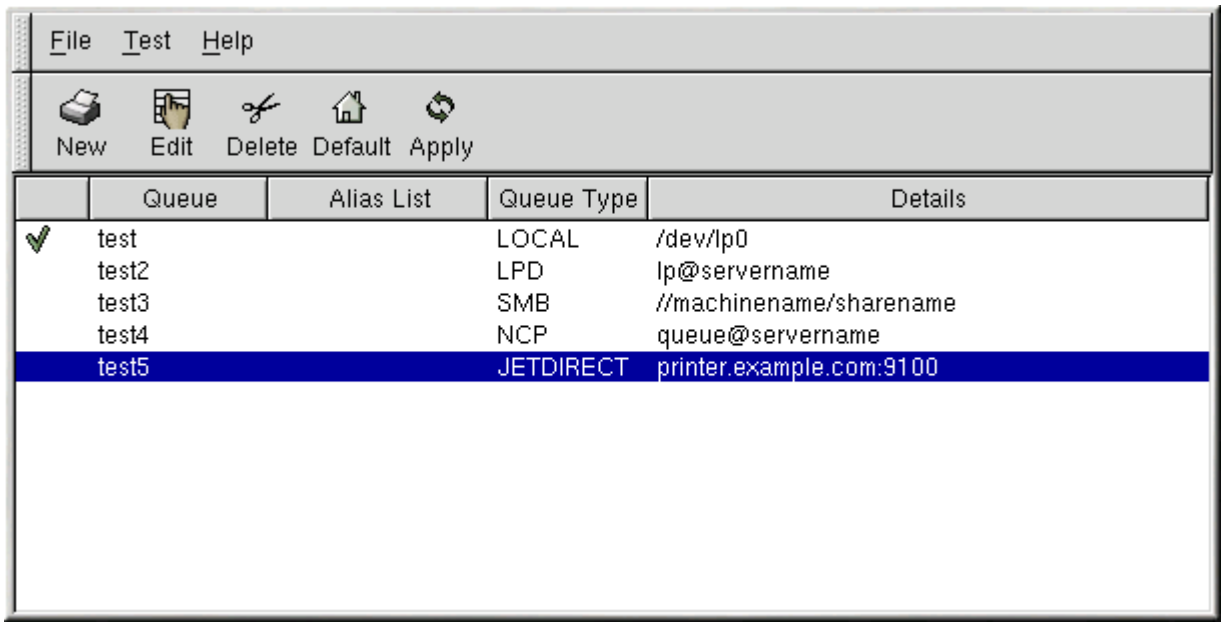
Để sử dụng **printconf**, bạn phải có quyền truy cập mức root. Để khởi động **printconf**, theo một trong các cách sau đây

- Trên màn hình GNOME, chọn **Main Menu Button** => **Programs** => **System** => **Printer Configuration** để khởi động trong chế độ đồ họa.
- Trên màn hình KDE, chọn **Main Menu Button** => **System** => **Printer Configuration** để khởi động chế độ đồ họa.
- Đánh lệnh `printtool` tại dấu nhắc shell (VD: XTerm hoặc GNOME terminal) để khởi động **printconf**

Bạn cũng có thể chạy **printconf** dưới dạng một ứng dụng trong chế độ text nếu bạn không cài đặt hệ thống X Window hoặc bạn thích sử dụng giao diện text hơn. Khi đó, bạn phải log in theo tài khoản root (hoặc dùng lệnh `su` để chuyển sang người dùng root và đánh lệnh `/usr/sbin/printconf-tui` tại dấu nhắc shell.

Chú ý: bạn đừng sửa đổi tệp tin `/etc/printcap`, mỗi khi daemon máy in (lpd) được khởi động hay khởi động lại, tệp tin /etc/printcap mới sẽ được sinh ra tự động.

Nếu bạn muốn cài đặt máy in mà không sử dụng **printconf**, khi đó bạn phải chỉnh sửa tệp tin `etc/printcap.local`. Các đầu vào trong `/etc/printcap.local` không được hiển thị trong **printconf** nhưng được daemon máy in đọc khi khởi động dịch vụ in ấn. Mỗi khi bạn nâng cấp hệ thống của bạn lên phiên bản mới, tệp cấu hình sẽ được **printconf** chuyển sang định dạng mới và tệp tin cấu hình cũ sẽ được ghi dưới tên `/etc/printcap.old`.



*Hình 1: Cửa sổ **printconf** chính*

Có năm kiểu hàng đợi in được cấu hình bởi **printconf**:

- **Local Printer** — máy in được gắn trực tiếp vào máy tính của bạn thông qua cổng song song hoặc cổng USB. Kiểu hàng đợi in **Queue Type** sẽ được thiết lập là **LOCAL.**
- **Unix Printer (lpd Spool)** — máy in được gắn trên một hệ thống UNIX khác mà có thể được truy nhập thông qua mạng TCP/IP. Kiểu hàng đợi in **Queue Type** cho máy UNIX ở xa sẽ được thiết lập là **LPD.**
- **Windows Printer (SMB)** — máy in được gắn trên một hệ thống khác (Windows) có chia sẻ máy in thông qua mạng SMB (sử dụng dịch vụ samba để chia sẻ tài nguyên trên mạng: máy in, dữ liệu.....), kiểu hàng đợi in **Queue Type** lúc đó sẽ được thiết lập là **SMB.**
- **Novell Printer (NCP Queue)** — máy in được gắn vào một hệ thống sử dụng công nghệ mạng Novell's NetWare. Kiểu hàng đợi in cho máy in Novel ở xa sẽ được thiết lập là **NCP.**
- **JetDirect Printer** — máy in được nối trực tiếp vào mạng (máy in mạng). Kiểu hàng đợi in **Queue Type** cho máy in JetDirect sẽ được thiết lập là **JETDIRECT.**

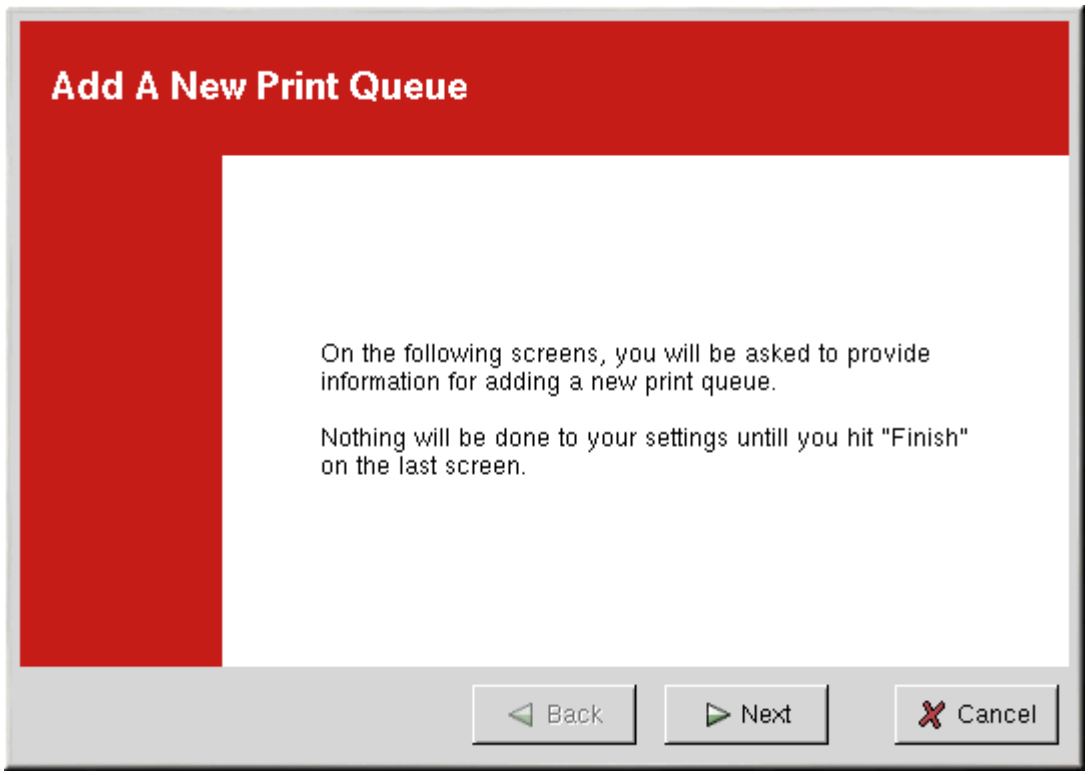
Chú ý: Khi bạn thêm một hàng đợi in mới hay sửa đổi hàng đợi in cũ, bạn phải khởi động lại daemon máy in (lpd) để những thay đổi đó có hiệu lực.

Chọn **Apply** ghi lại những thay đổi mà bạn vừa thực hiện và khởi động lại daemon máy in. Các thay đổi sẽ chưa được ghi trong tệp tin cấu hình `/etc/printcap` cho đến khi daemon máy in (lpd) được khởi động lại. Để thực hiện công việc này, chọn **File** => **Save Changes** và sau đó chọn **File** => **Restart lpd.**

Nếu một máy in xuất hiện trong danh sách in với **Queue Type** được thiết đặt là **INVALID**, cấu hình máy in có thể thiếu các tuỳ chọn cần có cho máy in hoạt động. Chọn **Delete** để xóa máy in khỏi danh sách.

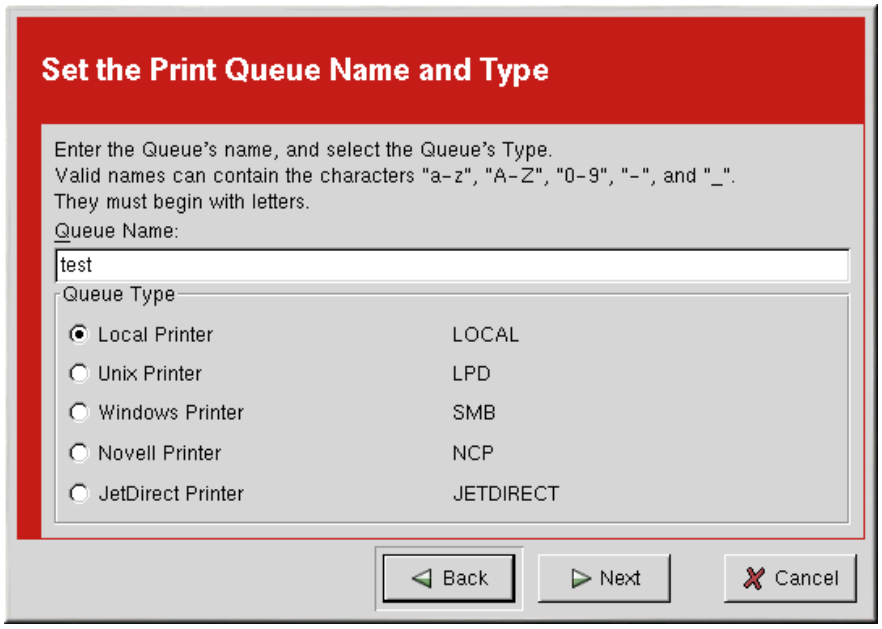
4.4.2. Cài đặt máy in cục bộ

Để cài đặt một máy in gắn trên cổng song song hay cổng USB của máy tính, nhấn nút New trên cửa sổ printconf chính như trên, chọn Next để tiếp tục.



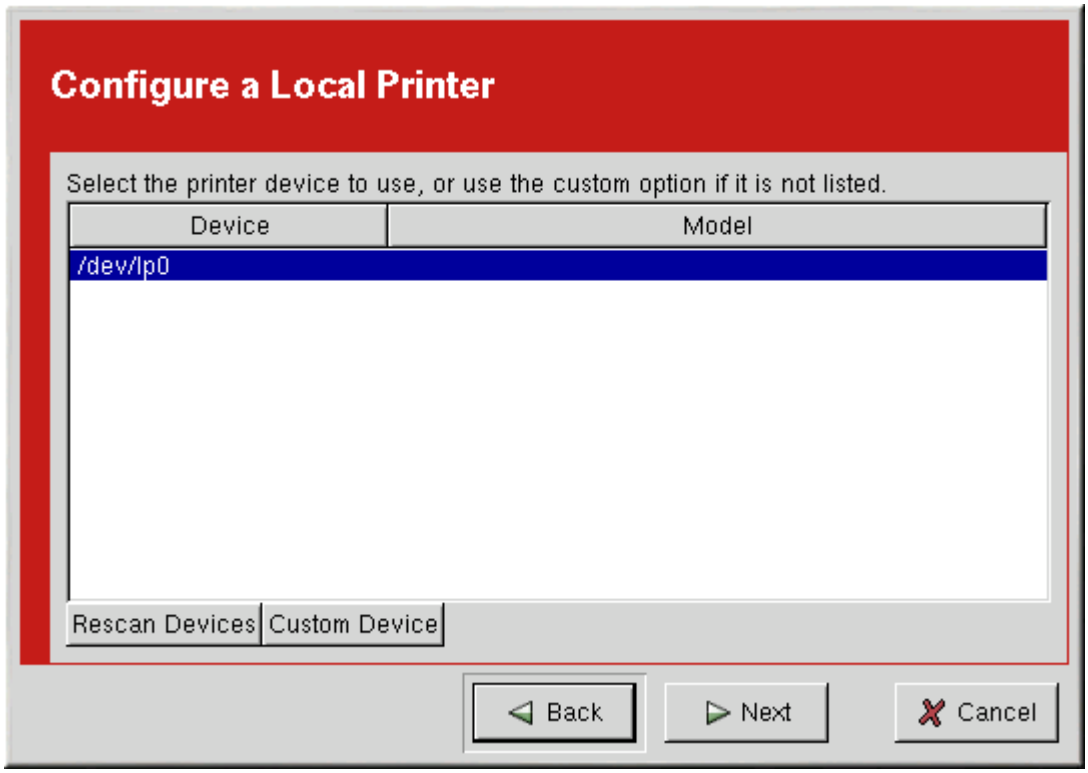
Hình 2: Cài đặt máy in

Nhập tên máy in trong trường **Queue Name** và chọn **Local Printer** từ danh sách **Queue Type** nhấn **Next** để tiếp tục.



Hình 3: Cài đặt máy in cục bộ

printconf sẽ có gắng phát hiện máy in và hiển thị như trong hình 4.

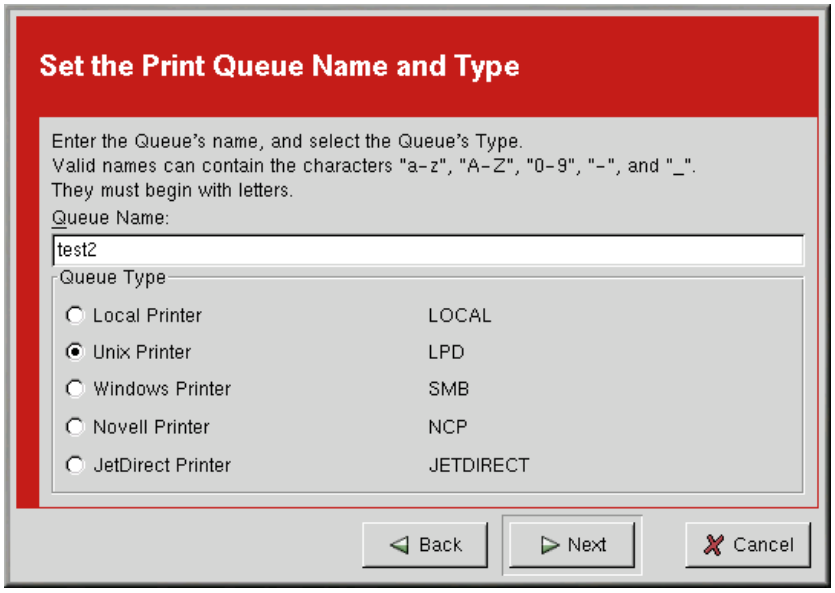


Hình 4: Chọn thiết bị máy in

4.4.3. Cài đặt máy in trên hệ thống Unix ở xa

Để cài đặt một máy in gắn trên một hệ thống Linux ở xa trong cùng một mạng, nhấn nút **New** trong cửa sổ chính **printconf.** Một cửa sổ như hình 2 sẽ xuất hiện, chọn **Next** để tiếp tục.

Cửa sổ như hình 3 xuất hiện. Bạn cũng phải nhập tên máy in vào trường **Queue Name** và chọn **Unix Printer** từ trong thực đơn **Queue Type**, nhấn **Next** để tiếp tục.



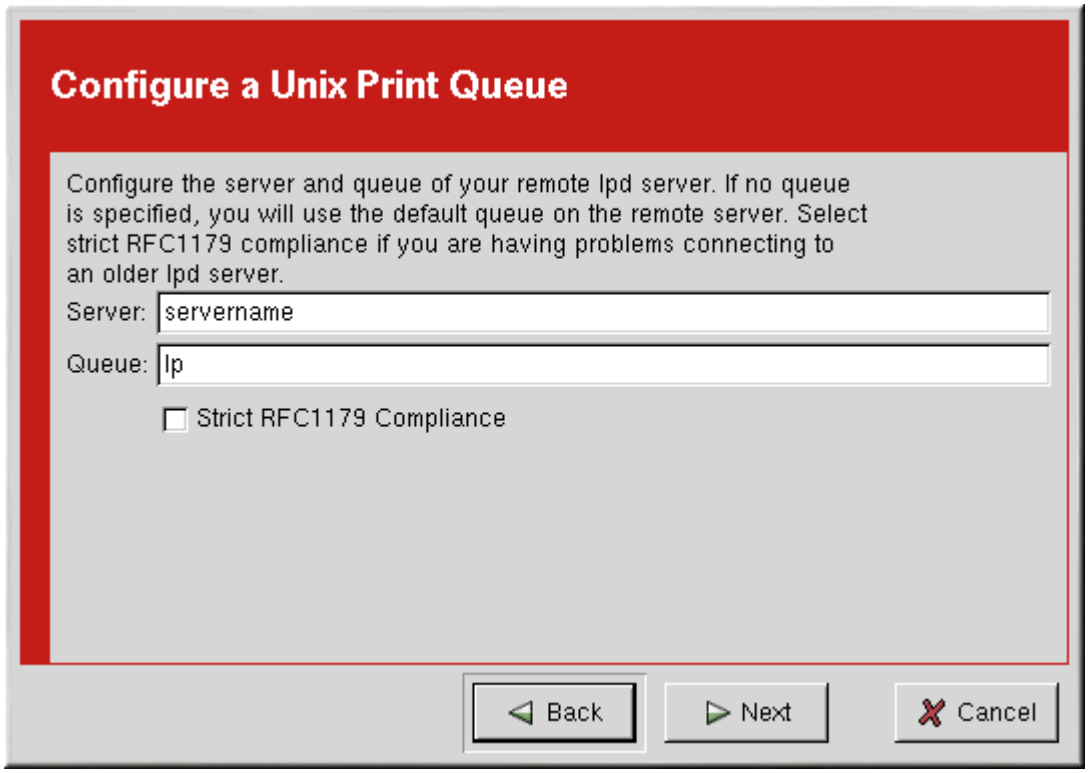
Hình 5: Cài đặt máy in Unix ở xa

Cửa sổ tiếp theo cho phép bạn cấu hình máy chủ in ở xa đó.

- **Server** — Hstname hoặc địa chỉ IP của máy ở xa mà máy in gắn vào.
- **Queue** — Hàng đợi máy in ở xa, ngầm định là `lp`.

Ngầm định không chọn tuỳ chọn **Strict RFC1179 Compliance**. Chỉ khi nào bạn gặp vấn đề về in ấn với một hàng đợi với một hàng đợi lpd không phải Linux, hãy chọn tuỳ chọn này để cấm các tính năng in ấn LPRng nâng cao.

Nhấn **Next** để tiếp tục.

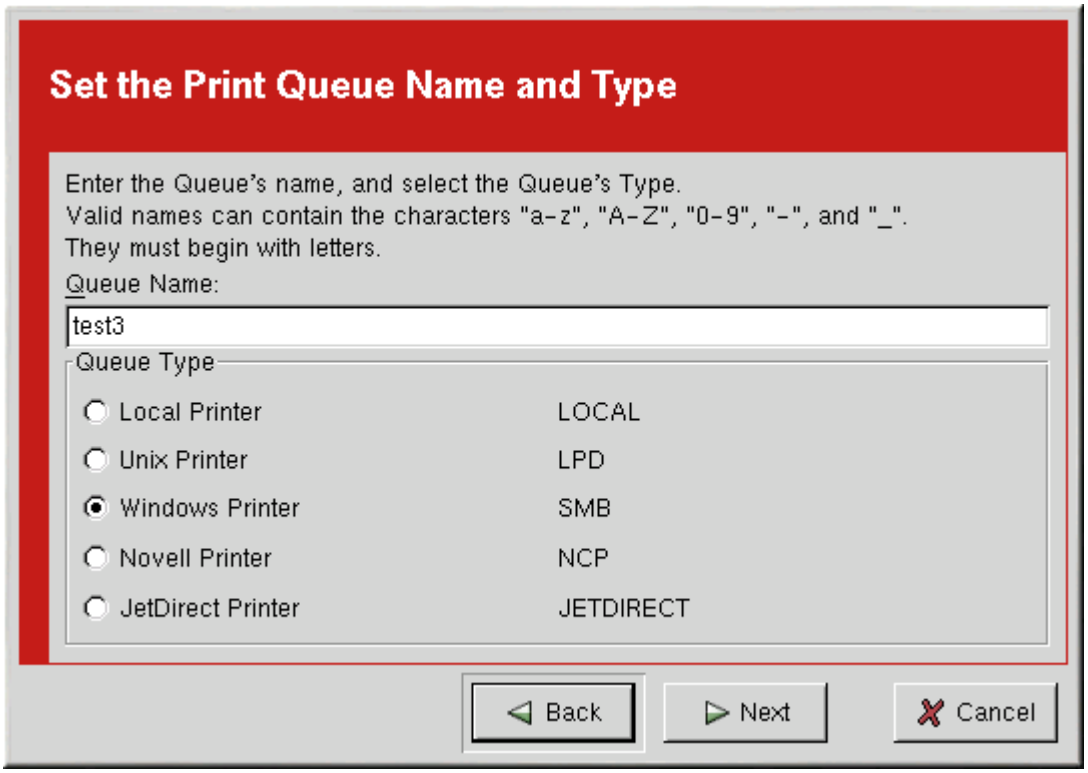


Hình 6: Chọn Printer Server

Bước tiếp theo là chọn kiểu máy in kết nối với hệ thống ở xa đó. Chú ý rằng máy ở xa phải được cấu hình để cho phép một máy cụ bộ có thể đưa yêu cầu và in ấn. Để thực hiện điều đó, bạn phải tạo một file /etc/hosts.lpd trên máy ở xa mà máy in gắn kèm và thêm vào các địa chỉ IP hay hostname của các máy muốn in trên các dòng riêng rẽ trong tệp tin.

4.4.4. Cài đặt máy in Samba (SMB)

Các bước thực hiện ban đầu tương tự hai bước ở trên. Trong thực đơn Queue Type, chọn Windows Printer và nhấn Next để tiếp tục.

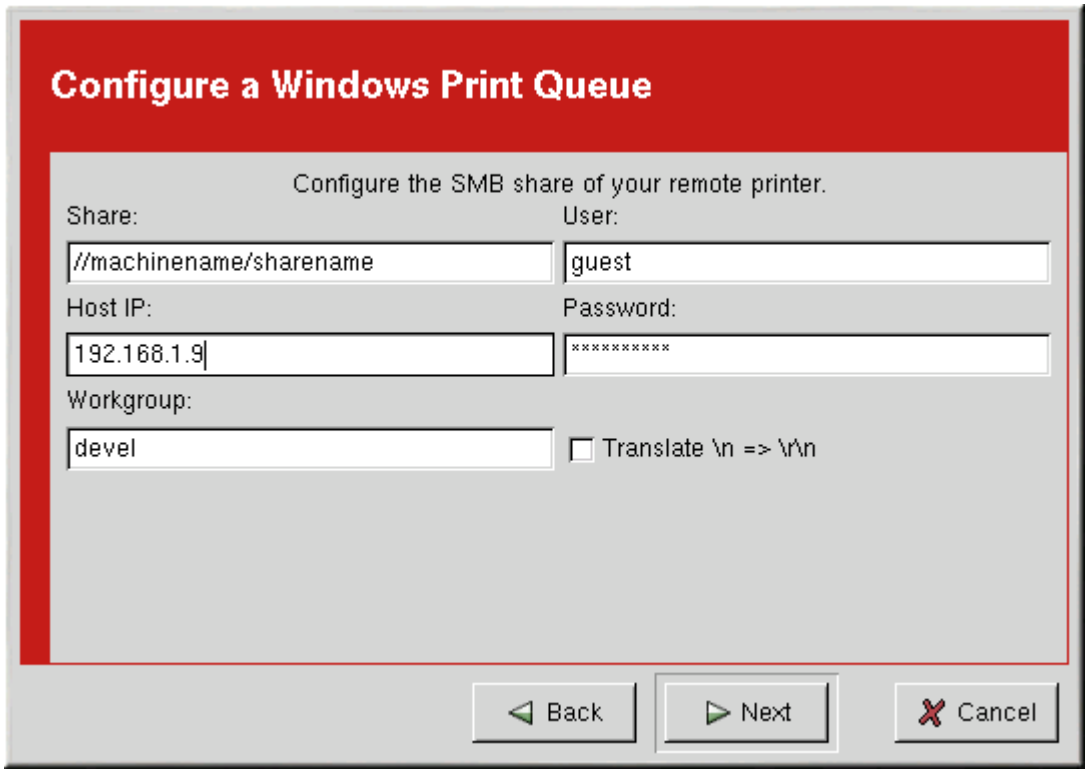


Hình 7: Cài đặt máy in SMB

Trong cửa sổ của hình 8, điền các thông số cấu hình sau:

- **Share** — Tên của máy in được chia sẻ mà bạn muốn in tại đó. Tên này phải cùng tên với tên được định nghĩa cho máy in Samba trên máy Windows ở xa. Chú ý cú pháp phải như sau: *//machinename/sharename*.
- **User** — Tên người dùng được phép truy nhập vào máy in. Tên này phải tồn tại trên hệ thống Windows và người dùng có quyền truy nhập máy in. Tên thường là **guest** đối với các máy Windows servers, hoặc **nobody** đối với các máy Samba servers.
- **Host IP** — Hostname hay địa chỉ IP của hệ thống ở xa chia sẻ máy in SMB.
- **Password** — Mật khẩu (nếu có) của người dùng định nghĩa trong trường **User**
- **Workgroup** — Tên workgroup máy chạy Samba thuộc vào.

Chọn nút **Translate \n => \r\n** để chuyển đổi các ký tự cuối dòng sang khuôn dạng mà hệ thống Microsoft Windows có thể đọc được.
Nhấn **Next** để tiếp tục.



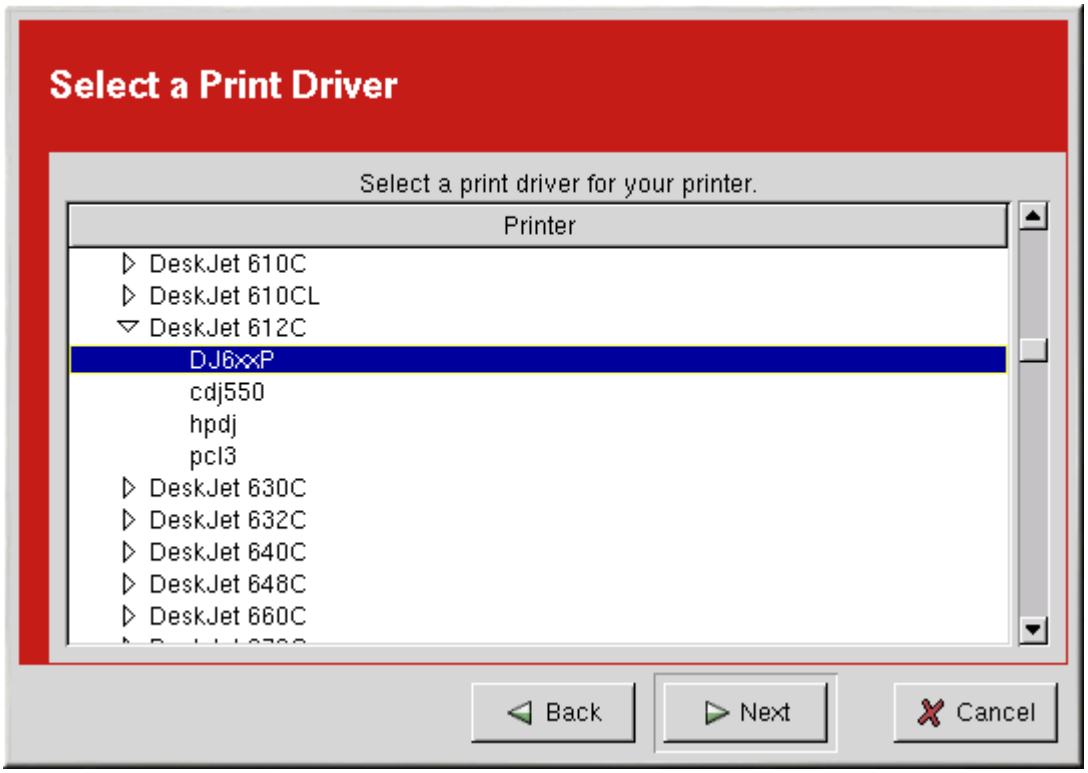
Hình 8: Chọn Print Server

Bước tiếp theo là chọn kiểu máy in được kết nối với hệ thống SMB ở xa.

4.4.5. Chọn trình điều khiển Print Driver và kết thúc

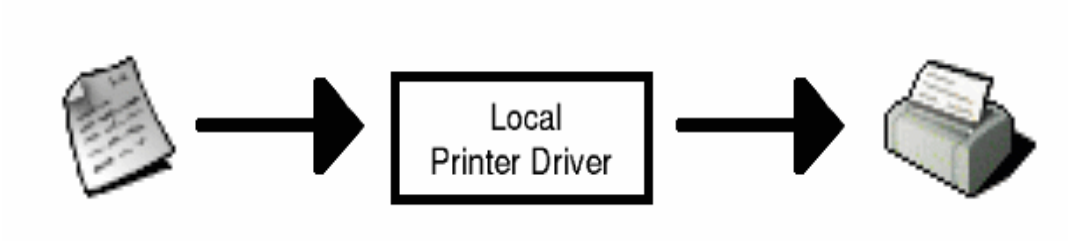
Sau khi đã chọn kiểu hàng đợi máy in và cài đặt các thông số liên quan, bước tiếp theo kà chọn trình điều khiển máy in.

Bạn sẽ thấy một cửa sổ như hình 13. Nếu bạn cấu hình một máy in cục bộ, hãy chọn trình điều khiển in từ trong danh sách, chọn nhà sản xuất và loại máy in của bạn.

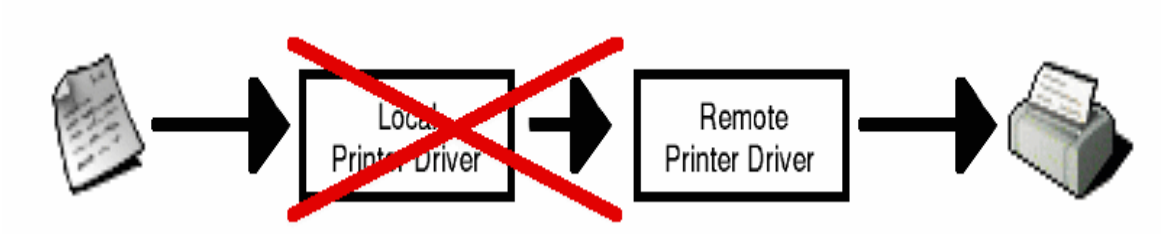


Hình 13: Chọn trình điều khiển máy in

Máy in cục bộ:



Nếu bạn cấu hình máy in ở xa (LPD, SMB, hay NCP), máy in chủ ở xa sẽ in ấn theo trình điều khiển máy in của nó. Cố gắng chọn đúng trình điều khiển máy in ở xa đó.



Bước cuối cùng là khẳng định lại các thông số cấu hình, nhấn nút **Apply** để ghi lại các thay đổi và trong tệp tin cấu hình `etc/printcap` và khởi động lại daemon máy in (`lpd`). Hãy in thử 1 trang xem cấu hình bạn thiết lập đã đúng chưa.

4.4.6. Thay đổi thông số cấu hình các máy in có sẵn

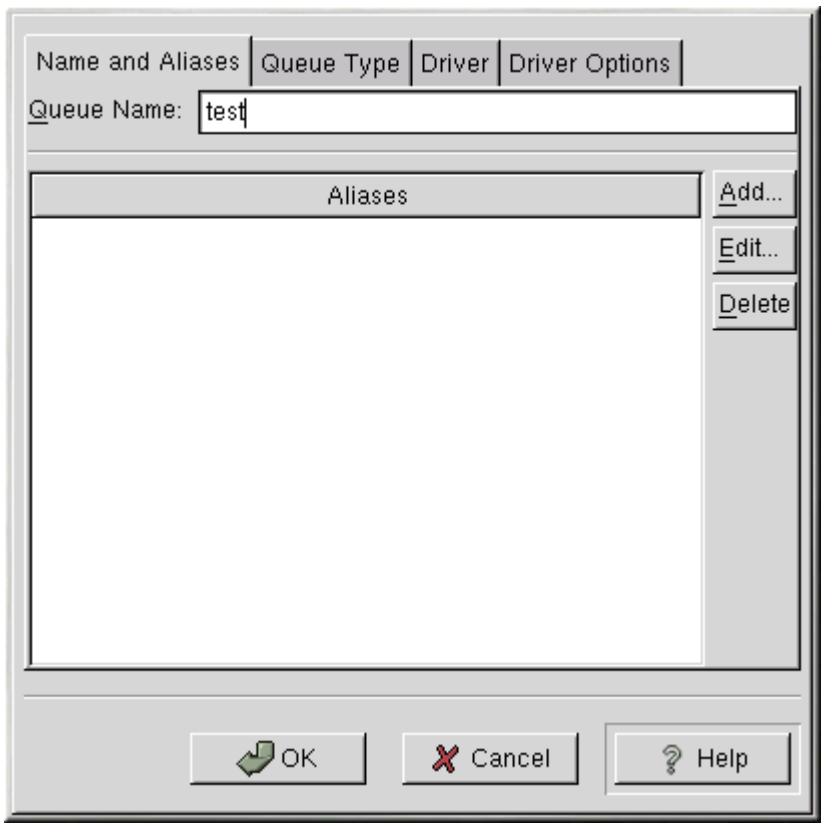
Để xoá một máy in đang tồn tại, chon máy in và nhấn nút **Delete** trên thanh công cụ, máy in sẽ được loại bỏ trong danh sách máy in. Nhấn nút **Apply** để ghi lại các thay đổi và khởi động lại daemo

Để thiết lập một máy in ngầm định, chọn máy in từ danh sách và nhấn nút **Default** trên thanh công cụ. Máy in ngầm định sẽ có icon [redacted] xuất hiện bên cạnh tên máy in.

Nếu bạn muốn thay đổi cấu hình của một máy in, bạn không thể thay đổi các thiết đặt đó một cách trực tiếp mà chỉ được ghi đè lên như sau:

Chọn máy in, chọn **File** => **Override Queue** từ thực đơn. Khi đó, máy in sẽ có kí hiệu [redacted] ở cạnh tên máy in.

Chọn nút Edit để thực hiện việc hiệu chỉnh các thông số. Cửa sổ như hình 14 xuất hiện cho phép bạn thay đổi lại các thông số của máy in.



Hình 14: Thay đổi thông số máy in

4.4.7. Backup các thông số cấu hình máy in

Thông số cấu hình của bạn được đưa vào tệp tin `/etc/printcap` và được daemon máy in đọc khi khởi động. Bạn có thể sử dụng các lệnh để backup lại các file cấu hình ví dụ như backup file cấu hình máy in và ghi thành file settings.xml

```
/usr/sbin/printconf-tui --Xexport > settings.xml
```

Để khôi phục lại file cấu hình đã được backup theo cách trên, bạn có thể sử dụng lệnh dưới đây

```
/usr/sbin/printconf-tui --Ximport < settings.xml
```

4.4.8. Quản lý công việc in ấn

Khi bạn muốn in một file văn bản từ **Emacs** hoặc in một hình ảnh từ **The GIMP**, công việc này sẽ được đưa vào hàng đệm in. Nếu muốn xem danh sách các công việc in ấn, đưa lệnh lpq vào dấu nhắc shell, ví dụ:

```
Rank   Owner/ID            Class  Job Files       Size Time
active user@localhost+902      A      902 sample.txt   2050
01:20:46
```

Nếu muốn dừng một công việc in nào đó, đưa lệnh `lprm` *`job number`* với tham số là định danh của công việc in mà bạn biết được thông qua lệnh **lpq** ở trên. Bạn cũng có thể in ấn thông qua lệnh `lpr sample.txt` để in file văn bản `sample.txt`.

5. Trình diễn thiết lập mạng và cài đặt diul-up trên Linux

5.1. Thiết lập mạng Linux

Chúng ta sẽ xem xét quá trình nối một máy Linux vào mạng Ethernet để trao đổi thông tin bằng giao thức TCP/IP trên Ethernet.

5.1.1. HĐH Linux và card mạng

Để nối một máy Linux vào một mạng Ethernet, bạn cần phải có đầu tiên là một card mạng mà Linux đã có chương trình driver. Sau đây là một số mạng mà Linux có trợ giúp (danh sách sau không đầy đủ và các phiên bản mới của Linux hỗ trợ rất nhiều các card mạng khác nhau) :

3Com 3C509

3Com 3C503/16

Novell NE1000

Novell NE2000

Western Digital WD8003

Western Digital WD8013

Hewlett-Packard HP27245

Hewlett-Packard HP27247

Hewlett-Packard HP27250

Giả sử các bạn muốn gắn máy của mình vào một mạng LAN Ethernet và bạn đã có một card mạng. Vấn đề đầu tiên là sự nhận biết của Linux đối với card này. Nếu card của bạn là một card khá phổ biến như 3c509 của 3COM hay NE2000 của Novell, HDH Linux sẽ nhận biết sự hiện diện của card trong quá trình boot. Để biết xem kết

quả nhận biết card mạng, ta có thể xem xét các thông báo của kernel Linux trong quá trình boot của hệ thống qua lệnh **dmesg**

```
Freeing unused kernel memory: 60k freed
Adding Swap: 72572k swap-space (priority -1)
eth0: 3c509 at 0x300 tag 1, BNC port, address 00 a0 24 4f 3d dc, IRQ 10.
3c509.c:1.16 (2.2) 2/3/98 becker@cesdis.gsfc.nasa.gov.
eth0: Setting Rx mode to 1 addresses.
```

Hai dòng in đậm báo rằng card mạng 3c509 đã được kernel nhận biết. Trong trường hợp kernel không nhận biết card ☹, chúng ta phải làm lại kernel Linux và đặt module điều khiển (driver) của card vào trong kernel hay cấu hình ở chế độ load module.

Để cấu hình tiếp nối mạng qua TCP/IP chúng ta phải xác định rõ các thông tin liên quan đến địa chỉ IP của máy. Các thông tin cần biết là :

Địa chỉ IP của máy

Netmask

Địa chỉ của mạng

Broadcast

Địa chỉ IP của gateway

Chúng ta sẽ lần lượt điểm qua các khái niệm cơ bản trên và sẽ học sâu hơn trong phần TCP/IP của khóa học.

Địa chỉ IP của máy là một dãy 4 số viết dước dạng A.B.C.D, trong đó mỗi số nhận giá tri từ 0-255. Nếu máy của bạn kết nối một mạng nhỏ tại nhà do bạn thiết lập thì địa chỉ kiểu 192.168.1.D là một địa chỉ nên đặt, với D là các số khác nhau cho từng máy. Nếu máy của bạn sẽ hòa nhập với một mạng LAN đã có trước đó và bạn muốn kết nối với các máy khác thì hỏi người quản trị mạng về địa chỉ IP bạn có thể gán cho máy của mình cùng với tất cả các thông số tiếp theo.

Netmask. Tương tự như trên, nếu bạn tự quản, netmask sẽ là 255.255.255.0

Địa chỉ mạng. Nếu bạn tự quản, địa chỉ của mạng sẽ là 192.168.1.0

Broadcast. Nếu bạn tự quản, broadcast là 192.168.1.255

Địa chỉ gateway. Đây là địa chỉ của máy cho phép bạn kết nối với mạng LAN khác, tức là các máy tính với 3 số đầu của địa chỉ không giống bạn là 192.168.1. Bạn bỏ trống nếu bạn chỉ liên lạc với các máy cùng mạng 192.168.1.XXX. Chú ý là địa chỉ mạng của máy gateway bắt buộc phải trùng với địa chỉ mạng của bạn.

Sau khi đã xác định các thông số, ví dụ như

IP address = 192.168.1.15

Netmask = 255.255.255.0

suy ra network address = 192.168.1.0 và broadcast = 192.168.1.255

Gateway = 192.168.1.1

5.1.2. Cấu hình card mạng

Lệnh ifconfig

Sau khi làm cho kernel nhận biết sự hiện diện của card mạng, công tác tiếp theo là cấu hình TCP/IP cho card. Trong quá trình cài đặt Linux Redhat 6.X, bình thường chúng ta đã được chương trình cài đặt hỏi và cấu hình hộ . Trong trường hợp khi chúng ta bổ sung card mạng sau khi Linux đã được cài đặt, chúng ta có thể sử dụng tiện ích **netconf** cho mục đích này hoặc chúng ta sử dụng lệnh **ifconfig** để tự cài đặt.

Lệnh **ifconfig** được sử dụng trong quá trình boot hệ thống để cấu hình các trang thiết bị mạng. Sau đó, trong quá trình vận hành, **ifconfig** được sử dụng cho debug, hoặc để cho người quản trị hệ thống thay đổi cấu hình khi cần thiết .

Lệnh **ifconfig** không có tùy chọn dùng để hiển thị cấu hình hiện tại của máy.

```
[root@pasteur tnminh]# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:A0:24:4F:3D:DC
          inet      addr:192.168.2.20            Bcast:192.168.2.255
Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:531 errors:4 dropped:0 overruns:0 frame:4
          TX packets:1854 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0x300

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:1179 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1179 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Để gán địa chỉ IP 193.105.106.10 cho card mạng Ethernet đầu tiên ta dùng lệnh

```
ifconfig eth0 193.105.106.10 netmask 255.255.255.0 broadcast
192.105.106.255
```

Linux cho phép bạn sử dụng bí danh (alias) cho card mạng, tức là cho phép bạn có nhiều địa chỉ IP cho cùng một card vật lý. Kết quả nhận được gần giống như bạn có gắn nhiều card vật lý lên máy. Do đó, bạn có thể dùng một card để nối với nhiều mạng logic khác nhau. Cú pháp của lệnh này là :

```
ifconfig eth0:0 208.148.45.58 netmask 255.255.255.248 broadcast
208.148.45.255 up
```

Các tập tin cấu hình của kết nối mạng là **/etc/sysconfig/network-scripts/ifcfg-ethX** với X là 0,1 ... hay 0:0, 0:1 .... Bạn có thể thay đổi cấu hình kết nối mạng bằng cách sửa đổi lại tập tin này bằng một chương trình soạn thảo text như **mc** chẳng hạn, sau đó khởi động lại kết nối mạng bằng

```
/etc/rc.d/init.d/network restart
```

Nhớ kiểm tra lại kết quả qua lệnh **ifconfig**.

Lệnh route

Lệnh **route** cho phép làm các thao tác đến bảng dẫn đường (forwarding table) của kernel. Nó được sử đầu tiên để xác định đường dẫn cố định (static) đến những máy hoặc những mạng qua các card mạng ethernet đã được cấu hình trước đó bởi ifconfig. Lệnh **route** không có tùy chọn (option) cho phép hiển thị bảng dẫn đường hiện tại của kernel (Lệnh **netstat –r** cũng có tác dụng tương tự)

```
[root@pasteur tnminh]# /sbin/route

Kernel IP routing table
Destination     Gateway   Genmask   Flags Metric Ref Use Iface
192.168.2.20    *   255.255.255.255 UH   0      0     0   eth0
192.168.2.0     *   255.255.255.0   U    0      0     0   eth0
127.0.0.0       *   255.0.0.0       U    0      0     0   lo
default       192.168.2.10   0.0.0.0   UG    0    0   0 eth0
```

Để chỉ ra rằng card mạng eth0 được nối với một mạng 208.148.45.56 ta dùng lệnh route như sau :

```
route add -net 208.148.45.56 eth0
```

Còn nếu chúng ta muốn sử dụng bí danh của card mạng để nối vào một mạng logic khác, ta có thể sử dụng lệnh

```
route add -net 193.105.106.0 eth0:0
```

Công tác cuối cùng là phải chỉ ra các địa chỉ của gateway mặc định.

```
route add default gw 193.105.106.1 metric 1
```

Biết sử dụng thành thạo cú pháp của 2 lệnh **ifconfig** và **route** rất quan trọng, nó cho phép các cán bộ quản trị thay đổi cấu hình kết nối mạng của một server một cách nhanh chóng và không phải khởi động lại máy. Vì vậy, server luôn sẵn sàng. Bạn cũng có thể sử dụng tiện ích **netconfig** để cấu hình liên kết mạng nếu chưa thành thạo nhiều cú pháp của các lệnh trên.

Lệnh ping

Ứng dụng của lệnh này là để thử xem 2 máy có kết nối được với nhau chưa. Cú pháp cơ bản của lệnh rất đơn giản là *ping địa_chỉ_IP_máy_đích.* Ví dụ như

```
[tnminh@proxy tnminh]$ ping sun

    PING sun.vnuhcm.edu.vn (172.16.1.4): 56 data bytes
    64 bytes from 172.16.1.4: icmp_seq=0 ttl=255 time=0.1 ms
    64 bytes from 172.16.1.4: icmp_seq=1 ttl=255 time=0.2 ms
    64 bytes from 172.16.1.4: icmp_seq=2 ttl=255 time=0.1 ms
    64 bytes from 172.16.1.4: icmp_seq=3 ttl=255 time=0.1 ms

    --- sun.vnuhcm.edu.vn ping statistics ---
    4 packets transmitted, 4 packets received, 0% packet loss
    round-trip min/avg/max = 0.1/0.1/0.2 ms
```

Nếu 2 máy có thể liên lạc được với nhau, chúng ta sẽ biết thêm thời gian trả lời để cho biết sự thông thoáng về mạng giữa 2 máy. Có thể nói, **ping** phải chạy trước tiên trước tất cả các hoạt động mạng khác.

Chú ý: Nên sử dụng ping –n để tránh trục trặc do dịch vụ DNS làm ảnh hưởng tới việc kết quả thử kết nối mạng.

Lệnh Traceroute

Đây cũng là lệnh cho phép chẩn đoán hoạt động của mạng. Cú pháp của lệnh giống như lệnh **ping** nhưng kết quả không chỉ dừng ở sự trả lời mà còn chỉ ra các thiết bị trung gian nằm giữa 2 máy.

```
# tnminh@nefertiti ~ > traceroute 203.162.44.33

traceroute to 203.162.44.33 (203.162.44.33): 1-30 hops, 38 byte packets
 1  makeda.pasteur.fr (157.99.64.3), 1.66 ms, 1.66 ms, 1.66 ms
 2   418.ATM4-0.GW21.Defense.OLEANE.NET (195.25.28.149), 5.0 ms, 4.17 ms,
4.17 m
 3   FastEth0-0.GW16.Defense.OLEANE.NET (195.25.25.208), 4.17 ms, 4.17 ms,
4.17s
```

```
 4   100.ATM6-1.GW2.Telehouse.OLEANE.NET (194.2.3.245), 5.0 ms, 5.0 ms, 5.0
ms

 ..............

14   210.132.93.210 (210.132.93.210), 849 ms (ttl=241!), 807 ms (ttl=241!),
970

s (ttl=241!)

15      202.167.121.195   (202.167.121.195),   905   ms   !H   203.162.3.42
(203.162.3.42), 1

88 ms (ttl=242!)
```

Lệnh **traceroute** là một công cụ hiệu quả cho phép ta phát hiện lỗi trong quá trình phân đường (IP routing). Ví dụ kết nối từ A -> C có trục trặc và với traceroute tới C từ máy A, ta có thể phát hiện ra máy A kết nối máy B, rồi máy B lại kết nối máy A ... do cấu hình routing của A và B sai.

Chú ý là khi chúng ta thử kết nối với một máy ở xa trong Internet, do nhiều mạng áp dụng các bức tường lửa (firewall) nên nhiều khi lệnh ping và traceroute không chạy nhưng trên thực chất là mạng vẫn thông.

5.1.3. Các tiện ích mạng: Telnet và ftp

Telnet

Telnet là mọt tiện ích cho phép đăng nhập vào một máy tính ở xa và làm việc giống như với máy tại chỗ. Ví dụ, có thể dùng telnet để chạy một chương trình trong một siêu máy tính ở cách xa hàng ngàn dặm. Telnet sử dụng giao thức TCP/IP, cổng 23.

Sử dụng: giả sử máy của bạn đang chạy Window và bạn đã được cấp một tài khoản trong máy chủ Linux.

1. Nhấn chuột vào "Start" chọn "RUN".
2. Gõ vào: "**telnet** <tên hay địa chỉ IP>" của máy chủ mà bạn có tài khoản. Ví dụ "telnet linuxcourse.iti.edu.vn" và nhấn OK.
3. Nếu kết nối đến máy chủ thông suốt, một cửa sổ sẽ hiện lên mời bạn cung cấp tên tài khoản và mật khẩu.
4. Nhập vào tên tài khoản *username* và *password* để đăng nhập.
5. Đăng nhập thành công thì bạn sẽ đứng tại thư mục nhà (home directory) của mình.
6. Bắt đầu phiên làm việc của bạn. Ví dụ, dùng câu lệnh "ls -al" để hiển thị tất cả các tệp trong thư mục.
7. Kết thúc phiên làm việc, gõ "**exit**".

FTP

FTP là viết tắt của Tệp Transfer Protocol, một tiện ích tải tệp ở xa. Với ftp có thể lấy tệp ở máy từ xa về máy tính của mình (download) và ngược lại, gửi một tệp từ máy của mình lên máy ở xa (upload) nếu bạn có quyền write vào thư mục ở máy đó. FTP sử dụng giao thức TCP/IP, cổng 21.

Sử dụng FTP

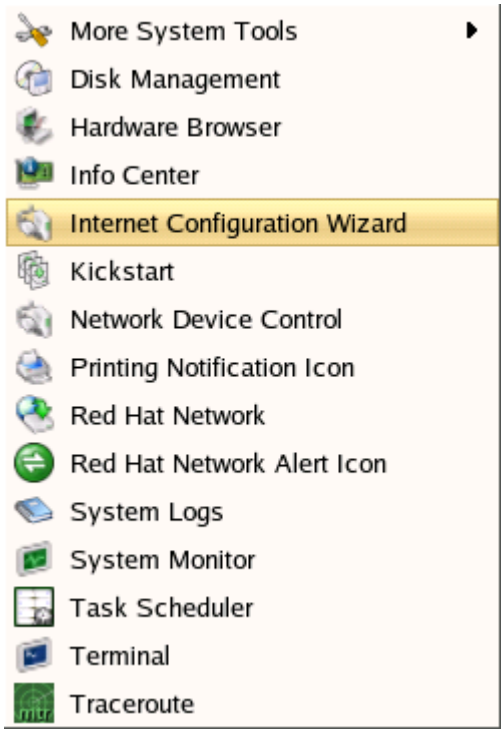
Cách tải xuống (download):

- Telnet vào máy ở xa.
- Gõ lệnh **ftp** <tên máy ở xa>.
- Máy sẽ yêu cầu tên đăng nhập và password. Một trong những chế độ cho phép mọi người tải tệp về tự do là dùng tên đăng nhập "anonymous" và password là địa chỉ email của bạn.
- Chuyển đến thư mục có các tệp ta muốn tải về.
- Gõ lệnh: **get** <tên tệp muốn tải về>.
- Để kết thúc gõ **quit.**

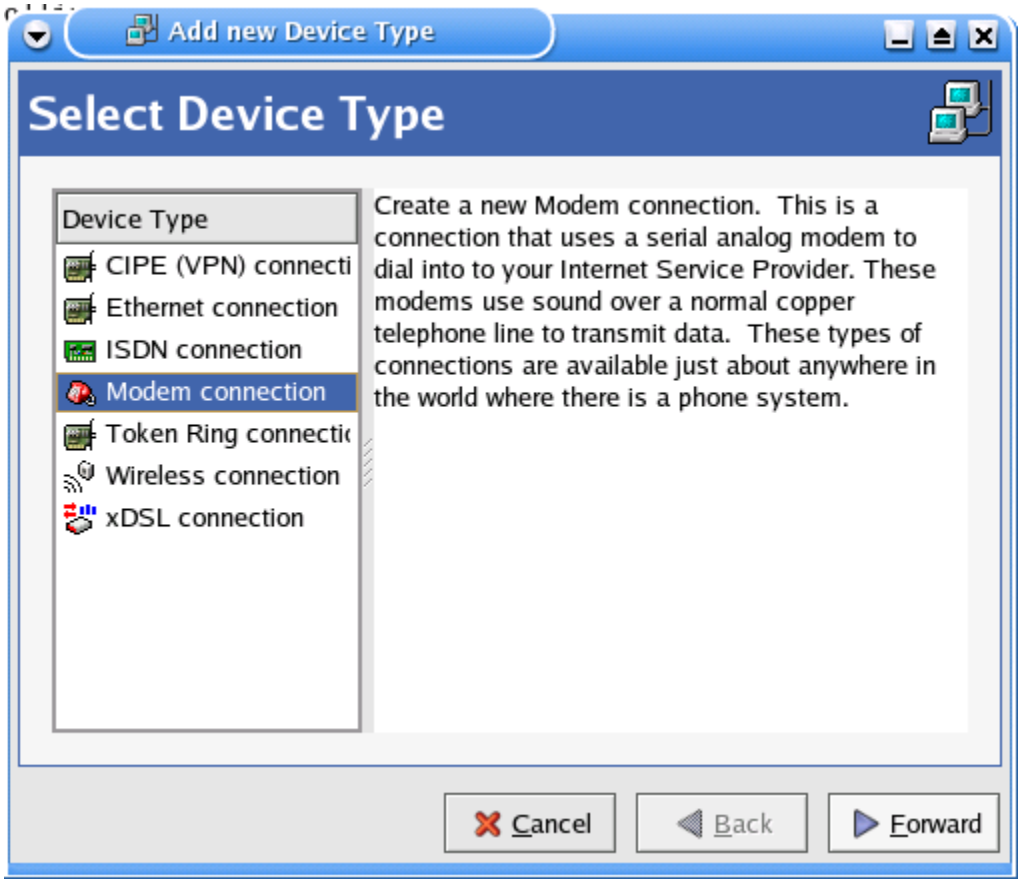
Cách tải lên (upload): Tương tự như trên, nhưng dùng câu lệnh **put** thay cho câu lệnh **get**.

5.2. Cài đặt diul-up trên Linux
5.2.1. Cài đặt

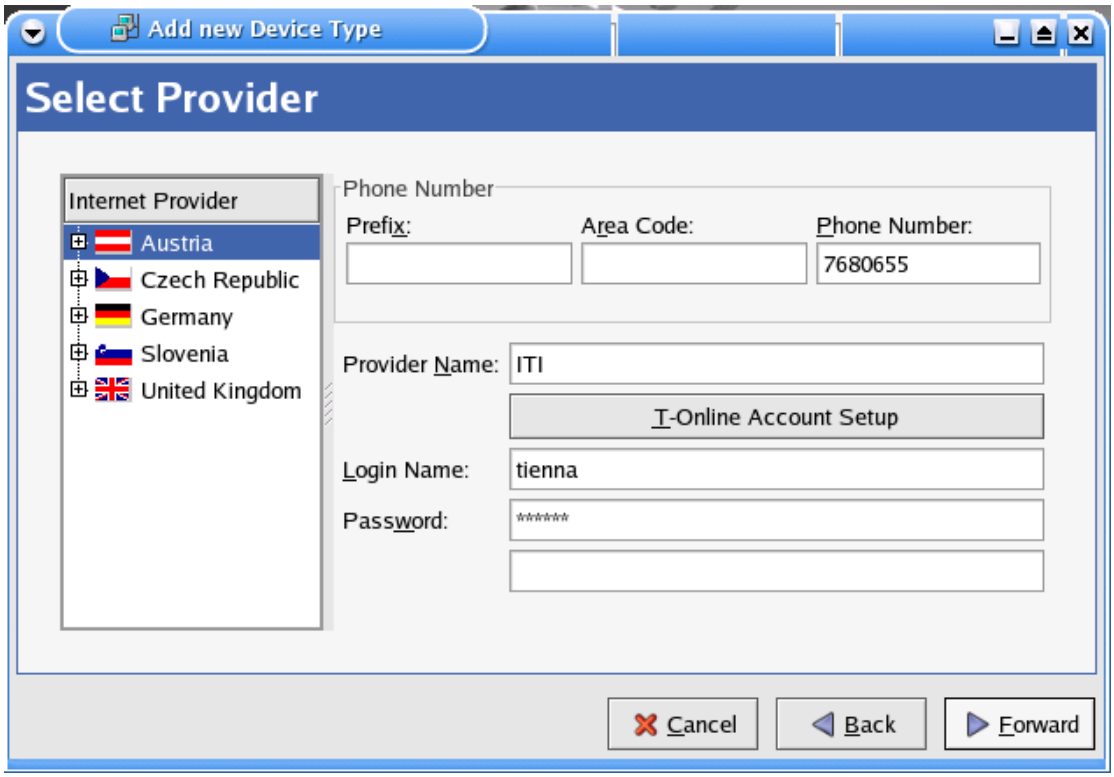
Chọn Internet Configuration Wizard tug menu System configuration



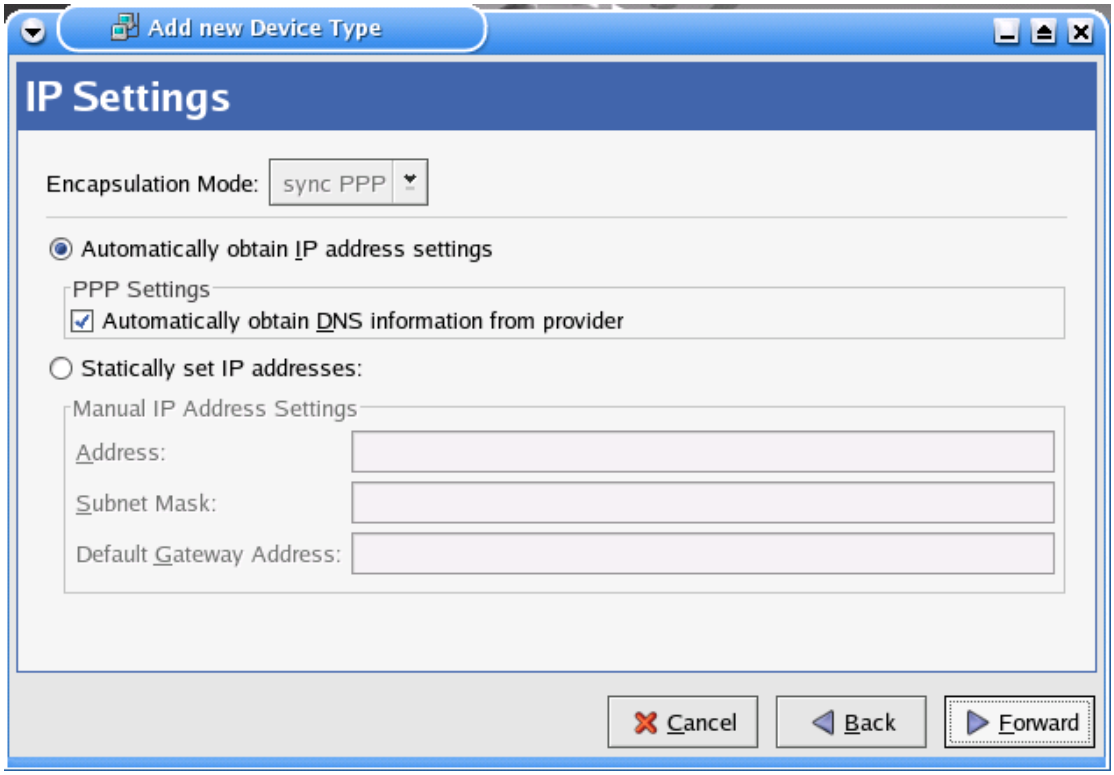
Sau đó màn hình này sẽ chỉ thị



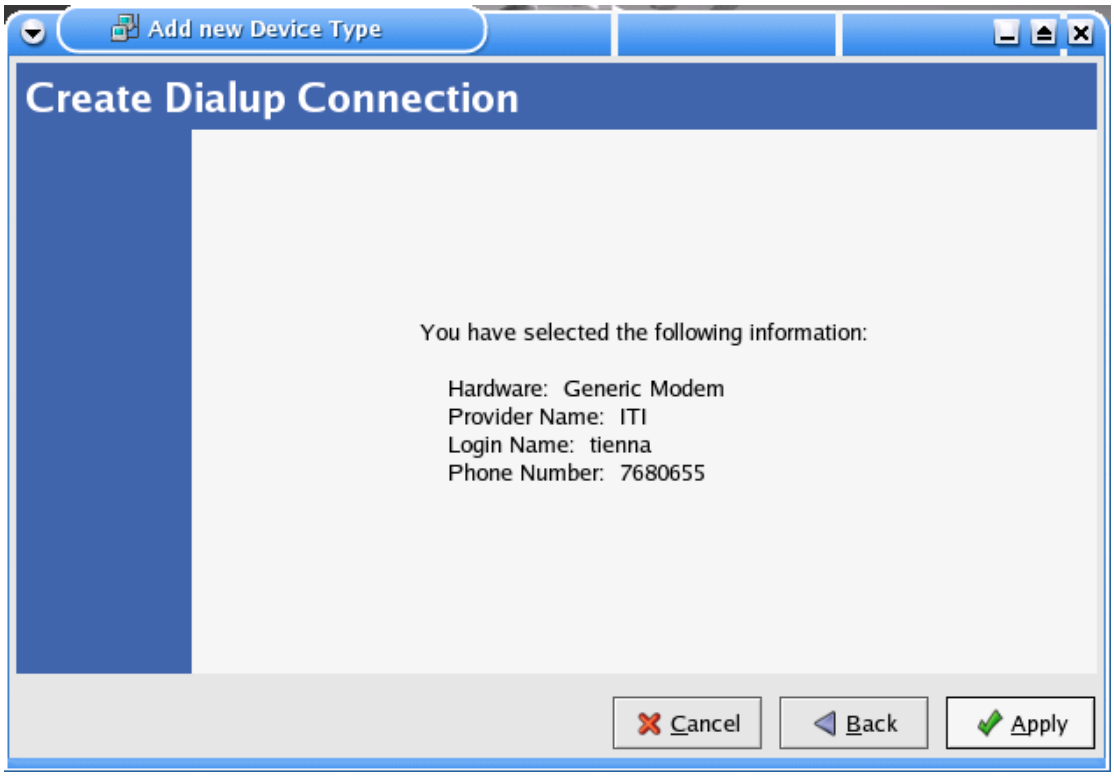
chọn Modem connection, chọn Forward.



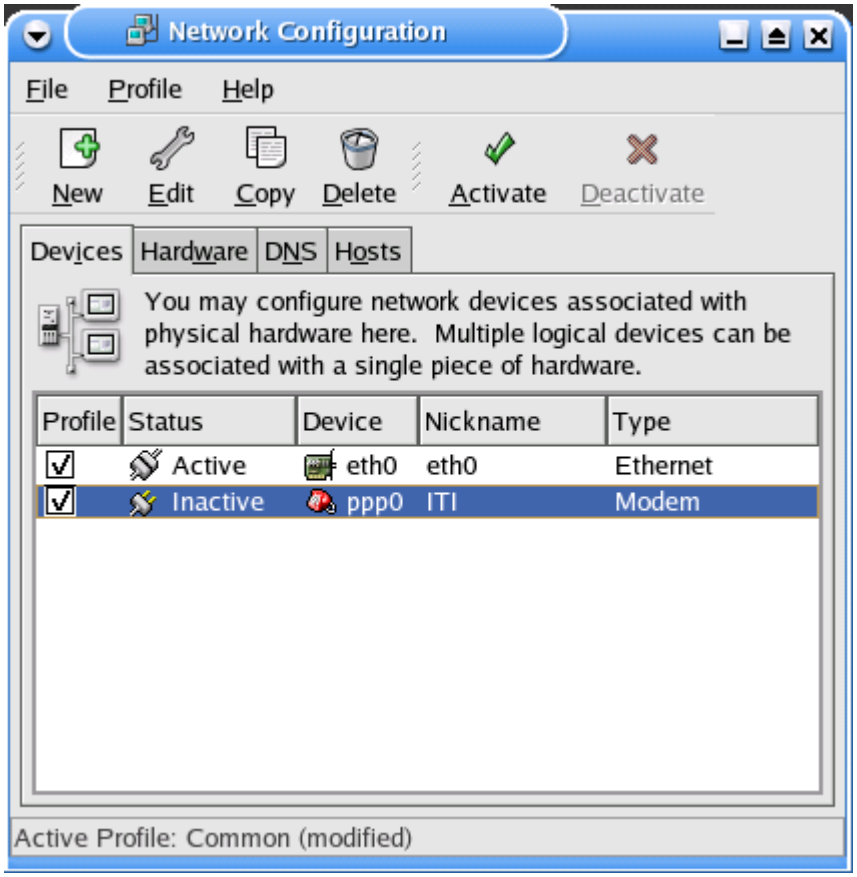
Nhập vào các thông tin quay số., sau đó chọn Forward



Chọn gán IP động, chọn Forward



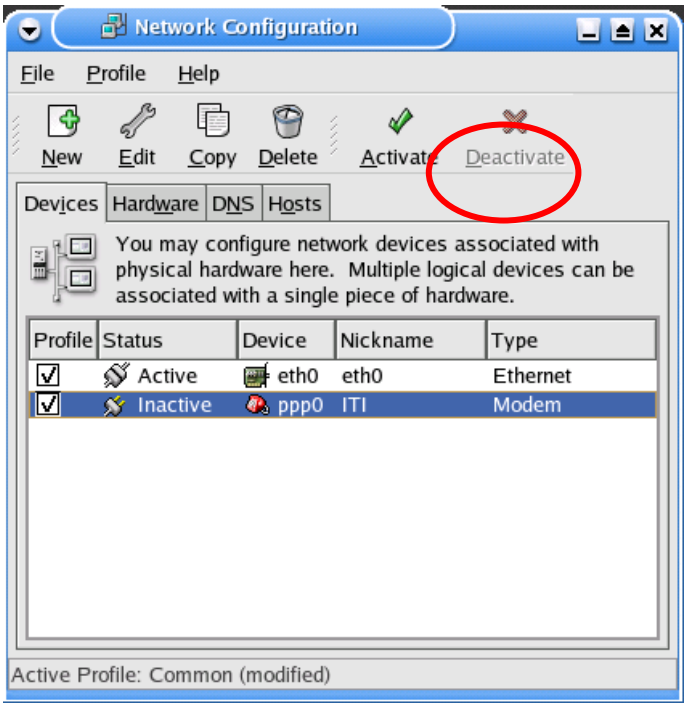
Chọn Apply, sau đó cửa sổ Network configuration hiện ra



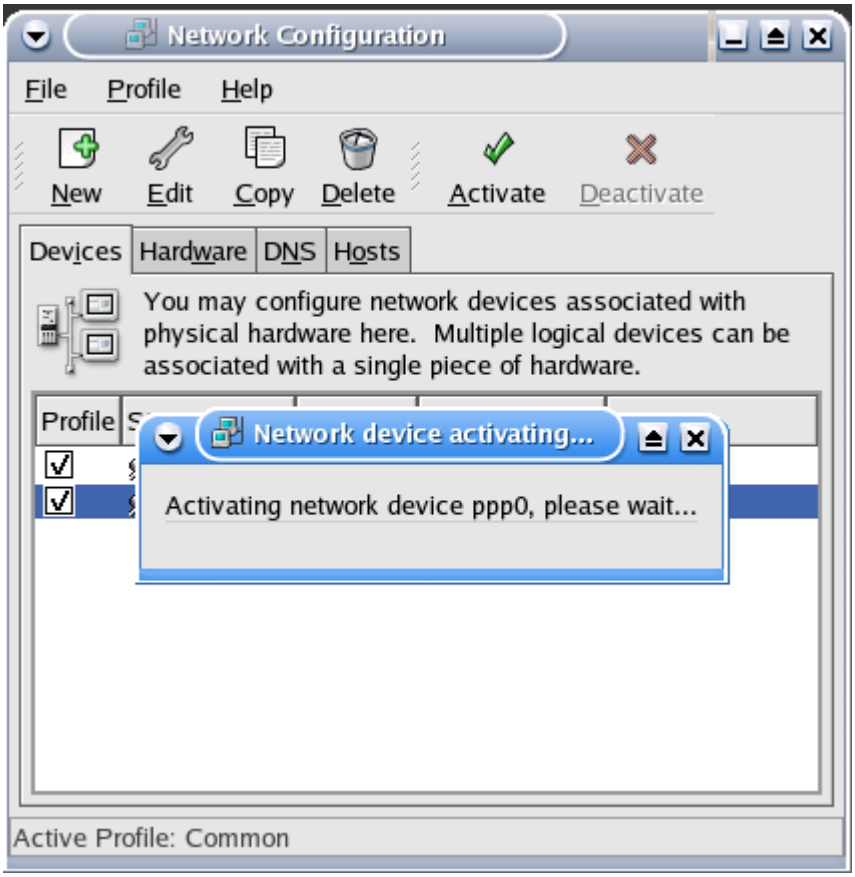
Đến đây chúng ta đã hoàn tất bước cài đặt modem.

5.2.2. Quay số

Tại màn hình này chọn giao diện ppp0 và click vào nút lệnh Avtive

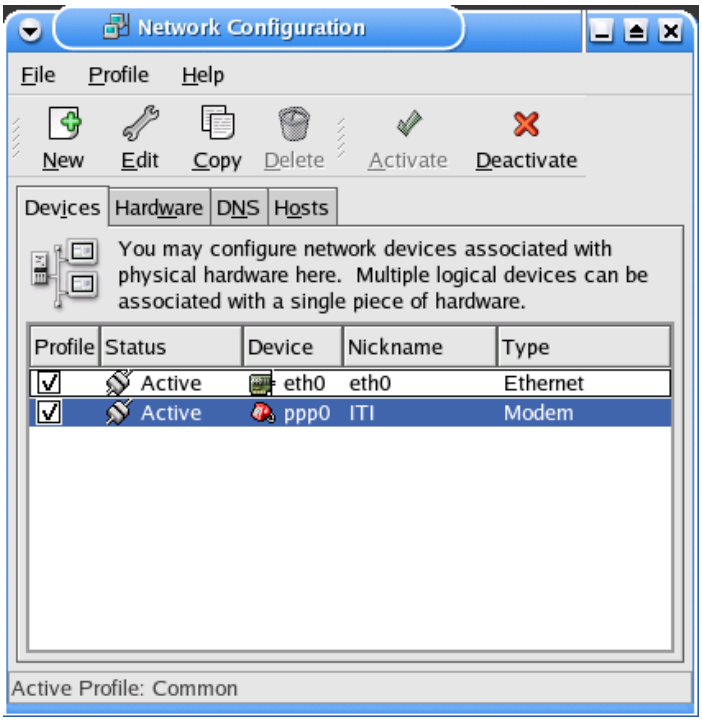


Máy tính bắt đầu quay số. file log sẽ được cất vào /var/log/message.

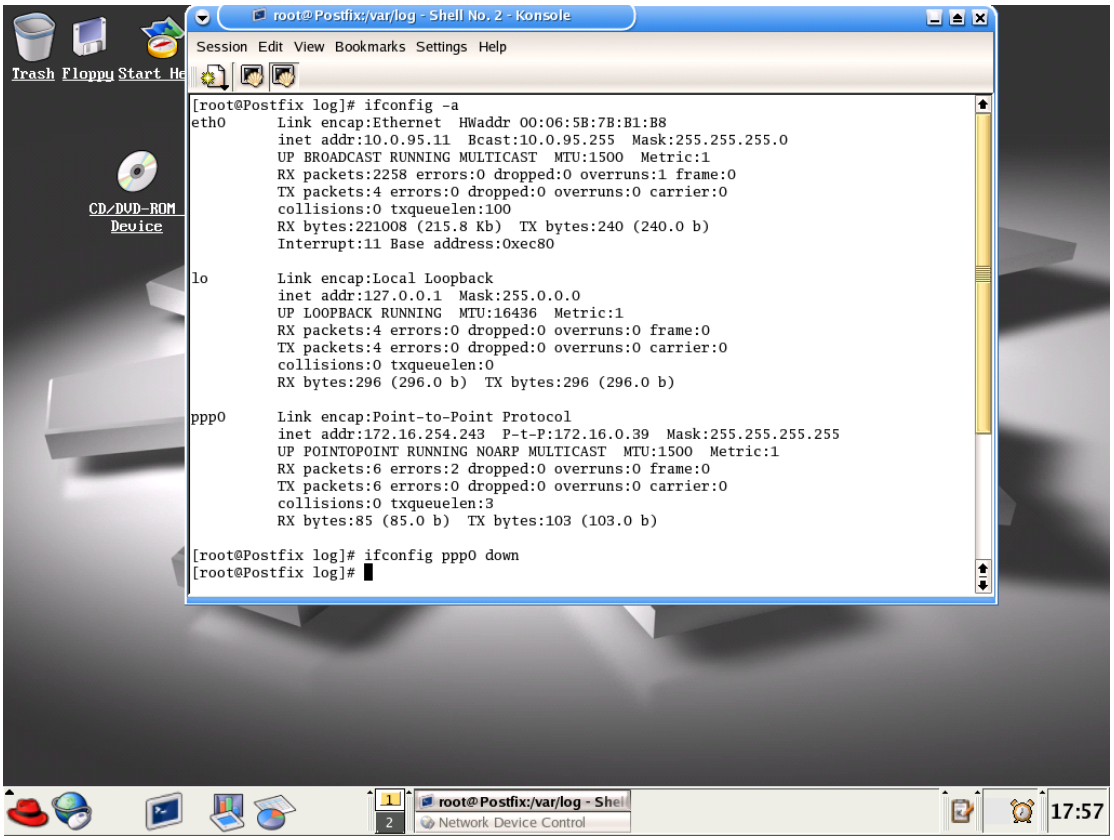


yess wait...

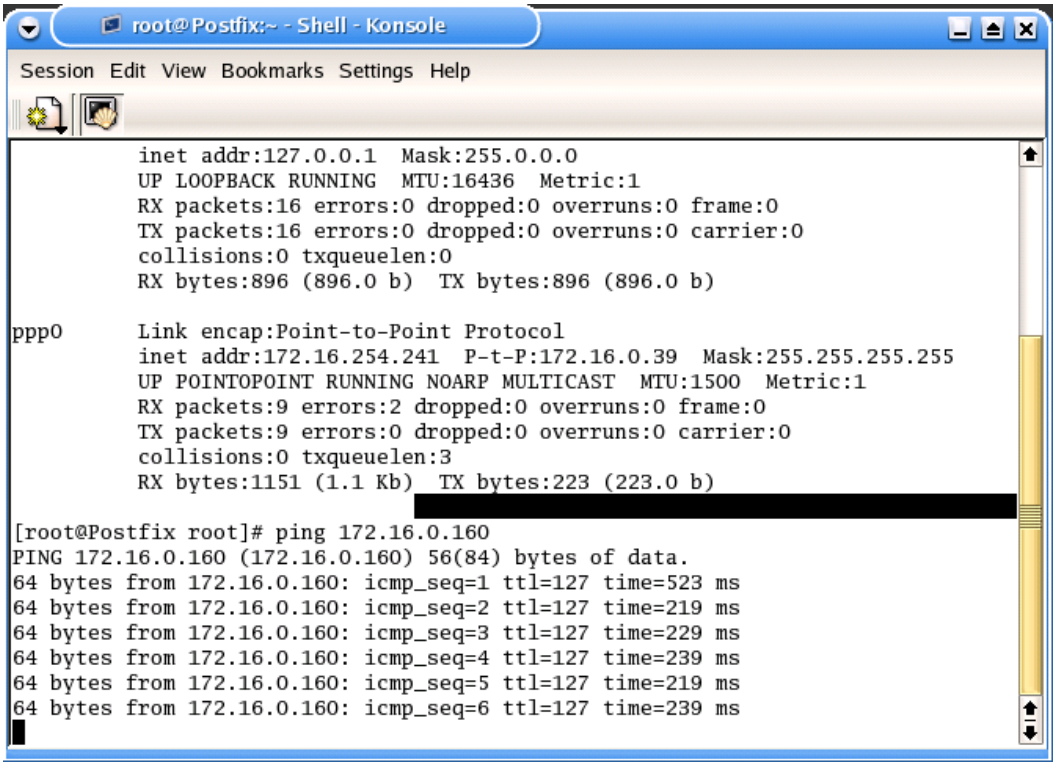
Khi xong màn hình network configuration sẽ báo giao diện ppp0 là active.



Có thể kiểm tra địa chi IP động và máy cung cấp DHCP qua lệnh “ifconfig -a”



Lúc này kết nối coi như đã được thiết lập, có thể dùng ping đết kiểm tra.



Bây giờ thì chúng ta có thể truy cập internet thông qua trình duyệt.

6. Lập trình shell.

Lập trình shell là một trong những công cụ hữu ích nhất cho việc quản trị hệ thống. Khả năng viết một chương trình ngắn để hoàn thành một công việc đòi hỏi nhiều thời gian mạnh hơn rất nhiều so với các công cụ quản trị Linux khác được biết đến. Lập trình Shell có thể làm cho cuộc sống của người quản trị trở lên dễ thở hơn và nó là một kỹ năng bắt buộc đối với người quản trị Linux. Có thể nhận thấy có rất nhiều công việc của những người quản trị hệ thống đối mặt hàng ngày liên quan đến các file và thư mục. Bất cứ khi nào bạn phải xử lý với một số lượng lớn các file, lập trình shell sẽ làm cho công việc của bạn trở lên dễ dàng hơn. Phần này sẽ chỉ cho bạn cách lập trình Shell cơ bản, nó có thể giúp cho bạn thực hiện các công việc hàng ngày.

6.1. Tạo và chạy chương trình Shell

Nó một cách đơn giản nhất, lập trình shell chỉ là các file chứa một hoặc nhiều câu lệnh shell hay câu lệnh Linux. Bạn có thể sử dụng các chương trình đơn giản thực hiện các công việc lặp đi lặp lại, để thay cho hai hay nhiều câu lệnh luôn luôn được thực thi cùng nhau bằng một câu lệnh, để tự động cài đặt các chương trình khác, và để viết các ứng dụng tương tác đơn giản.

Để tạo một chương trình shell, bạn phải tạo một file sử dụng một trình soạn thảo và đưa các câu lệnh shell hay Linux mà bạn muốn được thực thi vào trong file. Giả sử rằng bạn có một ổ CD-ROM đã được gắn vào hệ thống Linux. Thiết bị CD-ROM này được gắn vào hệ thống khi hệ thống được khởi động lần đầu. Nếu bạn cần thay đổi đĩa CD đã có trong ổ CD bằng một đĩa CD mới. Một cách để bạn thực hiện được công việc này là bạn "nhả" ổ CD-ROM khỏi hệ thống sử dụng câu lệnh umount, và sau đó gắn lại ổ sử dụng câu lệnh mount . Các câu lệnh chỉ ra ở dưới đầy cho bạn thấy tuần tự các bước thực hiện:

```
umount /dev/cdrom

mount /dev/cdrom /cdrom
```

Thay việc gõ cả hai câu lệnh mỗi lần bạn thay đổi đĩa CD, bạn có thể tạo một chương trình shell thực hiện cả hai câu lệnh này cho bạn. Để tạo chương trinh shell này bạn đưa cả hai câu lệnh vào trong một file có tên là remount (hoặc một tên bất kỳ nào khác mà bạn muốn).

Có một vài cách để thực hiện các câu lệnh trong file remount. Cách thứ nhất là bạn thay đổi thuộc tính cho file này có thể thực thi bằng cách thực hiện câu lệnh sau:

```
chmod +x remount
```

Câu lệnh này thay đổi quyền của file làm cho file có thể thực thi. Để chạy chương trình shell mới, gõ remount trên dòng lệnh.

Chương trình shell remount phải nằm trong một thư mục có trong đường dẫn tìm

kiếm của bạn, nếu không hệ thống sẽ không tìm thấy chương trình để thực thi. Nếu bạn không chay được chương trình bởi vì file đó không được tìm thấy, hãy xác định đường dẫn. Hoặc nếu bạn sử dụng tcsh để viết chương trình, dòng đầu tiên của chương trình shell phải bắt đầu với # để tcsh nhận ra nó như một file chương trình tcsh. Thực ra, cách an toàn (đảm bảo) nhất là ở dòng đầu của mỗi chương trình shell bạn thêm #!/bin/sh để đảm bảo chương trình shell được thực thi như một tiến trình Bourne shell. Điều này ngăn chặn nhiều vấn đề với ngôn ngữ lập trình C, shell sẽ cố gắng thông dịch cú pháp Bourne shell.

Một cách khác là bạn có thể thực thi chương trình shell là chạy shell mà chương trình được viết theo nó và tên chương trình như một khai báo cho shell. Trong trường hợp một chương trinh tcsh, bạn thực hiện câu lệnh sau:

```
tcsh remount
```

Câu lệnh này chạy một shell mới và nói cho nó thực thi các câu lệnh trong file remount.

Cách thứ ba để thực thi các câu lệnh trong một file chương trình shell là sử dụng câu lệnh . (dấu chấm) với cả shell pdksh và bash hoặc câu lệnh source trong shell tcsh. Các câu lệnh này nói cho shell thực thi file được truyền vào như đối số. Ví dụ, bạn có thể sử dụng câu lệnh sau để nói cho bash hoặc pdksh thực thi các câu lệnh trong file remount:

```
. remount
```

Để làm tương tự đối với tcsh, sử dụng câu lệnh sau:

```
source remount
```

Ví dụ sau trình bầy một tình huống khác, trong đó việc sử dụng chương trình shell sẽ giúp tiết kiệm rất nhiều thời gian. Giả sử rằng bạn đã phải làm việc với ba file khác nhua trong một thư mục mỗi ngày, và bạn muốn dự phòng ba file này vào một đĩa mềm vào cuối mỗi ngày. Để thực hiện được công việc này, bạn phải gõ một loạt các lệnh:

```
mount -t msdos /dev/fd0 /a

cp file1 /dev/fd0

cp file2 /dev/fd0

cp file3 /dev/fd0
```

Một cách dự phòng các file là gắn ổ đĩa mềm vào hệ thống và sau đó gõ ba câu lệnh copy, mỗi lệnh cho một file bạn muốn copy. Một cách đơn giản hơn là đưa bốn câu lệnh này vào trong một file có tên là backup và sau đó thực hiện câu lệnh backup khi bạn muốn copy ba file này vào đĩa mềm.

Bạn vẫn phải đảm bảo chương trình file shell backup có thể thực thi và nằm trong một thư mục mà có trong đường dẫn của bạn trước khi chạy câu lệnh. Bạn hãy cẩn thận khi sử dụng một tên file, nó có thể tương ứng với tên của một câu lênh hệ thống. Ví dụ, nếu có một chương trình được gọi là backup trong đường dẫn mà shell tìm kiếm trước khi đọc thư mục hiện tại, câu lệnh đó có thể được thi thay cho file câu lệnh shell. Vì lý do này, hãy cố sử dụng các tên file cho kịch bản shell của bạn không gần với các câu lệnh Linux.

6.2. Sử dụng các biến

Cũng giống như với hầu hết các ngôn ngữ lập trình, việc sử dụng các biến là rất quan trọng trong các chương trình shell. Tất nhiên, bạn đã được nhìn thấy một vài kiểu biến trước đó. Một vài ví dụ nói chung về biến được sử dụng là biến PATH và biến TERM. Các biến này là các ví dụ về các biến shell sẵn có, là các biến được định nghĩa bởi chương trình shell mà bạn đang sử dụng. Phần này miêu tả cách làm thế nào để bạn tạo các biến của chính bạn và sử chúng trong một vài chương trình shell.

6.2.1. Gán một giá trị cho một biến

Trong cả ba shell được cung cấp bởi Linux (shell Bourne, Korn, và C), bạn có thể gán một giá trị cho một biến bằng cách gõ tên biến theo su bởi dấu bằng và sau đó gõ giá trị mà bạn muốn gán cho biến. Ví dụ, để gán một giá trị 5 cho một biến có tên là count, vào câu lệnh sau trong bash hoặc pdksh:

```
count=5
```

Với tcsh, vào câu lệnh sau để đạt được kết quả tương tự:

```
set count = 5
```

Khi thiết lập một biến cho shell bash và pdksh, hãy chắc chắn rằng không có dấu cách ở cả hai bên dấu bằng. Với tcsh, điều này không quan trọng.

Bởi vì ngôn ngữ shell là một ngôn ngữ kịch bản phi kiểu, bạn không phải khai báo biến như bạn có thể đã từng làm điều này trong lập trình C hay Pascal. Bạn có thể sử dụng cùng một biến để lưu trữ xâu ký tự hay số nguyên. Bạn lưu một chuỗi ký tự vào trong một biến cũng giống như việc bạn lưu một số nguyên vào một biến, như có thể thấy trong ví dụ dưới đây:

```
name=Garry (for pdksh and bash)

set name = Garry (for tcsh)
```

Sau khi bạn lưu một giá trị vào một biến, bạn làm thế nào để có thể lấy giá trị đó trở lại? Bạn đặt trước tên biến với dấu đô la ($). Để in giá trị được lưu trữ trong biến count ra màn hình, vào câu lệnh sau:

```
echo $count
```

Nếu bạn quên dấu $ trước câu lệnh, lệnh echo sẽ hiển thị từ “count” trên màn hình.

6.2.2. Tham số về các biến Shell cĩ sẵn

Khi bạn chạy chương trình shell yêu cầu hay hỗ trợ một số các tùy chọn dòng lệnh, mỗi tùy chọn này được lưu trữ trong một đối số. Đối số đầu tiên được lưu trữ trong một biến có tên là 1, đối số thứ hai được lưu trữ trong biến có tên là 2, và tiếp tục như thế. Shell đặt tên các biến này, vì vậy bạn không thể đặt tên như thế cho các biến mà bạn định nghĩa. Để lấy giá trị từ các biến này, bạn phải đặt trước tên biến với một dấu $ như bạn làm đối các biến mà bạn định nghĩa.

Chương trình shell reverse dưới đây chờ nhận hai đối số. Chương trình lấy hai đối số dòng lệnh và in ra đối số thứ hai ở dòng đầu tiên và đối số đầu tiên ở dòng thứ hai:

```
echo "$2"

echo "$1"
```

Nếu bạn gọi tới chương trình bằng cách gõ dòng lệnh sau

```
reverse hello there
```

Chương trình sẽ trả lại kết quả

```
there hello
```

Một số các biến shell quan trọng được xây dựng sẵn mà bạn cần biết khi làm việc nhiều với lập trình shell. Bảng 6.2.1 đưa ra danh sách các biến này và mô tả tóm tắt mỗi biến được sử dụng để làm gì.

Bảng 6.2.1 Các biến shell có sẵn.

Biến	*Sử dụng*
$#	Lưu số các đối số dòng lệnh được đưa vào chương trình shell
$?	Lưu giá trị tồn tại của câu lệnh được thực thi sau cùng
$0	Lưu từ đầu tiên của câu lệnh được đưa vào, đó là tên của chương trình shell
$*	Lưu tất cả các đối số được đưa vào từ dòng lệnh ("$1 $2 ...")

"$@"	Lưu tất cả các đối số được đưa vào từ dòng lệnh, có dấu nháy kép riêng ("$1" "$2" ...)

6.3. Sử dụng dấu trích dẫn

Việc sử dụng các dấu trích dẫn là rất quan trọng trong lập trình shell. Shell sử dụng cả hai kiểu dấu trích dẫn và ký tự và dấu gạch chéo ngược để thực hiện các chức năng khác nhau. Cả dấu nháy kép (""), dấu nháy đơn ('), và dấu gạch ngược (\) được sử dụng để ẩn các ký tự đặc biệt trong shell. Các dấu nháy có một ý nghĩa đặc biệt trong shell và nó không nên sử dụng để chứa các xâu. Mỗi một phương thức có một mức độ che dấu khác nhau các ký tự đặc biệtk trong shell.

Khi bạn bao quanh các ký tự với dấu nháy kép, tất cả các ký tự trống được ẩn trong shell, nhưng tất cả các ký tự khác vẫn được thông dịch. Kiểu dấu nháy kép này sử dụng hữu ích nhất khi bạn gán các chuỗi chứa nhiều hơn một từ vào một biến. Ví dụ, để gán chuỗi *hello there* cho biến greeting, nhập vào câu lệnh sau:

```
greeting="hello there" (in bash and pdksh)

set greeting = "hello there" (in tcsh)
```

Câu lệnh này lưu trữ toàn bộ chuỗi *hello there* vào biến greeting như một từ. Nếu bạn gõ vào câu lệnh mà không sử dụng dấu nháy kép, bash và pdksh có thể không hiểu câu lệnh và có thể trả lại một thông báo lỗi, và tcsh có thể gán giá trị hello cho biến greeting và bỏ qua phần đuôi của dòng lệnh.

Dấu nháy đơn là hình thức sử dụng mạnh nhất của dấu nháy. Chúng ẩn tất cả các ký tự đặc biệt trong shell. Kiểu dấu nháy này hữu ích nếu câu lệnh của bạn đưa vào có dụng ý cho một chương trình hơn là cho shell. Ví dụ, bạn có thể sử dụng dấu nháy đơn để ghi chuỗi hello there, nhưng bạn không thể sử dụng phương thức này trong một số trường hợp. Ví dụ, nếu chuỗi được gán cho biến greeting chứa biến khác, bạn phải sử dụng dấu nháy kép. Giả sử rằng bạn muốn đưa tên của người sử dụng trong biến greeting. Bạn gõ câu lệnh sau:

```
greeting="hello there $LOGNAME" (for bash and pdksh)

set greeting="hello there $LOGNAME" (for tcsh)
```

Biến LOGNAME là một biến shell chứa tên đăng nhập của người sử dụng Linux đã đăng nhập hệ thống.

Câu lệnh này lưu trữ giá trị *hello there root* vào trong biến greeting nếu bạn đã đăng nhập vào Linux là root. Nếu bạn cố ghi câu lệnh này sử dụng dấu nháy đơn, dấu nháy

đơn sẽ làm ẩn dấu $ trong shell, và shell không biết rằng nó được yêu cầu thực hiện thay thế một biến. Kết quả, biến greeting được gán giá trị *hello there $LOGNAME*.

Sử dụng dấu gạch ngược là cách thứ ba để che dấu các ký tự đặc biệt trong shell. Giống như phương thức dấu nháy đơn, dấu gạch ngược ẩn tất cả các ký tự đặc biệt trong shell, nhưng nó chỉ có thể ẩn một ký tự tại một thời điểm, chứ không phải một nhóm các ký tự. Bạn có thể viết lại ví dụ greeting sử dụng dấu gạch ngược thay cho dấu nháy kép bằng cách sử dụng câu lệnh sau:

```
greeting=hello\ there (for bash and pdksh)

set greeting=hello\ there (for tcsh)
```

Trong câu lệnh này, dấu gạch ngược ẩn ký tự trống trong shell và chuỗi *hello there* được gán cho biến greeting.

Dấu gạch ngược thường được sử dụng nhiều nhất khi bạn muốn ẩn chỉ một ký tự trong shell. Vấn đề này xuất hiện khi bạn muốn đưa vào một ký tự đặc biệt trong một chuỗi. Ví dụ, để lưu giá của một hộp đĩa máy tính vào một biến có tên là disk_price, sử dụng câu lệnh sau.

```
disk_price=\$5.00 (for bash and pdksh)

set disk_price = \$5.00 (tcsh)
```

Dấu gạch ngược trong ví dụ này ẩn dấu đô la trong shell. Nếu dấu gạch ngược không có ở đó, shell có thể cố tìm một biến có tên là 5 và thực hiện một phép thay thế biến trên biến đó. Nếu không có biến tên là 5 được định nghĩa, shell có thể một gán giá trị .00 cho biến disk_price. (shell này có thể thay thế một giá trị rỗngcho biến $5) Bạn cũng có thể sử dụng dấu nháy đơn trong ví vụ disk_price để ẩn ký hiệu $ trong shell.

Dấu nháy ngược (``) thực hiện một chức năng khác. Bạn sử dụng chúng khi bạn muốn sử dụng các kết quả của một câu lệnh trong một câu lệnh khác. Ví dụ, để đặt giá trị của biến contents bằng danh sách các file có trong thư mục hiện tại, gõ câu lệnh sau:

```
contents=`ls` (for bash and pdksh)

set contents = `ls` (for tcsh)
```

Câu lệnh này thực thi câu lệnh ls và lưu kết quả của câu lệnh vào biến contents . Như sẽ được chỉ ra trong các đoạn sau, đặc điểm này có thể rất hữu ích khi bạn muốn ghi kết quả của một chương trình shell thực hiện một vài hoạt động vào trong một câu lệnh khác.

6.4. Sử dụng câu lệnh test

Trong bash và pdksh, câu lệnh test được sử dụng để tính giá trị của một biểu thức có điều kiện. Thông thường, bạn sử dụng câu lệnh test để tính giá trị điều kiện trong một

lệnh có điều kiện hoặc tính giá trị đầu vào hay điều kiện tồn tại cho một câu lệnh lặp. Câu lệnh test có cú pháp sau:

```
test expression
```

hoặc

```
[ expression ]
```

Bạn có thể sử dụng một vái toán tử có sẵn với câu lệnh test. Các toán tử này được phân loại thành bốn nhóm khác nhau: các toán tử xâu, các toán tử số, các toán tử file, và các toán tử logic.

Bạn sử dụng các toán tử xâu để tính giá trị biểu thức xâu. Bảng 6.4.1 đưa ra danh sách các toán tử xâu mà ba ngôn ngữ lập trình shell hỗ trợ.

Bảng 6.4.1 Các toán tử chuỗi cho câu lệnh test.

Toán tử	*Ý nghĩa*
str1 = str2	Trả lại giá trị true nếu str1 giống với str2
str1 != str2	Trả lại giá trị true nếu str1 không giống str2
str	Trả lại giá trị true nếu str khác rỗng
-n str	Trả lại giá trị true nếu độ dài của str lớn hơn 0
-z str	Trả lại giá trị true nếu độ dài của str bằng 0

Các toán tử số thực hiện các chức năng tương tự các toán tử string ngoại trừ việc chúng hoạt động trên các đối số kiểu số. Bảng 6.4.2 liệt kê danh sách các toán tử số được sử dụng trong câu lệnh test.

Bảng 6.4.2 Các toán tử số cho câu lệnh test.

Toán tử	*Ý nghĩa*
int1 -eq int2	Trả lại giá trị true nếu int1 bằng int2
int1 -ge int2	Trả lại giá trị true nếu int1 lớn hơn hoặc bằng int2
int1 -gt int2	Trả lại giá trị true nếu int1 lớn hơn int2
int1 -le int2	Trả lại giá trị true nếu int1 nhỏ hơn int2
int1 -lt int2	Trả lại giá trị true nếu int1 nhỏ hơn int2
int1 -ne int2	Trả lại giá trị true nếu int1 không bằng int2

Bạn sử dụng các toán tử file cho câu lệnh test để thực hiện các chức năng chẳng hạn như kiểm tra để xem các file có tồn tại hay không và kiểm tra để xem file thuộc loại nào, file được đưa vào như một đối số cho câu lệnh test. Bảng 6.4.3 đưa ra danh sách các toán tử file cho câu lệnh test.

Bảng 6.4.3 Các toán tử File cho câu lệnh test.

Toán tử	*Ý nghĩa*
-d file	Trả lại giá trị true nếu file được xác định là một thư mục
-f file	Trả lại giá trị true nếu file được xác định là một file thông thường
-r file	Trả lại giá trị true néu file xác định là có thể đọc bởi tiến trình
-s file	Trả lại giá trị true nếu file xác định có độ dài khác 0
-w file	Trả lại giá trị true nếu file có thể ghi được bởi tiến trình
-x file	Trả lại giá trị true nếu file xác định là có thể thực thi

Bạn sử dụng các toán tử logic cho câu lệnh test để kết hợp các toán tử số, xâu, hay file hoặc phủ định một toán tử đơn số, xâu, hoặc file. Bảng 6.4.4 đưa ra danh sách cá toán tử logic cho câu lệnh test.

Bảng 6.4.4 Các toán tử Logic chó câu lệnh test.

Toán tử	*Ý nghĩa*
! expr	Trả lại giá trị true nếu expr khác true
Expr1 -a expr2	Trả lại giá trị true nếu expr1 và expr2 là true
Expr1 -o expr2	Trả lại giá trị true nếu expr1 hoặc expr2 là true

Shell tcsh không có câu lệnh test, nhưng các biểu thức của tsch thực hiện các chức năng tương tự. Các toán tử tcsh hỗ trợ hầu hết giống như được hỗ trợ trong ngôn ngữ C. Bạn thường sử dụng các biểu tức này trong các cau lệnh if và while. Trong đoạn sau,phần "Sử dụng các lệnh có điều kiện " và "Sử dụng các lệnh lặp " sẽ nói về các câu lệnh này. Giống như câu lệnh test trong bash và pdksh, các biểu thức trong tcsh hỗ trợ các toán tử số, xâu, file, và logic. Bảng 6.4.5 đưa ra danh sách các toán tử được hỗ trợ trong cá biểu thức của tcsh.

Bảng 6.4.5 Các tóan tử số cho for các biểu thức tcsh.

Toán tử	*Ý nghĩa*
int1 <= int2	Trả lại giá trị true nếu int1 nhỏ hơn int2
int1 >= int2	Trả lại giá trị true nếu int1 lớn hơn hoặc bằng int2
int1 < int2	Trả lại giá trị true nếu int1 nhỏ hơn int2
int1 > int2	Trả lại giá trị true nếu int1 lớn hơn int2

Bảng 6.4.6 đưa ra danh sách các toán tử xâu mà các biểu thức của tcsh hỗ trợ.

Table 6.4.6. Các toán tử xâu cho các biểu thức của tcsh.

Toán tử	*Ý nghĩa*

str1 == str2	Trả lại giá trị true nếu str1 bằng str2
str1 != str2	Trả lại giá trị true nếu str1 không bằng str2

Bảng 6.4.7 đưa ra danh sách cá toán tử file mà các biểu thức tcsh hỗ trợ.

Bảng 6.4.7 Các toán tử File cho các biểu thức tcsh.

Toán tử	*Ý nghĩa*
-r file	Trả lại giá trị true nếu file có thể đọc được
-w file	Trả lại giá trị true nếu file có thể ghi được
-x file	Trả lại giá trị true nếu file có thể thực thi
-e file	Trả lại giá trị true nếu file tồn tại
-o file	Trả lại giá trị true nếu file được sở hữu bởi người sử dụng hiện tại
-z file	Trả lại giá trị true nếu file có kích thước bằng 0
-f file	Trả lại giá trị true nếu file là file thông thường
-d file	Trả lại giá trị true nếu file là một thư mục

Bảng 6.4.8 Đưa ra danh sách các toán tử logic được hỗ trợ trong các biểu thức của tcsh.

Table 6.4.8 Các toán tử Logical cho cá biểu thức của tcsh.

Toán tử	*Ý nghĩa*
exp1 || exp2	Trả lại giá trị true nếu exp1 là true hoặc exp2 là true
exp1 && exp2	Trả lại giá trị true nếu cả hai exp1 và exp2 là true
! exp	Trả lại true nếu exp khác true

6.5. Sử dụng các câu lệnh rẽ nhánh

Trong các shell bash, pdksh và tcsh, mỗi shell có hai hình thức khác nhau của câu lệnh rẽ nhánh. Bạn sử dụng các lệnh này để thực thi các phần khác nhau của chương trình shell phụ thuộc vào các điều kiện nhất định có đúng hay không. Với hầu hết các lệnh thực hiện, cú pháp cho các câu lệnh này khác nhau giữa các shell.

6.5.1. Lệ nh if

Tất cả ba shell đều hỗ trợ các câu lệnh if-then-else statements lồng nhau. Các lệnh này cung cấp cho bạn cách thực hiện các câu lệnh test điều kiện phức tạp trong chương trình shell của bạn. Cú pháp của lệnh if trong bash và pdksh là giống nhau:

```
if [ expression ]

then
```

```
 commands
elif [ expression2 ]
 commands
else
 commands
fi
```

Chú ý rằng shell bash và pdksh sử dụng đảo ngược của tên câu lệnh trong hâu hết các lệnh phức tạp để kết thúc câu lệnh. Trong lệnh bên trên, từ khóa fi được sử dụng đề làm kí hiệu kết thúc cho câu lệnh if.

Cả hai mệnh đề elif và else đều là các phần tùy chọn của lệnh if. Lệnh elif là rút gọn của else if. Lệnh này được thực thi nếu các biểu thức nằm trong lệnh if hoặc tất cả các biểu thức trong các lệnh elif ở trước đó đều không có giá trị true. Các câu lệnh nằm trong lệnh else được thực thi chỉ nếu không một biểu thức nào trong mệnh đề if và trong bất kỳ mệnh đề elif nào có giá trị true.

Trong tcsh, lệnh if có hai dạng khác nhau. Dạng thứ nhất cung cấp cùng một chức năng như lệnh if trong bash và pdksh. Dạng này của lệnh if có cú pháp như sau:

```
if (expression1) then
 commands
else if (expression2) then
 commands
else
 commands
endif
```

Lại một lần nữa các phần if và else của lệnh if là tùy chọn. Lệnh này cũng có thể được viết với elif. Nếu mã ở bên trên trình bày toán bộ chương trình tcsh, nó nên bắt đầu với dòng sau để đảm bảo chạy hoàn hảo:

```
#!/bin/sh
```

Dạng thứ hai của lệnh if mà tcsh cung cấp là biến đổi đơn giản của lệnh if dạng đầu tiên. Dạng này của lệnh if chỉ tính giá trị một biểu thức đơn. Nếu biểu thức là true nó sẽ thực thi câu lệnh đơn. Nếu biểu thức là false, không có điều gì xảy ra. Cú pháp cho dạng này của lệnh if là như sau.

```
if (expression) command
```

Bên dưới là một ví dụ về sử dụng lệnh if trong bash hay pdksh. Lệnh này kiểm tra xem có một file có tên là a .profile trong thư mục hiện tại hay không:

```
if [ -f .profile ]
then
 echo "There is a .profile file in the current directory."
else
 echo "Could not find the .profile file."
fi
```

Cũng với ví dụ trên sử dụng cú pháp của tcsh như sau:

```
#
if ( { -f .profile } ) then
 echo "There is a .profile file in the current directory."
else
 echo "Could not find the .profile file."
endif
```

Chú ý rằng trong ví dụ tcsh dòng đầu tiên bắt đầu với ký tự #. Ký hiệu này được yêu cầu để tcsh nhận ra file chứa các câu lệnh là một file kịch bản tcsh.

6.5.2. Lệ nh case

Lệnh case cho phép bạn so sánh một mẫu với một số các mẫu khác và thực thi một khối mã nếu một sự giống nhau được tìm thấy. Lệnh case trong shell mạnh hơn lệnh case trong Pascal hay lệnh switch trong C. Với lệnh shell trong case , bạn có thể so sánh các xâu với các kí tự đại diện trong chúng; bạn có thể chỉ có thể so sánh các kiểu được liệt kê hoặc các giá trị số nguyên trong Pascal và C.

Cú pháp cho lệnh case trong bash và pdksh là như sau:

```
case string1 in
 str1)
 commands;;
 str2)
 commands;;
 *)
```

```
 commands;;
esac
```

String1 được so sánh với str1 và str2. Nếu một trong các xâu này hộ với string1, các câu lệnh bên dưới nó cho đến khi gặp hai dấu chấm phẩy(;;) được thực hiện. Nếu không có xâu nào (str1 hoặc str2) hợp với string1, các câu lệnh kết hợp với dấu hoa thị được thực thi. Các câu lệnh này là điều kiện case mặc định bởi vì dấu hoa thị hợp với tất cả các xâu.

Câu lệnh trong tcsh tương đương với câu lệnh case trong bash và pdksh được gọi là lệnh switch. Lệnh này gần gũi với cú pháp lệnh switch trong C. Cú pháp cho lệnh switch là như sau:

```
switch (string1)
 case str1:
 statements
 breaksw
 case str2:
 statements
 breaksw
 default:
 statements
 breaksw
endsw
```

Lệnh này xử lý giống nư cách sử lý của lệnh case trong bash và pdksh. Mỗi xâu trong từ khóa case được so sánh với string1. Nếu xâu bất kỳ trong các xâu trên hợp với string1, các mã bên dưới nó cho đến khi gặp từ khóa breaksw keyword được thực hiện. Nếu không có xâu nào phù hợp, các mã ở bên dưới từ khóa default cho đến khi gặp từ khóa breaksw được thực thi.

Mã bên dưới là một ví dụ về lệnh case trong shell bash hay pdksh. Mã này kiểm tra xem tùy chọn đầu tiên trong dòng lệnh là -i hay -e. Nếu nó là -i, chương trình đếm số các dòng trong một file xác định bởi tùy chọn thứ hai trong dòng lệnh bắt đầu với ký tự i. Nếu tùy chọn thứ nhất là -e, chương trình đếm số các dòng trong file được xác định bởi tùy chọn thứ hai của dòng lệnh bắt đầu với ký tự e. Nếu tùy chọn thứ nhất của dòng lệnh khác -i và khác -e, chương trình sẽ in ra thông tin báo lỗi trên màn hình.

```
case $1 in
 -i)
```

```
 count=`grep ^i $2 | wc -l`
 echo "The number of lines in $2 that start with an i is $count"
 ;;
 -e)
 count=`grep ^e $2 | wc -l`
 echo "The number of lines in $2 that start with an e is $count"
 ;;
 * )
 echo "That option is not recognized"
 ;;
esac
```

Ví dụ tương tự được viết theo cú pháp tcsh:

```
# remember that the first line must start with a # when using tcsh
switch ( $1 )
 case -i | i:
 set count = `grep ^i $2 | wc -l`
 echo "The number of lines in $2 that begin with i is $count"
 breaksw
 case -e | e:
 set count = `grep ^e $2 | wc -l`
 echo "The number of lines in $2 that begin with e is $count"
 breaksw
 default:
 echo "That option is not recognized"
 breaksw
endsw
```

6.6. Sử dụng các lệnh lặp

Ngôn ngữ shell cũng cung cấp lệnh lặp mà thường được sử dụng nhất. Các lệnh lặp này được thao tác khi bạn cần thực hiện một hành động lặp đi lặp lại, chẳng hạn như khi bạn xử lý danh sách các file.

6.6.1. Lệ nh for

Lệnh for thực thi các câu lệnh chứa trong nó một số lần. Lệnh for có hai dạng khác nhau trong bash và pdksh. Dạng thứ nhất của lệnh for mà bash và pdksh hỗ trợ có cú pháp như sau:

```
for var1 in list
do
 commands
done
```

Trong dạng này, lệnh for thực thi một lần cho mỗi phần tử nằm trong danh sách. Danh sách này có thể được thay đổi chứa các từ được phân biệt với nhau bởi dấu cách, hoặc nó có thể là một danh sách các giá trị được gõ trực tiếp vào trong câu lệnh. Mỗi lần qua vòng lặp, biến var1 được gán cho phần tử hiện tại trong danh sách và tiếp tục cho đến khi phần tử cuối cùng trong danh sách.

Dạng thứ hai của lệnh for có cú pháp như sau:

```
for var1
do
 statements
done
```

Trong dạng này, lệnh for thực thi một lần cho mỗi phần tử nằm trong biến var1. Khi bạn sử dụng cú pháp này của lệnh for, chương trình shell giả sử rằng biến var1 chứa tất cả các đối số được đưa vào trong chương trình shell từ dòng lệnh. Điển hình, dạng này của lệnh for là tương đương với viết các lệnh sau:

```
for var1 in "$@"
do
 statements
done
```

Tương đương với lệnh for trong tcsh là lệnh foreach. Nó xử lý tương tự như lệnh for trong bash và pdksh. Cú pháp của lệnh foreach như sau:

```
foreach name (list)
 commands
end
```

Một lần nữa, nếu mã này là một chương trình hoàn thiện, nó nên bắt đầu với kí hiệu # (và tốt nhất là #!/bin/sh để buộc thực thi theo Bourne shell). Dưới đây là một ví dụ về sử dụng lệnh for trong bash hay pdksh. Ví dụ này lấy các tùy chọn dòng lệnh số lượng bất kỳ các file text. Chương trình đọc mỗi file trong các file này, chuyển đổi tất cả các ký tự thành chữ hoa, và sau đó lưu trữ kết quả trong một file có cùng tên nhưng có phần mở rộng là .caps.

```
for file

do

tr a-z A-Z < $file >$file.caps

done
```

Chương trình sau là một ví dụ tương tự được viết theo ngôn ngữ shell tcsh:

```
#

foreach file ($*)

 tr a-z A-Z < $file >$file.caps

end
```

6.6.2. Lệ nh while

Một lệnh lặp khác được đưa vào ngôn ngữ lập trình shell là lệnh while. Lệnh này thực thi một khối các câu lệnh theo một điều kiện nào đó. Cú pháp của lệnh while trong bash và pdksh là như sau:

```
while expression

do

 statements

done
```

Cú pháp cho lệnh while trong tcsh là như sau:

```
while (expression)

 statements

end
```

Dưới đây là một ví dụ về lệnh while theo ngôn ngữ shell bash hay pdksh. Chương trình này đưa ra danh sách các đối số được đưa vào chương trình cùng với số các đối số.

```
count=1

while [ -n "$*" ]

do

 echo "This is parameter number $count $1"

 shift

 count=`expr $count + 1`

done
```

Lệnh shift chuyển đổi số dòng lệnh lên một sang bên trái (xem đoạn sau"Lệnh shift" để biết thêm thông tin). Chương trình bên dưới tương tự được viết cho ngôn ngữ tcsh:

```
#

set count = 1

while ( "$*" != "" )

 echo "This is parameter number $count $1"

 shift

 set count = `expr $count + 1`

end
```

6.6.3. Lệ nh until

Lệnh until có cú pháp và chức năng tương tự lệnh while. Chỉ có sự khác biệt thực sự giữa hai lệnh là lệnh until thực thi mã trong khối của nó khi giá trị của biểu thức là sai và lệnh while thực thi các khối lệnh cảu nó nếu biểu thức có giá trị là true. Cú pháp cho lệnh until trong bash và pdksh là như sau:

```
until expression

do

 commands

done
```

Để làm cho ví dụ được sử dụng với lệnh while làm việc với lệnh until, tất cả những gì bạn phải làm chỉ là phủ định điều kiện, như chỉ ra trong đoạn mã bên dưới:

```
count=1

until [ -z "$*" ]
```

```
do
 echo "This is parameter number $count $1"
 shift
 count=`expr $count + 1`
done
```

Chỉ có sự khác nhau trong ví dụ này là và ví dụ về lệnh while là tùy chọn -n của lệnh test, nó có nghĩa rằng xâu không có độ dài bằng 0, được thay bởi tùy chọn -z , nó có nghĩa là chuỗi có độ dài bằng 0. Trong thực tế, lệnh until ít được dùng bởi vì với bất kỳ lệnh until nào, bạn cũng có thể viết được bằng lệnh while. Lệnh until không được hỗ trợ trong tcsh.

6.6.4. Lệ nh shift

Tất cả các shell bash, pdksh, và tcsh đều hỗ trợ một lệnh gọi là lệnh shift. Lệnh shift chuyển các giá trị hiện tại được lưu trữ trong các đối số dòng lệnh lên một vị trí sang trái. Ví dụ, nếu các giá trị của các đối số là

```
$1 = -r $2 = file1 $3 = file2
```

và bạn thực hiện lệnh shift

```
shift
```

kết quả các đối số được đưa vào như sau:

```
$1 = file1 $2 = file2
```

Bạn có thể dịch chuyển các đối số qua nhiều hơn một vị trí bởi một số xác định vớikèm theo với lệnh shift. Lệnh sau dịch chuyển đối số lên hai vị trí:

```
shift 2
```

Lệnh này rất hữu ích khi có một chương trình shell cần phân tích các tùy chọn dòng lệnh. Các tùyd chọn thường được đặt trước bởi một dấu nối và một ký tự để chỉ ra tùy chọn nào được sử dụng. Bởi vì các tùy chọn luôn luôn được xử lý trong một vòng lặp của một loại câu lệnh, bạn sẽ thường muốn nhảy đến đối số tiếp theo một khi bạn đã xác định được tùy chọn nào nên được xử lý tiếp theo. Ví dụ, chương trình shell sau chờ hai tùy chọn dòng lệnh, một xác định một file đầu vào và một xác định một file đầu ra. Chương trình đọc file đầu vào, chuyển tất cả các ký tự trong file input thành chữ hoa, và sau đó lưu trữ kết quả trong file đầu ra xác định:

```
while [ "$1" ]
do
```

```
 if [ "$1" = "-i" ] then
 infile="$2"
 shift 2
 else if [ "$1" = "-o" ] then
 outfile="$2"
 shift 2
 else
 echo "Program $0 does not recognize option $1"
 fi
done
tr a-z A-Z <$infile >$outfile
```

6.6.5. Lệ nh select

Shell pdksh đưa ra một lệnh lặp mà bash và tcsh không hỗ trợ, lệnh select. Nó hơi khác với các lệnh lặp khác bởi vì nó không thực thi một khối mã lệnh shell theo một điều kiện true hoặc false. Những gì lệnh select làm là cho phép bạn tự động tạo các menu text đơn giản. Cú pháp của lệnh select như sau:

```
select menuitem [in list_of_items]
do
 commands
done
```

Khi bạn thực thi lệnh select, pdksh tạo một đối tượng menu được đánh số cho mỗi phần tử có trong list_of_items. list_of_items này có thể là một biến chứa nhiều hơn một phần tử, chẳng hạn như choice1 choice2 hoặc nó có thể là một danh sách các lựa chọn được gõ vào từ dòng lệnh, như trong ví dụ sau:

```
select menuitem in choice1 choice2 choice3
```

Nếu danh sách list_of_items is không được cung cấp, lệnh select sử dụng các đối số dòng lệnh cho lệnh thực hiện.

Khi người sử dụng của chương trình có chứa lệnh select chọn một trong số các phần tử của menu bằng cách gõ vào số tương ứng với nó, lệnh select lưu giá trị của phần tử được lựa chọn trong biến menuitem. Các lệnh trong khối do sau đó có thể thực hiện các hoạt động trên phần tử menu này.

Dưới đây là một ví dụ về việc sử dụng lệnh select như thế nào. Ví dụ này hiển thị ba phần tử của menu. Khi người sử dụng chọn một phần tử, chương trình sẽ hỏi bạn xem có phải phần tử đó được lựa chọn không, nếu người sử dụng gõ khác với y hoặc Y, chương trình sẽ hiển thị lại menu.

```
select menuitem in pick1 pick2 pick3

do

 echo "Are you sure you want to pick $menuitem"

 read res

 if [ $res = "y" -o $res = "Y" ]

 then

 break

 fi

done
```

Ví dụ này giới thiệu một vài lệnh mới. Lệnh read được sử dụng để lấy dữ liệu vào từ người sử dụng. Nó lưu bất kỳ cái gì người sử dụng gõ vào biến xác định. Lệnh break để kết thúc vòng lặp lệnh while, select, hoặc for.

6.6.6. Lệ nh repeat

Shell tcsh có một lệnh lặp không có trong pdksh hay bash. Lệnh này là lệnh repeat. Lệnh repeat thực thi câu lệnh đơn theo một số lần xác định. Cú pháp cho lệnh repeat là như sau:

```
repeat count command
```

Ví dụ sau của lệnh repeat lấy một tập hợp các số là các tùy chọn dòng lệnh và in ra số các dấu chấm lên màn hình. Chương trình này hoạt động như một chương trình minh họa rất thô sơ.

```
#

foreach num ($*)

 repeat $num echo -n "."

 echo ""

end
```

Bạn có thể viết lại lệnh repeat bất kỳ bằng lệnh while hay lệnh for; cú pháp repeat chỉ thuận tiện hơn mà thôi.

6.7. Sử dụng các hàm

Ngôn ngữ shell cho phép bạn dịnh nghĩa hàm của chính bạn. Các hàm này được định nghĩa giống như cách bạn định nghĩa các hàm trên ngôn ngữ lập trình C hay các ngôn ngữ lập trình khác. Thuận lợi chính của việc sử dụng hàm để tổ chức, tránh viết tất cả các mã shell của bạn trong một dòng. Mã được viết sử dụng các hàm có khuynh hướng dễ hơn trong việc đọc và bảo trì và cũng là khuynh hướng nhỏ gọn hơn bởi vì bạn có thể nhóm các mã chung vào trong một hàm thay việc đưa nó vào tất cả các nơi cần nó.

Cú pháp để tạo một hàm trongbash và pdksh là như sau:

```
fname () {
 shell commands
}
```

Cùng với cú pháp trước , pdksh cho phép cú pháp sau:

```
function fname {
 shell commands
}
```

Cả hai dạng này đều được xử lý chính xác như nhau theo cùng một cách.

Sau khi bạn đã định nghĩa hàm của bạn sử dụng một trong các dạng trên, bạn có thể gọi đến nó bằng cách vào lệnh sau:

```
fname [parm1 parm2 parm3 ...]
```

Chú ý rằng bạn có thể đưa số lượng bất kỳ các đối số vào trong hàm của bạn. Khi bạn đưa các đối số vào trong một hàm, nó xem các đối số này như đối số của một chương trình shell khi bạn đưa các đối số này từ dòng lệnh. Ví dụ, chương trình shell sau chứa vài hàm, mỗi hàm thực hiện một nhiệm vụ mà được kết hợp với các tùy chọn dòng lệnh. Ví dụ này bao trùm nhiều nội dung trong phần này. Nó đọc tất cả các file được đưa vào từ dòng lệnh và phụ thuộc vào tùy chọn được sử dụng, viết ra file với tất cả các ký tự hoa, viết ra file với tất cả các ký tự thường, hoặc in các file.

```
upper () {
 shift
 for i
 do
```

```
 tr a-z A-Z <$1 >$1.out
 rm $1
 mv $1.out $1
 shift
 done; }
lower () {
 shift
 for i
 do
 tr A-Z a-z <$1 >$1.out
 rm $1
 mv $1.out $1
 shift
 done; }
print () {
 shift
 for i
 do
 lpr $1
 shift
 done; }
usage_error () {
 echo "$1 syntax is $1 <option> <input files>"
 echo ""
 echo "where option is one of the following"
 echo "p -- to print frame files"
 echo "u -- to save as uppercase"
 echo "l -- to save as lowercase"; }
case $1
in
```

```
 p | -p) print $@;;
 u | -u) upper $@;;
 l | -l) lower $@;;
 *) usage_error $0;;
esac
```

Chương trình tcsh không hỗ trợ các hàm.

6.8. Tổng kết

Trong chương này, bạn đã thấy được nhiều đặc điểm của các ngôn ngữ lập trình bash, pdksh và tcsh. Khi bạn sử dụng Linux, bạn sẽ thấy rằng bạn sử dụng các ngôn ngữ lập trình shell càng ngày càng thường xuyên. Cho dù ngôn ngữ shell rất mạnh và dễ học, bạn có thể gặp phải một vài vấn đề khi chương trình shell không phù hợp với vấn đề bạn giải quyết. Trong những trường hợp như vậy, bạn có thể nghiên cứu tìm hiểu các ngôn ngữ khác có thể sử dụng có trong Linux.

7. Cài đặt và quản trị WebServer

Hướng dẫn cài đặt trên môi trường Linux.

Cài đặt trên môi trường Linux hoàn toàn không khó như những gì chúng ta nghĩ khi mới tiếp xúc với hệ điều hành này. Quá trình cài đặt chỉ đơn giản, chúng ta thực hiện câu lệnh rpm với cú pháp sau:

rpm –[ivhqladefUV] [-force] [nodeps] [--oldpackage] package list

Đây là chương trình quản lý các gói cài. Nó cho phép bạn quản lý các gói RPM, thực hiện rất dễ dàng việc cài đặt và gỡ bỏ phần mềm. Để cài đặt phần mềm có tên là precious-software-1.0.i386.rpm chạy câu lệnh sau:

rpm –i precious-software-1.0.i386.rpm

bạn có thể làm cho việc cài đặt trông đẹp mắt hơn bằng cách sử dụng tùy chọn –ivh thay cho tùy chọn –i. Nếu bạn đã cài một gói phần mềm rồi nhưng vì một lý do nào đó bạn lại muốn cài lại nó đè lên phiên bản cũ, bạn chỉ cần sử dụng tùy chọn –force cho lệnh rpm. Nếu bạn muốn nâng cấp một phần mềm, bạn sử dụng tùy chọn –U.Ví dụ:

rpm –Uvh precious-software-1.0.i386.rpm

Tuy nhiên bạn đã cài một phiên bản mới và bây giờ bạn muốn cài lại phiên bản cũ, nếu bạn muốn sử dụng lệnh trên, hệ thống sẽ báo lỗi phiên bản đã cài đặt là phiên bản mới hơn phiên bản mà bạn muốn cài. Để có thể thực hiện được điều này bạn sử dụng

tùy chọn --oldpackage cùng với tùy chọn –U để cài đặt phiên bản cũ. Để tìm kiếm các gói cài đã được cài vào hệ thống của bạn, bạn sử dụng lệnh sau:

rpm -qa

Để tìm các gói cài của một chương trình như sendmail, bạn có thể sử dụng lệnh

rpm –q sendmail

Hệ thống sẽ trả lại gói cài đã sử dụng để cài sendmail. Để phát hiện gói cài nào của một file xác định như /bin/tcsh, ta sử dụng câu lệnh:

rpm –qf /bin/tcsh

Để đảm bảo rằng một gói được cài chưa được thay đổi theo bất cứ cách nào, bạn có thể sử dụng tùy chọn –V. Ví dụ để tất cả các file đã được cài ở trạng thái nguyên bản không bị thay đổi sử dụng lệnh

rpm –Va

Tùy chọn này trở lên rất hữu ích nếu bạn nhận thức được rằng một hay nhiều gói cài có thể bị phá hủy bởi người khác.

Để gỡ các gói cài khỏi hệ thống bạn sử dụng lệnh rpm với tùy chọn –e

rpm –e sendmail

Nếu bạn thấy rằng việc gỡ bỏ gói cài có thể bị dừng bởi các chương trình khác bởi vì chúng phụ thuộc vào nó hay các file của nó, bạn phải quyết định xem bạn có tiếp tục bỏ gói cài hay chương trình này hay không, nếu bạn muốn gỡ bỏ bạn có thể sử dụng tùy chọn –nodeps cùng với tùy chọn –e để ép buộc rpm gỡ bỏ gói cài đó.

Quản trị WebServer

7.2.1. Phần mềm Apache

Máy chủ web nghe yêu cầu từ phía client, như bộ trình duyệt Nestcape Navigator hoặc Internet Explorer. Khi nhận được yêu cầu máy chủ xử lý yêu cầu và trả dữ liệu lại cho máy client. Dữ liệu trả về máy trạm thường là các trang định dạng có chứa hình ảnh và text. Trình duyệt nhận dữ liệu và hiển thị trang dữ liệu cho người dùng. Khái niệm máy chủ web rất đơn giản, nó đợi yêu cầu, thực hiện, rồi trả lại cho người dùng.

Máy chủ web nói chuyện với các máy client và máy trạm thông qua giao thức HTTP (Hypertext Transfer Protocol). Điều này cho phép máy trạm kết nối tới nhiều nhà cung cấp dịch vụ web mà không gặp phải các vấn đề về tương thích.

Phần lớn các yêu cầu được định dạng dưới dạng trang HTML (Hypertect Markup Language). HTML cho phép liên kết nhiều văn bản và tài nguyên khác nhau. Siêu văn bản cho phép liên kết tới các trang văn bản khác trên cùng một máy tính hoặc trên các máy tính đặt trên khắp thế giới.

Apache được phát triển dựa trên NCSA web server, là phiên bản cung cấp đầy đủ các tính năng của máy chủ (HTTP) web do dự án Apache Server thực hiện. Apache cung cấp một máy chủ web mã nguồn mở, tin cậy, hiệu quả và dễ dàng mở rộng. Phần mềm máy chủ bao gồm: daemon server, file cấu hình, công cụ quản trị, và tài liệu.

Phần mềm Apache Server sẵn có có trên trang Apache Group. Bạn có thể tải về từ các địa chỉ http://www.apache.org/dist/. Bạn tải về file .tar.gz tương ứng với phiên bản bạn muốn sử dụng. Ví dụ, Phiên bản mới nhất được viết là Apache 1.3.12, vì vậy file bạn cần tải về là apache_1.3.12.tar.gz Bạn có thể lấy mã nguồn từ địa chỉ http://www.apache.org/dist/apache_1.3.12.tar.gz.

Giải nén file

Để giải nén file này, sử dụng câu lệnh sau (giả sử rằng bạn đã để file trong thư mục temp):

```
cd temp
gzip -d -c apache_1.3.12.tar.gz | tar xvf -
```

Câu lệnh này tạo một thư mục apache_1.3.12 trong thư mục temp

7.2.2. Biên dịch và cài đặt

Chạy các câu lệnh sau:

```
cd apache_1.3.12
./configure --prefix=<path-to-apache>
make
make install
```

Chú ý sử dụng đường dẫn đầy đủ thay cho <path-to-apache>. Đường dẫn đầy đủ này nên là nơi bạn muốn cài đặt apache server, chẳng hạn như

```
./configure --prefix=/afs/uncc.edu/usr/q/zlian/apache
```

7.2.3. Khởi động và tắt WebServer

Khởi động Apache

```
<path-to-apache>/bin/apachectl start
```

Ví dụ:

```
/afs/uncc.edu/usr/q/zlian/Apache/bin/apachectl start
```

Tắt Apache

```
<path-to-apache>/bin/apachectl stop
```

Ví dụ:

```
/afs/uncc.edu/usr/q/zlian/Apache/bin/apachectl stop
```

7.2.4. Cấu hình Apache

Theo cách truyền thống, cấu hình Apache được chia thành ba file cấu hình: `httpd.conf`, `access.conf`, và `srm.conf`. Theo thứ tự các file này có ý nghĩa như sau, `httpd.conf` là file cấu hình server chính, `access.conf` là file định nghĩa các quyền truy cập, và `srm.conf` các tài nguyên server được định nghĩa, chẳng hạn như ánh xạ các thư mục và các biểu tượng. Trong 1.3.4, ba file này được trộn vào một file chung `httpd.conf`, nó có thể tìm thấy trong thư mục `conf`. Ví dụ:

```
/afs/uncc.edu/usr/q/zlian/apache/conf/
```

Chỉ dẫn quan trọng cho cấu hình của bạn:

ServerName

ServerName chỉ ra địa chỉ IP của máy chủ cài đặt dịch vụ WebServer, thông thường nếu máy của bạn là máy cục bộ, không nối mạng, địa chỉ này mặc định là 127.0.0.1 tương ứng với tên máy là localhost. Nếu máy này có địa chỉ mạng, bạn có thể thay thế bằng địa chỉ IP của máy. Để xem địa chỉ của máy bạn thực hiện lệnh:

```
ifconfig –a
```

Listen

Chỉ dẫn này nói cho server lắng nghe các yêu cầu trên dịa chỉ IP được xác định và/hoặc cổng TCP/IP . Mặc định, server lắng nghe cổng 80, nhưng bạn nên sử dụng cổng lớn hơn 1024, bởi vì số ít hơn 1024 rất hay được sử dụng trong các tiến trình của hệ thống. Như trong ví dụ sau, Apache nghe trên cả hai cổng port 8080 and 8081.

```
listen 8080
listen 8081
```

Với cấu hình này, bạn có thể kiểm tra xem server của bạn chạy thành công hay chưa bằng cách gõ vào địa chỉ sau trên trình duyệt:

http://localhost:8080

hoặc

http://localhost:8081

DocumentRoot

Thư mục tài liệu mặc định là <path-to-apache>/htdocs, bạn có thể để tài liệu html , ví dụ billchu.html, trong thư mục này và kiểm tra. Ví dụ:

```
http://152.15.35.2:8080/billchu.html
```

Bạn cũng có thể thay đổi thư mục tài liệu bằng sử dụng hướng dẫn sau trong file httpd.conf:

```
DocumentRoot /usr/web
```

Sau đó một truy cập đến `http://www.my.host.com/index.html` sẽ tương ứng `/usr/web/index.html`.

Thường xuất hiện trong khi cấu hình như sau: (i.e., "`DocumentRoot /usr/web/`") thêm một ký tự "/"ở đuôi, bạn nên tránh điều này.

7.2.5. Xc thực người đng

Để ngăn chặn truy cập vào các file trên server của bạn, bạn nên sử dụng bảo vệ user/password, Bạn có thể sử dụng các hướng dẫn sau.

```
AuthType
AuthName
AuthUserFile
AuthGroupFile
require
<Directory></Directory>
<Files></Files>
```

`AuthType` Lựa chọn kiểu xác thực người sử dụng cho một thư mục. Chỉ có `Basic` và `Digest` là thực thi hiện tại.

`AuthName` Đặt tên của xác thực cho một thư. Tên xác thực này sẽ được gửi đến client để những người sử dụng biết loại username và password nào để gửi. `AuthName` có một đối số; Nếu tên xác thực có dấu cách nó phải được đặt trong dấu trích dẫn.

`AuthUserFile` Đặt tên của file văn bản thuần túy chứa danh sách những người sử dụng và mật khẩu cho việc xác thực người sử dụng. Tên file là đường dẫn đến đến file người sử dụng. Nếu nó không phải là đường dẫn tuyệt đối (ví dụ, nếu nó không bắt đầu với '/'), Nó được xem như đường dẫn tương đối đến ServerRoot.

`AuthGroupFile` Đặt tên của một file văn bản thuần túy chứa danh sách các nhóm người sử dụng cho việc xác thực người sử dụng. Tên file là đường dẫn đến file group. Nếu nó không phải là đường dẫn tuyệt đối (ví dụ, không bắt đầu với dấu '/'), nó được xem như đường dẫn tương đối đến ServerRoot.

`require` Chọn những người sử dụng nào có thể truy cậo vào một thư mục. Cú pháp cho phép là:

1. Chỉ những người sử dụng được đặt tên có thể truy cập thư mục:

   ```
   require user userid userid ...
   ```

2. Chỉ những người sử dụng trong các nhóm được đặt tên có thể truy cập thư mục:

   ```
   require group group-name group-name ...
   ```

3. Tất cả những người sử dụng có thể truy cập thư mục:

   ```
   require valid-user
   ```

`<Directory>` và `</Directory>` được sử dụng để nhóm một nhóm các hướng dẫn và nó sẽ chỉ được áp dụng cho thư mục được đặt tên và các thư mục con

của thư mục đó. Một hướng dẫn bất kỳ được cho phép có trong một directory có thể được sử dụng.

<Files> và </Files> pcung cấp quyền truy cập bởi tên file (bao gồm đường dẫn đến file).

Ví dụ:

```
<Directory
"/afs/uncc.edu/usr/q/zlian/apache/htdocs/manual">
       AuthType Basic
       AuthName "Restricted Directory"
       AuthUserFile passwd
       AuthGroupFile /dev/null
       require valid-user
</Directory>
```

Để thiết lập file password, bạn có thể sử dụng công cụ có tên là htpasswd được cung cấp bởi Apache. Trước tiên tạo file password bằng cách:

```
% touch passwd
```

Trong thư mục "<path-to-apache>/bin/". Để thêm một người sử dụng, thực hiện lệnh:

```
% htpasswd <path-to-password-file>/passwd zlian
New password:
Re-type new password:
```

Đến đây bạn đã hoàn thành xong việc cấu hình Apache và thực hiện xác thực người sử dụng cho dịch vụ web của bạn.

8. Quản trị các tiến trình

8.1. Tiến Trình

8.1.1. Tiến trình tiền cảnh

Khi bạn đang trên dấu nhắc hệ thống (# hoặc $) và gọi một chương trình, chương trình trở thành một tiến trình và đi vào hoạt động dưới sự kiểm soát của hệ thống. Dấu nhắc của hệ thống sẽ không xuất hiện khi tiến trình đang chạy Khi tiến trình hoàn thành tác vụ và chấm dứt, hệ điều hành sẽ trả lại dấu nhắc để bạn gõ tiếp lệnh thực thi chương trình khác. Chương trình hoạt động theo cách này được gọi là chương trình tiền cảnh (foreground). Ví dụ khi bạn thực hiện lệnh:

ls –R /

Bạn sẽ phải chờ đợi rất lâu cho đến khi lệnh thực hiện xong bạn mới có thể nhập vào lệnh mới để thực hiện công việc tiếp theo của bạn.

8.1.2. Tiến trình hậu cảnh

Nếu có cách nào đó yêu cầu Linux đưa các tiến trình chiếm nhiều thời gian xử lí hoặc ít tương tác với người dùng ra hoạt động phía hậu cảnh (background) trả lại ngay dấu nhắc để có thể thực hiện các tiến trình ở tiền cảnh thì tốt hơn. Điều này có thể thực hiện được bằng cách kết hợp chỉ thị & với lệnh gọi chương trình mà ta sẽ tìm hiểu ở phần sau, khi đó tiến trình sẽ hoạt động ở phía hậu cảnh và trả lại ngay dấu nhắc cho chúng ta làm công việc khác. Các tiến trình như vậy gọi là các tiến trình hậu cảnh. Việc chạy tiến trình ở hậu cảnh rất thuận tiện , chúng cho phép nhiều chương trình tương tác với nhau.

8.2. Điều khiển và giám sát các tiến trình

Như để cập trước đây, các tiến trình thường trực thường được bắt đầu bằng tiến trình init khi khởi động. Bạn có thể điều khiển tiến trình nào chạy ngay khi khởi động bằng cách cấu hình lại các file cấu hình và kịch bản của init. Ngoại trừ các tiến trình thường trực, các loại tiến trình khác mà bạn sẽ chạy được gọi là các tiến trình của người sử dụng hay các tiến trình tương tác. Bạn phải chạy một tiến trình tương tác thông qua một shell. Mỗi một shell chuẩn cung cấp một dòng lệnh khi người sử dụng vào tên của một chương trình. Khi người sử dụng vào tên chương trình hợp lệ trên dòng lệnh, shell sẽ tự tạo một bản copy như một tiến trình mới và thay thế tiến trình mới với chương trình được đặt tên trên dòng lệnh. Nói một cách khác shell sẽ chạy chương trình được đặt tên như một tiến trình khác. Để lấy thông tin về tất cả các tiến trình đang chạy trên hệ thống của bạn, bạn cần chạy tiện ích có tên là ps

8.2.1 Sử dụng lệnh ps để lấy thông tin trạng thái của tiến trình

Tiện ích này tạo ra một báo cáo về tất cả các tiến trình trên hệ thống của bạn. ví dụ, nếu bạn chạy lệnh ps , nó sẽ hiển thị kết quả như sau:

```
  PID  TTY      TIME     CMD
13636  pts/1    00:00:00 bash
13696  pts/1    00:00:00 man
13699  pts/1    00:00:00 sh
13700  pts/1    00:00:00 sh
13704  pts/1    00:00:00 less
16692  pts/1    00:00:00 tail
17252  pts/1    00:00:00 ps
```

Dưới đây là giải thích về ý nghĩa của các trường

Trường	**Giải Thích**
USER hoặc UID	Tên của tiến trình

PID	ID (định danh) của tiến trình
%CPU	% CPU sử dụng của tiến trình
%MEM	% bộ nhớ tiến trình sử dụng
SIZE	Kích thước bộ nhớ ảo tiến trình sử dụng
RSS	Kích thước của bộ nhớ thực sử dụng bởi tiến trình
TTY	Vùng làm việc của tiến trình
STAT	Trạng thái của tiến trình
START	Thời gian hay ngày bắt đầu của tiến trình
TIME	Tổng thời gian sử dụng CPU
COMMAND	Câu lệnh được thực hiện
PRI	Mức ưu tiên của tiến trình
PPID	ID của tiến trình cha
WCHAN	Tên của hàm nhân khi tiến trình ngủ được lấy từ file /boot/System.map
FLAGS	Số cờ được kết hợp với tiến trình

Tiện ích ps cũng tiếp nhận một vài đối số từ dòng lệnh. Bảng bên dưới chỉ ra các tùy chọn được sử dụng chung:

Tùy Chọn	**Miêu tả**
A	Hiển thị các tiến trình của tất cả những người sử dụng
E	Hiển thị các biến môi trường của tiến trình sau khi dòng lệnh được thực thi
L	Hiển thị kết quả đầy đủ
U	Hiển thị tên người sử dụng và thời gian bắt đầu tiến trình
W	Hiển thị kết quả theo định dạng rộng. Bình thường, kết quả kết xuất bị cắt nếu nó không vừa một dòng. Sử dụng tùy chọn này bạn có thể ngăn chặn được điều đó
Txx	Hiển thị các tiến trình được kết hợp với vùng làm việc xx

X	Hiển thị các tiến trình không có điều khiển vùng làm việc

Ví dụ để hiển thị tất cả các tiến trình bạn thực hiện câu lệnh:

ps au

Để hiển thị tất cả các tiến trình của một người nào đó sử dụng:

ps au | grep username

Tuy nhiên, nếu bạn chỉ muốn tìm các tiến trình đang tồn tại với người sử dụng bất kỳ, bạn sử dụng câu lệnh:

ps aux

Để tìm kiếm PID của một tiến trình cha sử dụng:

ps l ***pid***

Với ***pid*** là PID của một tiến trình nào đó.

ps e

Thông tin biến môi trường được bổ sung vào trường COMMAND

8.2.2. Phát tín hiệu cho một chương trình đang chạy

Sử dụng lệnh kill hủy một tiến trình

Câu lệnh kill là một kịch bản shell được xây dựng sẵn, thường được tìm thấy trong thư mục /bin. Bạn có thể dùng lệnh này để dừng một tiến trình nào đó. bạn có thể chạy:

kill *PID*

Với *PID* là PID của tiến trình nào đó

Sử dụng lệnh killall hủy một tiến trình

Tiện ích này cho phép bạn dừng một tiến trình bằng tên. Ví dụ bạn có một tiến trình được goi là signal_demo.pl và bạn muốn dừng tiến trình này. Bạn sử dụng lệnh:

killall signal_demo.pl

Chạy một tiến trình ở hậu cảnh hoặc tiền cảnh

Thông thường khi chúng ta chạy một tiến trình từ thiết bị đầu cuối (bàn phím) hay shell, bạn chạy tiến trình ở tiền cảnh. Khi bạn chạy tiến trình ở tiền cảnh, bạn phải đợi cho nó kết thúc. Tuy nhiên, thay vì việc đợi cho nó kết thúc, bạn có thể chạy nó ở hậu cảnh bằng việc thêm một ký hiệu '&' ở cuối dòng lệnh. Điều này hữu ích khi một tiến trình chạy trong thời gian dài và bạn cần phải làm một công việc khác. Ví dụ, để khởi động hệ quản trị CSDL PosgresSQL với postmaster bạn thực hiện:

postmaster –i &

Vậy khi nào bạn biết một tiến trình hậu cảnh đang chạy hay đã dừng. Bạn có thể sử dụng lệnh:

ps -af

để xem tất cả các tiến trình trong đó có cả tiến trình ở hậu cảnh.

Tạm dừng tiến trình

Nếu một tiến trình đang chạy ở tiền cảnh và bạn muốn đưa chúng vào hậu cảnh, bạn thực hiện công việc này bằng cách nhấn tổ hợp phím Ctrl + Z. Khi nhận được tín hiệu Ctrl+Z tiến trình sẽ bị tạm dừng và được đưa vào hậu cảnh. Tuy nhiên bạn chưa biết được chương trình của chúng ta đã dừng chưa và đã chuyển vào hậu cảnh chưa. Lệnh jobs hiển thị trạng thái của tất cả các tiến trình đang chạy ở hậu cảnh:

[1] Stopped man ln (wd: /home/trantu/exam)

[2]- Stopped tail

[3]+ Stopped ls -R /

Đánh thức tiến trình

Để đánh thức một tiến trình ta sử dụng lệnh bg kết hợp với số tác vụ trong hàng đợi liệt kê. Trong ví dụ ở trên ta có thể thực hiện lệnh:

bg 3

Một lần nữa ta sử dụng lệnh jobs, ta sẽ thấy thông tin hiện trên màn hình như sau:

[1] Stopped man ln (wd: /home/trantu/exam)

[2]- Stopped tail

[3]+ Running ls -R /

Để chuyển một tiến trình từ hậu cảnh sang chạy trên tiền cảnh bạn dùng lệnh fg. Ví dụ:

fg 3

8.2.3. Giao tiếp giữa các tiến trình

Đôi khi các tiến trình cần trao đổi thông tin cho nhau để xử lý. Chẳng hạn như lệnh ls của Linux chỉ biết liệt kê và ghi toàn bộ dữ liệu về thông tin của file, thư mục ra màn hình. Lệnh ls không có cơ chế dừng khi màn hình đầy. Trong khi lệnh more lại có khả năng đọc dữ liệu và đưa ra màn hình theo từng trang để người dùng có thời gian xem qua. Các chương trình cần có nhu cầu chuyển dữ liệu cho nhau xử lý. Một cơ chế được sử dụng khá phổ biến trên Linux là pipe (đường ống). Bạn sử dụng chỉ thị | để biểu thị đường ống. Ví dụ:

ls –R | more

Hoặc bạn có thể tìm chính xác tên tiến trình như:

ps –af | grep '[bash]'

8.3 Lập kế hoạch các tiến trình

8.3.1 Sử dụng lệnh at

Tiện ích at cho phép bạn sắp xếp một câu lệnh để thực thi trong thời gian sau đó. Ví dụ, để xem dung lương đĩa sử dụng cho toàn bộ các file, thư muc của hệ thống bạn gọi tiện ích du vào lúc 8:40 p.m, bạn có thể chạy lệnh sau:

at 20:40

Câu lệnh sẽ hiển thị dấu nhắc "at>" yêu cầu bạn nhập vào câu lệnh để thực hiện theo thời gian đã được đưa vào. Bạn gõ vào dòng lệnh:

du –a > /tmp/du.out

Sau khi bạn gõ lệnh Enter, nó sẽ hiển thị lại dấu nhắc cho phép bạn nhập vào các câu lệnh tiếp theo. Bạn có thể chọn Ctrl+D để kết thúc.

Nếu vì một lý do nào đó, bạn muốn dừng cộng việc mà bạn đã lập lịch, bạn có thể sử dụng lệnh atrm để xóa công việc đó trước khi nó được thực hiện. Bạn cần phải biết số thứ tự của công việc mà bạn muốn hủy, để tìm ra các công việc mà bạn đã lập lịch, bạn chạy câu lệnh atq để tìm số thứ tự công việc, sau đó dùng atrq với đối số là số thứ tự của công việc muốn hủy. Ví dụ:

atrq 1

8.3.2 Sử dụng crontab

Có nhiều công việc trên Linux cần được lập lịch một cách thường xuyên, ví dụ để xóa các file cũ được sinh ra bởi hệ thống trong thư mục tmp hàng ngày, hay hàng tuần bạn cần phải chạy một tiến trình mỗi ngày hay mỗi tuần. Tiện ích cron cho phép bạn thực hiện các công việc như thế. Thực ra cron bao gồm crond daemon, được khởi động bởi tiến trình init. Crond đọc các lịch công việc từ /etc/crontab và các file trong /var/spoon/cron. Thư mục cron này lưu trữ các file lập lịch (thường được gọi là crontab hay cron table) cho những người sử dụng thông thường được phép chạy các công việc cron. Là một superuser, bạn có thể xác định một danh sách những người sử dụng được phép chạy các công việc cron trong file /etc/cron.allow. Tương tự, bạn có thể xác định những người sử dụng không được phép thực hiện các công việc cron trong file /etc/cron.deny. Cả hai file này đều sử dụng một định dạng cơ bản: một username trên một dòng. Nếu một người được phép thực hiện các công việc cron, người đó có thể sử dụng tiện ích crontab để thực hiện công việc lập lịch. Ví dụ, khi bạn được phép, bạn có thể gõ lệnh:

crontab –e

và soạn thảo các công việc cần thực hiện. Một công việc cron phải có định dạng sau:

minute(s) hour(s) day(s) month weekday username command argument(s)

Các trường từ 1 đến 5 có định dạng sau

9. Bảo mật hệ thống

Cùng với sự phát triển không ngừng của truyền thông kỹ thuật số, Internet và sự phát triển nhảy vọt của nền công nghiệp phần mềm, bảo mật máy tính là một vấn đề ngày càng trở nên quan trọng. Cần phải hiểu rằng không có hệ thống máy tính nào là an

toàn tuyệt đối. Tất cả những gì bạn có thể làm là giúp cho hệ thống của bạn trở nên an toàn hơn.

Kể từ khi Linux được phát triển một cách rộng rãi và nhanh chóng, đặc biệt là trong các giao dịch kinh doanh quan trọng, an ninh là một vấn đề quyết định sự sống còn của Linux. Với hàng trăm công cụ bảo vệ sẵn có, người dùng Linux được trang bị tốt hơn để ngăn chặn và duy trì một hệ thống an toàn. Linux không những hoạt động tốt mà còn có những tính năng và sản phẩm liên quan cho phép xây dựng một môi trường tương đối an toàn.

9.1. Những nguy cơ an ninh trên Linux

Linux và các ứng dụng trên nó có thể không ít các lỗ hổng an ninh hơn những hệ điều hành khác. Theo quan điểm của một số chuyên gia máy tính, Linux có tính an toàn cao hơn các hệ điều hành của Microsoft, vì các sản phẩm của Microsoft không được xem xét kỹ lưỡng và chặt chẽ bằng các sản phẩm mã nguồn mở như Linux. Hơn nữa, Linux dường như là "miễn nhiễm" với virus máy tính (hiện tại đã có xuất hiện một vài loại virus hoạt động trên môi trường Linux nhưng không ảnh hưởng gì mấy đến người dùng Linux). Nhưng một hệ thống Linux được cấu hình không tốt sẽ tệ hơn nhiều so với một hệ thống Microsoft được cấu hình tốt !!! Khi có được một chính sách an ninh tốt và hệ thống được cấu hình theo đúng chính sách đó thì sẽ giúp bạn tạo được một hệ thống an toàn (ở mức mà chính sách của bạn đưa ra).

Nhưng sự an toàn không phải là thứ có thể đạt được như một mục tiêu cuối cùng. Đúng hơn đó là tập hợp của những cách cài đặt, vận hành và bảo trì một hệ điều hành, mạng máy tính, ... Nó phụ thuộc vào các hoạt động hàng ngày của hệ thống, người dùng và người quản trị. Bạn phải bắt đầu từ một nền tảng ban đầu và từ đó cải thiện tính an toàn của hệ thống của bạn nhiều nhất có thể được mà vẫn đảm bảo các hoạt động bình thường của hệ thống.

9.2. Xem xét chính sách an ninh của bạn

Kết nối vào Inernet là nguy hiểm cho hệ thống mạng của bạn với mức an toàn thấp. Từ những vấn đề trong các dịch vụ TCP/IP truyền thống, tính phức tạp của việc cấu hình máy chủ, các lỗ hổng an ninh bên trong quá trình phát triển phần mềm và nhiều nhân tố khác góp phần làm cho những hệ thống máy chủ không được chuẩn bị chu đáo có thể bị xâm nhập và luôn tồn tại những nguy cơ tiềm tàng về vấn đề an toàn trong đó.

Mục đích của một chính sách an toàn hệ thống là quyết định một tổ chức sẽ phải làm như thế nào để bảo vệ chính nó. Để có được một chính sách an ninh hiệu quả, người xây dựng các chính sách này phải hiểu và có thể kết hợp tất cả các thông tin, yêu cầu, ...

Khi một tình huống xảy ra nằm ngoài dự kiến, chẳng hạn một sự xâm nhập trái phép vào hệ thống của bạn, câu hỏi lớn nhất là "sẽ phải làm gì đây ?"

Không may là có hàng triệu câu trả lời khác nhau cho câu hỏi đó. Nếu một người mà chưa từng phải đối phó với một kẻ xâm nhập trước đây thì kẻ xâm nhập có thể dễ dàng biến mất vì các dấu vết đã trở nên qúa cũ và không còn hữu ích nữa.

Những sai sót trong chính sách an ninh không chỉ liên quan đến những kẻ xâm nhập, mà còn liên quan đến những vấn đề bình thường như thời tiết, thiên tai, cháy, nổ, hư hỏng thiết bị,... Do vậy, việc thiết lập một chính sách an ninh tốt cho việc giải quyết những sự cố phải được lên kế hoạch kỹ lưỡng, được xem xét và chứng nhận bởi người có quyền hạn trong công ty.

Một chính sách an ninh tốt nên bao gồm các vấn đề sau :

- Chính sách phục hồi dữ liệu khi có sự cố
- Chính sách phục hồi hệ thống trong trường hợp hư hỏng thiết bị
- Chính sách, cách thức điều tra những kẻ xâm nhập trái phép
- Chính sách, cách thức điều tra khi công ty bị cáo buộc xâm nhập vào các hệ thống khác
- Cách thức, quy trình và nơi thông báo sự xâm nhập trái phép từ bên ngoài hay gây ra bởi các nhân viên của mình.
- Chính sách an ninh về mặt vật lý của hệ thống

  ...

Bạn có thể nhờ tư vấn của các công ty, tổ chức làm dịch vụ tư vấn về an toàn máy tính để giúp bạn xây dựng một chính sách an ninh tốt. Các công ty này có các chuyên gia về an toàn máy tính, họ có sẵn các biểu mẫu chính sách an ninh nên có thể thiết lập nhanh chóng các chính sách mà bao gồm tất cả các mặt trong việc an toàn hệ thống máy tính.

9.3. Tăng cường an ninh cho KERNEL

Mặc dù thừa hưởng những đặc tính của hệ thống UNIX và khá an ninh hơn một số hệ điều hành khác, hệ thống GNU/Linux hiện nay vẫn tồn tại những nhược điểm sau:

- Quyền của user ‘root’ có thể bị lạm dụng. User ‘root’ có thể dễ dàng thay đổi bất kỳ điều gì trên hệ thống.
- Nhiều file hệ thống có thể dễ dàng bị sửa đổi. Nhiều file hệ thống quan trọng như /bin/login có thể bị sửa đổi bởi hacker để cho phép đăng nhập không cần mật khẩu. Nhưng những file loại này lại hiếm khi nào thay đổi trừ phi khi nâng cấp hệ thống.
- Các module có thể được dùng để chặn kernel. “Loadable Kernel Module” là một thiết kế tốt để tăng cường tính uyển chuyển, linh hoạt cho kernel. Nhưng sau khi một module được nạp vào kernel, nó sẽ trở thành một phần của kernel và có thể hoạt động như kernel nguyên thủy. Vì vậy, các chương trình mục đích xấu có thể được viết dạng module và nạp vào kernel, rồi sau đó hoạt động như một virus.
- Các process không được bảo vệ. Các process như web server có thể trở thành mục tiêu bị tấn công của hacker sau khi thâm nhập hệ thống.

Để cải thiện tính an ninh cho các server Linux, chúng ta cần có một kernel an toàn hơn. Điều này có thể thực hiện được bằng cách sửa đổi kernel nguyên thuỷ bằng các ‘patch’ tăng cường tính an ninh cho hệ thống. Các patch này có các tính năng chính yếu sau:

- Bảo vệ – bảo vệ các file hệ thống quan trọng khỏi sự thay đổi ngay cả với user root. Bảo vệ các process quan trọng khỏi bị ngừng bởi lệnh 'kill'. Chặn các tác vụ truy cập IO mức thấp (RAW IO) của các chương trình không được phép.
- Phát hiện – Phát hiện và cảnh báo với người quản trị khi server bị scan. Cũng như khi có các tác vụ trên hệ thống vi phạm các luật (rules) định trước.
- Đối phó – Khi phát hiện sự vi phạm trên hệ thống, các ghi nhận chi tiết sẽ được thực hiện cũng như có thể ngừng lập tức phiên làm việc gây ra

Một vài công cụ sửa đổi kernel được sử dụng rộng rãi là LIDS (Linux Intrusion Detection System), Medusa, ...

9.4. An toàn các giao dịch trên mạng

Có rất nhiều dịch vụ mạng truyền thống giao tiếp thông qua giao thức văn bản không mã hoá, như TELNET, FTP, RLOGIN, HTTP, POP3. Trong các giao dịch giữa người dùng với máy chủ, tất cả các thông tin dạng gói được truyền qua mạng dưới hình thức văn bản không được mã hoá. Các gói tin này có thể dễ dàng bị chặn và sao chép ở một điểm nào đó trên đường đi. Việc giải mã các gói tin này rất dễ dàng, cho phép lấy được các thông tin như tên người dùng, mật khẩu và các thông tin quan trọng khác. Việc sử dụng các giao dịch mạng được mã hoá khiến cho việc giải mã thông tin trở nên khó hơn và giúp bạn giữ an toàn các thông tin quan trọng. Các kỹ thuật thông dụng hiện nay là IPSec, SSL, TLS, SASL và PKI.

Quản trị từ xa là một tính năng hấp dẫn của các hệ thống UNIX. Người quản trị mạng có thể dễ dàng truy nhập vào hệ thống từ bất kỳ nơi nào trên mạng thông qua các giao thức thông dụng như telnet, rlogin. Một số công cụ quản trị từ xa được sử dụng rộng rãi như linuxconf, webmin cũng dùng giao thức không mã hoá. Việc thay thế tất cả các dịch vụ mạng dùng giao thức không mã hoá bằng giao thức có mã hoá là rất khó. Tuy nhiên, bạn nên cung cấp việc truy cập các dịch vụ truyền thống như HTTP/POP3 thông qua SSL, cũng như thay thế các dịch vụ telnet, rlogin bằng SSH.

9.5. Linux firewall

An toàn hệ thống luôn luôn là một vấn đề sống còn của mạng máy tính và firewall là một thành phần cốt yếu cho việc đảm bảo an ninh.

Một firewall là một tập hợp các qui tắc, ứng dụng và chính sách đảm bảo cho người dùng truy cập các dịch vụ mạng trong khi mạng bên trong vẫn an toàn đối với các kẻ tấn công từ Internet hay từ các mạng khác. Có hai loại kiến trúc firewall cơ bản là : Proxy/Application firewall và filtering gateway firewall. Hầu hết các hệ thống firewall hiện đại là loại lai (hybrid) của cả hai loại trên.

Nhiều công ty và nhà cung cấp dịch vụ Internet sử dụng máy chủ Linux như một Internet gateway. Những máy chủ này thường phục vụ như máy chủ mail, web, ftp, hay dialup. Hơn nữa, chúng cũng thường hoạt động như các firewall, thi hành các chính sách kiểm soát giữa Internet và mạng của công ty. Khả năng uyển chuyển khiến cho Linux thu hút như là một thay thế cho những hệ điều hành thương mại.

Tính năng firewall chuẩn được cung cấp sẵn trong kernel của Linux được xây dựng từ hai thành phần : ipchains và IP Masquerading.

Linux IP Firewalling Chains là một cơ chế lọc gói tin IP. Những tính năng của IP Chains cho phép cấu hình máy chủ Linux như một filtering gateway/firewall dễ dàng. Một thành phần quan trọng khác của nó trong kernel là IP Masquerading, một tính năng chuyển đổi địa chỉ mạng (network address translation- NAT) mà có thể che giấu các địa chỉ IP thực của mạng bên trong.

Để sử dụng ipchains, bạn cần thiết lập một tập các luật mà qui định các kết nối được cho phép hay bị cấm. Ví dụ:

```
# Cho phép các kết nối web tới Web Server của bạn
/sbin/ipchains -A your_chains_rules -s 0.0.0.0/0 www -d 192.16.0.100
1024: -j ACCEPT

# Cho phép các kết nối từ bên trong tới các Web Server bên ngoài
/sbin/ipchains -A your_chains_rules -s 192.168.0.0/24 1024: -d
0.0.0.0/0 www -j ACCEPT

# Từ chối truy cập tất cả các dịch vu khác
/sbin/ipchains -P your_chains_rules input DENY
```

Ngoài ra, bạn có thể dùng các sản phẩm firewall thương mại như Check Point FireWall-1, Phoenix Adaptive Firewall, Gateway Guardian, XSentry Firewall, Raptor, ... hay rất nhiều các phiên bản miễn phí, mã nguồn mở cho Linux như T.Rex Firewall, Dante, SINUS, TIS Firewall Toolkit, ...

9.6. Dùng công cụ dò tìm để khảo sát hệ thống

Thâm nhập vào một hệ thống bất kỳ nào cũng cần có sự chuẩn bị. Hacker phải xác định ra máy đích và tìm xem những port nào đang mở trước khi hệ thống có thể bị xâm phạm. Quá trình này thường được thực hiện bởi các công cụ dò tìm (scanning tool), kỹ thuật chính để tìm ra máy đích và các port đang mở trên đó. Dò tìm là bước đầu tiên hacker sẽ sử dụng trước khi thực hiện tấn công. Bằng cách sử dụng các công cụ dò tìm như Nmap, hacker có thể rà khắp các mạng để tìm ra các máy đích có thể bị tấn công. Một khi xác định được các máy này, kẻ xâm nhập có thể dò tìm các port đang lắng nghe. Nmap cũng sử dụng một số kỹ thuật cho phép xác định khá chính xác loại máy đang kiểm tra.

Bằng cách sử dụng những công cụ của chính các hacker thường dùng, người qun trị hệ thống có thể nhìn vào hệ thống của mình từ góc độ của các hacker và giúp tăng cường tính an toàn của hệ thống. Có rất nhiều công cụ dò tìm có thể sử dụng như: Nmap, strobe, sscan, SATAN, ...

Dưới đây là một ví dụ sử dụng Nmap:

```
# nmap -sS -O 192.168.1.200
Starting nmap V. 2.54 by Fyodor (fyodor@dhp.com,
www.insecure.org/nmap/)
Interesting ports on comet (192.168.1.200):
Port State Protocol Service
```

```
7 open tcp echo
19 open tcp chargen
21 open tcp ftp
...
TCP Sequence Prediction: Class=random positive increments
Difficulty=17818 (Worthy challenge)
Remote operating system guess: Linux 2.2.13
Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

Tuy nhiên, sử dụng các công cụ này không thể thay thế cho một người quản trị có kiến thức. Bởi vì việc dò tìm thường dự báo một cuộc tấn công, các site nên ưu tiên cho việc theo dõi chúng. Với các công cụ dò tìm, các nhà quản trị hệ thống mạng có thể phát hiện ra những gì mà các hacker có thể thấy khi dò trên hệ thống của mình.

9.7. Phát hiện sự xâm nhập qua mạng

Nếu hệ thống của bạn có kết nối vào internet, bạn có thể trở thành một mục tiêu bị dò tìm các lỗ hổng về bảo mật. Mặc dù hệ thống của bạn có ghi nhận điều này hay không thì vẫn không đủ để xác định và phát hiện việc dò tìm này. Một vấn đề cần quan tâm khác là các cuộc tấn công gây ngừng dịch vụ (Denial of Services - DoS), làm thế nào để ngăn ngừa, phát hiện và đối phó với chúng nếu bạn không muốn hệ thống của bạn ngưng trệ.

Hệ thống phát hiện xâm nhập qua mạng (Network Intrusion Detection System - NIDS) theo dõi các thông tin truyền trên mạng và phát hiện nếu có hacker đang cố xâm nhập vào hệ thống (hoặc gây gây ra một vụ tấn công DoS). Một ví dụ điển hình là hệ thống theo dõi số lượng lớn các yêu cầu kết nối TCP đến nhiều port trên một máy nào đó, do vậy có thể phát hiện ra nếu có ai đó đang thử một tác vụ dò tìm TCP port. Một NIDS có thể chạy trên máy cần theo dõi hoặc trên một máy độc lập theo dõi toàn bộ thông tin trên mạng.

Các công cụ có thể được kết hợp để tạo một hệ thống phát hiện xâm nhập qua mạng. Chẳng hạn dùng tcpwrapper để điều khiển, ghi nhận các dịch vụ đã được đăng ký. Các chương trình phân tích nhật ký hệ thống, như swatch, có thể dùng để xác định các tác vụ dò tìm trên hệ thống. Và điều quan trọng nhất là các công cụ có thể phân tích các thông tin trên mạng để phát hiện các tấn công DoS hoặc đánh cắp thông tin như tcpdump, ethereal, ngrep, NFR (Network Flight Recorder), PortSentry, Sentinel, Snort, ...

Khi hiện thực một hệ thống phát hiện xâm nhập qua mạng bạn cần phải lưu tâm đến hiệu suất của hệ thống cũng như các chính sách bảo đảm sự riêng tư.

9.8. Kiểm tra khả năng bị xâm nhập

Kiểm tra khả năng bị xâm nhập liên quan đến việc xác định và sắp xếp các lỗ hổng an ninh trong hệ thống bằng cách dùng một số công cụ kiểm tra. Nhiều công cụ kiểm tra cũng có khả năng khai thác một số lỗ hổng tìm thấy để làm rõ quá trình thâm nhập trái phép sẽ được thực hiện như thế nào. Ví dụ, một lỗi tràn bộ đệm của chương trình phục vụ dịch vụ FTP có thể dẫn đến việc thâm nhập vào hệ thống với quyền ‘root’. Nếu người quản trị mạng có kiến thức về kiểm tra khả năng bị xâm nhập trước khi nó xảy ra, họ có thể tiến hành các tác vụ để nâng cao mức độ an ninh của hệ thống mạng.

Có rất nhiều các công cụ mạng mà bạn có thể sử dụng trong việc kiểm tra khả năng bị xâm nhập. Hầu hết các quá trình kiểm tra đều dùng ít nhất một công cụ tự động phân tích các lỗ hổng an ninh. Các công cụ này thăm dò hệ thống để xác định các dịch vụ hiện có. Thông tin lấy từ các dịch vụ này sẽ được so sánh với cơ sở dữ liệu các lỗ hổng an ninh đã được tìm thấy trước đó.

Các công cụ thường được sử dụng để thực hiện các kiểm tra loại này là ISS Scanner, Cybercop, Retina, Nessus, cgiscan, CIS, ...

Kiểm tra khả năng bị xâm nhập cần được thực hiện bởi những người có trách nhiệm một cách cẩn thận. Sự thiếu kiến thức và sử dụng sai cách có thể sẽ dẫn đến hậu quả nghiêm trọng không thể lường trước được.

9.9. Đối phó khi hệ thống bị tấn công

Gần đây, một loạt các vụ tấn công nhắm vào các site của những công ty lớn như Yahoo!, Buy.com, E-Bay, Amazon và CNN Interactive gây ra những thiệt hại vô cùng nghiêm trọng. Những tấn công này là dạng tấn công gây ngừng dịch vụ "Denial-Of-Service" mà được thiết kế để làm ngưng hoạt động của một mạng máy tính hay một website bằng cách gửi liên tục với số lượng lớn các dữ liệu tới mục tiêu tấn công khiến cho hệ thống bị tấn công bị ngừng hoạt động, điều này tương tự như hàng trăm người cùng gọi không ngừng tới 1 số điện thoại khiến nó liên tục bị bận.

Trong khi không thể nào tránh được mọi nguy hiểm từ các cuộc tấn công, chúng tôi khuyên bạn một số bước mà bạn nên theo khi bạn phát hiện ra rằng hệ thống của bạn bị tấn công. Chúng tôi cũng đưa ra một số cách để giúp bạn bảo đảm tính hiệu qủa của hệ thống an ninh và những bước bạn nên làm để giảm rủi ro và có thể đối phó với những cuộc tấn công.

Nếu phát hiện ra rằng hệ thống của bạn đang bị tấn công, hãy bình tĩnh. Sau đây là những bước bạn nên làm:

- Tập hợp 1 nhóm để đối phó với sự tấn công:

  - Nhóm này phải bao gồm những nhân viên kinh nghiệm, những người mà có thể giúp hình thành một kế hoạch hành động đối phó với sự tấn công.

- Dựa theo chính sách và các quy trình thực hiện về an ninh của công ty, sử dụng các bước thích hợp khi thông báo cho mọi người hay tổ chức về cuộc tấn công.
- Tìm sự giúp đỡ từ nhà cung cấp dịch vụ Internet và cơ quan phụ trách về an ninh máy tính:

  - Liên hệ nhà cung cấp dịch vụ Internet của bạn để thông báo về cuộc tấn công. Có thể nhà cung cấp dịch vụ Internet của bạn sẽ chặn đứng được cuộc tấn công.

  - Liên hệ cơ quan phụ trách về an ninh máy tính để thông báo về cuộc tấn công

- Tạm thời dùng phương thức truyền thông khác (chẳng hạn như qua điện thoại) khi trao đổi thông tin để đảm bo rằng kẻ xâm nhập không thể chặn và lấy được thông tin.
- Ghi lại tất cả các hoạt động của bạn (chẳng hạn như gọi điện thoại, thay đổi file, ...)
- Theo dõi các hệ thống quan trọng trong qúa trình bị tấn công bằng các phần mềm hay dịch vụ phát hiện sự xâm nhập (intrusion detection software/services). Điều này có thể giúp làm giảm nhẹ sự tấn công cũng như phát hiện những dấu hiệu của sự tấn công thực sự hay chỉ là sự quấy rối nhằm đánh lạc hướng sự chú ý của bạn(chẳng hạn một tấn công DoS với dụng ý làm sao lãng sự chú ý của bạn trong khi thực sự đây là một cuộc tấn công nhằm xâm nhập vào hệ thống của bạn).

  - Sao chép lại tất cả các files mà kẻ xâm nhập để lại hay thay đổi (như những đoạn mã chương trình, log file, ...)

- Liên hệ nhà chức trách để báo cáo về vụ tấn công.

Những bước bạn nên làm để giảm rủi ro và đối phó với sự tấn công trong tương lai :

- Xây dựng và trao quyền cho nhóm đối phó với sự tấn công
- Thi hành kiểm tra an ninh và đánh giá mức độ rủi ro của hệ thống
- Cài đặt các phần mềm an toàn hệ thống phù hợp để giảm bớt rủi ro
- Nâng cao khả năng của mình về an toàn máy tính

Các bước kiểm tra để giúp bạn bảo đảm tính hiệu quả của hệ thống an ninh

- Kiểm tra hệ thống an ninh mới cài đặt : chắc chắn tính đúng đắn của chính sách an ninh hiện có và cấu hình chuẩn của hệ thống.
- Kiểm tra tự động thường xuyên : để khám phá sự "viếng thăm" của những hacker hay những hành động sai trái của nhân viên trong công ty.
- Kiểm tra ngẫu nhiên: để kiểm tra chính sách an ninh và những tiêu chuẩn, hoặc kiểm tra sự hiện hữu của những lỗ hổng đã được phát hiện (chẳng hạn những lỗi được thông báo từ nhà cung cấp phần mềm)
- Kiểm tra hằng đêm những file quan trọng: để đánh giá sự toàn vẹn của những file và cơ sở dữ liệu quan trọng
- Kiểm tra các tài khoản người dùng: để phát hiện các tài khoản không sử dụng, không tồn tại, ...
- Kiểm tra định kỳ để xác định trạng thái hiện tại của hệ thống an ninh của bạn

BẠN CÓ THỂ XEM THÊM THÔNG TIN TẠI

Các trung tâm giúp đối phó tai nạn trên Internet

http://www.cert.org

- http://www.first.org
- http://ciac.llnl.gov/
- http://www.cert.dfn.de/eng/csir/europe/certs.html

Một số website về an toàn máy tính

- http://www.cs.purdue.edu/coast/
- http://www.linuxsecurity.com
- http://www.securityportal.com
- http://www.tno.nl/instit/fel/intern/wkinfsec.html
- http://www.icsa.net
- http://www.sans.org
- http://www.iss.com
- http://www.securityfocus.com

Thông tin về an toàn từ nhà cung cấp

- http://www.calderasystems.com/news/security/
- http://www.debian.org/security/
- http://www.redhat.com/cgi-bin/support/

Một số sách về an toàn máy tính

- Actually Useful Internet Security Techniques by Larry J. Hughes Jr.
- Applied Cryptography: Protocols, Algorithms and Source Code in C by Bruce Schneier
- Building Internet Firewall by Brent Chapman & Elizabeth D. Zwicky
- Cisco IOS Network Security by Mike Kaeo
- Firewalls and Internet Security by Bill Cheswick & Steve Bellovin
- Halting the Hacker: A practical Guide To Computer Security by Donal L. Pipkin
- Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Traps, Trace Back and Response by Edward G. Amoroso
- Intrusion Detection: Network Security Beyond the Firewall by Terry Escamilla
- Linux Security by Jonh S. Flowers

Giáo trình

Hệ điều hành mạng Linux

MỤC LỤC

BẢNG TỪ VIẾT TẮT

Hệ điều hành	HĐH
Multiplexed Information and Computing Service	Multics
Berkley Software Distribution	BSD
Midnight Commander	mc

CHƯƠNG 1: TỔNG QUAN VỀ UNIX/ LINUX

1. Lịch sử phát triển của Unix

Giữa năm 1960, AT&T Bell Laboratories và một số trung tâm khác tham gia vào một cố gắng tạo ra một hệ điều hành mới được đặt tên là Multics.

Đến năm 1969, chương trình Multics bị bãi bỏ vì đó là một dự án quá nhiều tham vọng. Thậm trí nhiều yêu cầu đối với Multics thời đó đến nay vẫn chưa có được trên các Unix mới nhất. ả hưng Ken Thompson, Dennis Ritchie, và một số đồng nghiệp của Bell Labs đã không bỏ cuộc.

Ken Thompson và Dennis Ritchie (Bell Lab thập niên 70)

Thay vì xây dựng một HĐH làm nhiều việc một lúc, họ quyết định phát triển một HĐH đơn giản chỉ làm tốt một việc là chạy chương trình (run program). HĐH sẽ có rất nhiều các công cụ (tool) nhỏ, đơn giản, gọn nhẹ (compact) và chỉ làm tốt một công việc. Bằng cách kết hợp nhiều công cụ lại với nhau, họ sẽ có một chương trình thực hiện một công việc phức tạp. Đó cũng là cách thức người lập trình viết ra chương trình. Peter ả eumann đặt tên Unix cho HĐH đơn giản này tiếp tục phát triển theo mô hình ban đầu và đặt ra một hệ thống tập tin mà sau này được phát triển thành hệ thống tập tin của Uả IX.

ả ăm 1973, Riche và Thompson viết lại nhân của hệ điều hành Uả IX trên ngôn ngữ C, và hệ điều hành đã trở nên dễ dàng cài đặt tới các loại máy tính khác nhau; tính chất như thế được gọi là tính khả chuyển của Uả IX. Trước đó, khoảng

năm 1971, hệ điều hành được thể hiện trên ngôn ngữ B (mà dựa trên ngôn ngữ B, Ritche đã phát triển thành ngôn ngữ C).

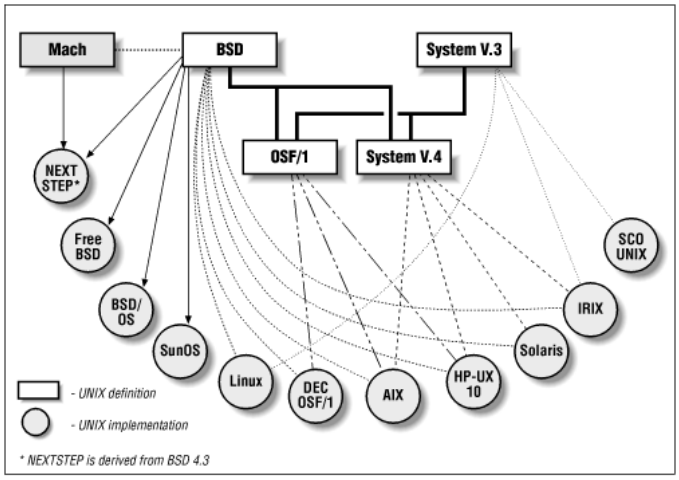
Khoảng năm 1977 bản quyền của Unix được giải phóng và HĐH Unix trở thành một sản phẩm thương mại.

Hai dòng Uả IX: System V của AT&T, ả ovell và Berkeley Software Distribution (BSD) của Đại học Berkeley.

+ ***System V:*** Các phiên bản Uả IX cuối cùng do AT&T xuất bản là System III và một vài phát hành (releases) của System V. Hai bản phát hành gần đây của System V là Release 3 (SVR3.2) và Release 4.2 (SVR4.2). Phiên bản SYR 4.2 là phổ biến nhất cho từ máy PC cho tới máy tính lớn.
+ ***BSD:*** Từ 1970 Computer Science Research Group của University of California tại Berkeley (UCB) xuất bản nhiều phiên bản Uả IX, được biết đến dưới tên Berkeley Software Distribution, hay BSD. Cải biến của PDP-11 được gọi là 1BSD và 2BSD. Trợ giúp cho các máy tính của Digital Equipment Corporation VAX được đưa vào trong 3BSD. Phát triển của VAX được tiếp tục với 4.0BSD, 4.1BSD, 4.2BSD, và 4.3BSD

Trước 1992, Uả IX là tên thuộc sở hữu của AT&T. Từ năm 1992, khi AT&T bán bộ phận Unix cho ả ovell, tên Unix thuộc sở hữu của X/Open foundation. Tất cả các hệ điều hành thỏa mãn một số yêu cầu đều có thể gọi là Unix. ả goài ra, Institute of Electrical and Electronic Engineers (IEEE) đã thiết lập chuẩn "An Industry-Recognized Operating Systems Interface Standard based on the Uả IX Operating System." Kết quả cho ra đời POSIX.1 (cho giao diện C) và POSIX.2 (cho hệ thống lệnh trên Unix)

Tóm lại, vấn đề chuẩn hóa Uả IX vẫn còn rất xa kết quả cuối cùng. ả hưng đây là quá trình cần thiết có lợi cho sự phát triển của ngành tin học nói chung và sự sống còn của HĐH Uả IX nói riêng.



Hình 1.1 Các phiên bản của Unix

Các nhóm nhà cung cấp khác nhau về Uả IX đang hoạt động trong thời gian hiện nay được kể đến như sau:

+ Unix International (viết tắt là UI). UI là một tổ chức gồm các nhà cung cấp thực hiện việc chuyển nhượng hệ thống Uả IX-5 và cung cấp bản AT&T theo các nhu cầu và thông báo phát hành mới, chẳng hạn như điều chỉnh bản quyền. Giao diện đồ họa người dùng là Open Look.
+ Open Software Foundation (OSF). OSF được hỗ trợ bởi IBM, DEC, HP ... theo hướng phát triển một phiên bản của Unix nhằm tranh đua với hệ thống Uả IX-5 phiên bản 4. Phiên bản này có tên là OSF/1 với giao diện đồ họa người dùng được gọi là MOTIF.
+ Free SoftWare Foundation (FSF): một cộng đồng do Richard Stallman khởi xướng năm 1984 chủ trương phát hành các phần mềm sử dụng tự do, trên cơ sở một hệ điều hành thuộc loại Uả IX.

Formatted: Bullets and Numbering

2. Lịch sử phát triển của Linux

Linux là một HĐH dạng Uả IX (Unix-like Operating System) chạy trên máy PC với bộ điều khiển trung tâm (CPU) Intel 80386 hoặc các thế hệ sau đó, hay các bộ vi xử lý trung tâm tương thích như AMD, Cyrix. Linux ngày nay còn có thể chạy trên các máy Macintosh hoặc SUả Sparc. Linux thỏa mãn chuẩn POSIX.1.

Linux được viết lại toàn bộ từ con số không, tức là không sử dụng một dòng lệnh nào của Unix, để tránh vấn đề bản quyền của Unix, tuy nhiên hoạt động của

Linux hoàn toàn dựa trên nguyên tắc của hệ điều hành Unix. Vì vậy nếu một người nắm được Linux, thì sẽ nắm được Uẩ IX. ẩ ên chú ý rằng *giữa các Unix sự khác nhau cũng không kém gì giữa Unix và Linux.*

ẩ ăm 1991 Linus Torvalds - sinh viên của đại học tổng hợp Helsinki, Phần lan, bắt đầu xem xét Minix, một phiên bản của Unix, làm ra với mục đích nghiên cứu cách tạo ra một hệ điều hành Unix chạy trên máy PC với bộ vi xử lý Intel 80386.

ẩ gày 25/8/1991, Linus cho ra version 0.01 và thông báo trên comp.os.minix của Internet về chương trình của mình.

1/1992, Linus cho ra version 0.12 với shell và C compiler, Linus đặt tên HĐH của mình là Linux. 1994, phiên bản chính thức 1.0 được phát hành.

Quá trình phát triển của Linux được tăng tốc bởi sự giúp đỡ của chương trình Gẩ U, đó là chương trình phát triển các Unix có khả năng chạy trên nhiều platform. Đến hôm nay, cuối 2001, phiên bản mới nhất của Linux kernel là 2.4.2-2, có khả năng điều khiển các máy đa bộ vi xử lý và rất nhiều các tính năng khác.

Hiện nay, Linux là một hệ điều hành giống Unix đầy đủ và độc lập. ẩ ó có thể chạy X-Window, TCP/IP, Emacs, Web, thư điện tử và các phần mềm khác. Hầu hết các phần mềm miễn phí và thương mại đều được chuyển lên Linux. Rất nhiều các nhà phát triển phần mềm bắt đầu chuyển sang viết trên Linux. ẩ gười ta thực hiện các phép đo benchmarks trên Linux và thấy rằng chúng thực hiện nhanh hơn khi thực hiện trên các máy trạm của Sun Microystem và Compaq, thậm chí nhiều khi còn nhanh hơn cả trên Windows 98 và Windowẩ T. Thật khó có thể hình dung được hệ điều hành "Unix" tí hon này phát triển nhanh thế nào!

Bây giờ, sau khi đã trải qua 1 thời gian rất dài phát triển và hoàn thiện bởi cộng đồng thế giới, Linux càng ngày càng trở nên mạnh mẽ, ổn định và độ tin cậy cao, và được chọn để sử dụng trong các cơ quan chính phủ. Các nước như Trung Quốc, ẩ hật Bản, Đức và một số các nước châu Âu đều cũng đã có kế hoạch phát triển riêng Linux cho đất nước của họ. Ở Việt ẩ am trong những năm gần đây đã có nhiều nhóm nghiên cứu và phát triển Linux sử dụng tiếng Việt là ngôn ngữ chính.

Trong giáo trình này Linux được sử dụng như một ví dụ cho việc tìm hiểu kỹ hơn về hệ điều hành Unix.

Vấn đề phân phối và giấy phép Linux

Về lý thuyết, mọi người có thể khởi tạo một hệ thống Linux bằng cách tiếp cận bản mới nhất các thành phần cần thiết từ các site ftp và biên dịch chúng. Trong thời kỳ đầu tiên, người dùng Linux phải tiến hành toàn bộ các thao tác này và vì vậy công việc là khá vất vả. Tuy nhiên, do có sự tham gia đông đảo của các cá nhân và nhóm phát triển Linux, đã tiến hành thực hiện nhiều giải pháp nhằm làm cho công việc khởi tạo hệ thống đỡ vất vả. Một trong những giải pháp điển hình nhất là cung cấp tập các gói chương trình đã tiền dịch, chuẩn hóa.

ả hững tập hợp như vậy hay những bản phân phối là lớn hơn nhiều so với hệ thống Linux cơ sở. Chúng thường bao gồm các tiện ích bổ sung cho khởi tạo hệ thống, các thư viện quản lý, cũng như nhiều gói đã được tiền dịch, sẵn sàng khởi tạo của nhiều bộ công cụ Uả IX dùng chung, chẳng hạn như phục vụ tin, trình duyệt web, công cụ xử lý, soạn thảo văn bản và thậm chí các trò chơi.

Cách thức phân phối ban đầu rất đơn giản song ngày càng được nâng cấp và hoàn thiện bằng phương tiện quản lý gói tiên tiến. Các bản phân phối ngày nay bao gồm các cơ sở dữ liệu tiến hóa gói, cho phép các gói dễ dàng được khởi tạo, nâng cấp và loại bỏ.

ả hà phân phối đầu tiên thực hiện theo phương châm này là Slakware, và chính họ là những chuyển biến mạnh mẽ trong cộng đồng Linux đối với công việc quản lý gói khởi tạo Linux.

Tiện ích quản lý gói RPM (RedHat Package Manager) của công ty RedHat là một trong những phương tiện điển hình.

ả hân Linux là phần mềm tự do được phân phối theo Giấy phép sở hữu công cộng phần mềm Gả U GPL.

2.1 Một số đặc điểm chính của Linux

Đa nền

Linux ban đầu được xem là bản sao của Unix và vận hành trên các máy tính cá nhân được trang bị bộ xử lý 386, 486 hoặc các bộ xử lý cấp cao hơn. Mặc dù ban đầu nó được phát triển cho các cấu trúc x86 nhưng hiện nay nó có thể vận hành trên các nền khác nhau như Alpha, Sparc, Dec, Sun, Power PC và một số nền 68000 như

Atari, Amiga...ả goài ra Linux còn chạy trên một số máy MIPS và các máy tính cá nhân mạnh.

Đa chương trình

Một thời điểm một người sử dụng có thể thực hiện đồng thời nhiều tác vụ. Với hệ điều hành đơn chương trình như MS-DOS một lệnh thực hiện sẽ chiếm toàn bộ thời gian CPU xử lý, ta chỉ có thể thực hiện lệnh kế khi lệnh trước đó đã được thực hiện xong. Còn trong hệ điều hành Uả IX ta có thể đặt lệnh chạy ở chế độ nền (background) đồng thời khi đó có thể thực hiện các lệnh kế.

Nhiều người sử dụng

ả hiều người sử dụng có thể sử dụng máy tính có cài Uả IX tại một thời điểm.

Độc lập phần cứng

Vì hệ điều hành Uả IX được viết bằng ngôn ngữ cấp cao cho nên nó rất dễ cài đặt trên các cấu hình phần cứng khác. Hơn nữa với cách tổ chức các thiết bị là các tập tin đặc biệt nên việc thêm vào hay loại bỏ các thiết bị rất dễ dàng.

Dùng chung thiết bị

Vì Unix là môi trường nhiều người sử dụng do đó các thiết bị ngoại vi như máy in,v.v... có thể được dùng chung bởi nhiều người sử dụng.

Tính ổn định

Linux có tính ổn định cao, đây là một trong những ưu điểm của Linux so với các hệ điều hành khác. Tính ổn định ở đây có nghĩa là nó ít bị lỗi khi sử dụng so với hầu hết các hệ điều hành khác. ả gười sử dụng Linux sẽ không phải lo lắng đến chuyện máy tính của mình bị hiện tượng "treo cứng" khi đang sử dụng nữa. Thông thường lý do để ta bắt buộc phải khởi động lại hệ thống là do mất điện, nâng cấp phần cứng hoặc phần mềm.

Tính bảo mật

Khi làm việc trên Linux người dùng có thể an tâm hơn về tính bảo mật của hệ điều hành. Linux là hệ điều hành đa nhiệm, đa người dùng, điều này có nghĩa là có thể có nhiều người dùng vào phiên làm việc của mình trên cùng một máy tại cùng một thời điểm. Linux cung cấp các mức bảo mật khác nhau cho người sử

dụng. Mỗi người sử dụng chỉ làm việc trên một không gian tài nguyên riêng, chỉ có người quản trị hệ thống mới có quyền thay đổi trong máy.

Tính hoàn chỉnh

Bản thân Linux đã kèm theo các trình tiện ích cần thiết. Tất cả các trình tiện ích mà ta mong đợi đều có sẵn hoặc ở một dạng tương đương rất giống. Trên Linux, các trình biên dịch như C, C++, …, đều được chuẩn hoá.

Tính tương thích

Linux tương thích hầu như hoàn toàn với hầu hết các chuẩn Unix như IEEE POSIX.1, Uå IX System V và BSD Unix. Trên Linux ta cũng có thể tìm thấy các trình giả lập DOS và Windows cho phép ta chạy các ứng dụng quen thuộc trên DOS và Windows. Linux cũng hỗ trợ hầu hết các phần cứng PC như đã nói phía trên.

Hệ điều hành 32-bit đầy đủ

å gay từ đầu Linux đã là hệ điều hành 32 bit đầy đủ. Điều đó có nghĩa là ta không còn phải lo về giới hạn bộ nhớ, các trình điều khiển EMM hay các bộ nhớ mở rộng,… khi sử dụng Linux.

Linux hỗ trợ tốt cho tính toán song song và máy tính cụm (PC-cluster) là một hướng nghiên cứu triển khai ứng dụng nhiều triển vọng hiện nay.

Linux có giao diện đồ hoạ (GUI):

Thừa hưởng từ hệ thống X-Window. Linux hỗ trợ nhiều giao thức mạng, bắt nguồn và phát triển từ dòng BSD. Thêm vào đó, Linux còn hỗ trợ tính toán thời gian thực

Dễ cấu hình

Ta không còn phải bận tâm về giới hạn 640K và tiến hành tối ưu hoá bộ nhớ mỗi lần cài đặt một trình điều khiển mới. Linux cho ta toàn quyền điều khiển về cách làm việc của hệ thống.

å hư vậy, qua phần giới thiệu ban đầu này ta có thể thấy rằng Linux là một hệ Unix đủ mạnh. å ó có thể được ứng dụng dễ dàng. å goài ra, việc sử dụng công cộng rộng rãi đang làm đà để Linux phát triển nhanh. Các quy trình thiết lập cho phép cài đặt trực tiếp hệ thống đã làm nó trở nên ngày càng phổ biến đối với những người sử dụng.

Tuy nhiên cũng tồn tại một số khó khăn làm cho Linux chưa thực sự trở thành một hệ điều hành phổ dụng, dưới đây là một số khó khăn điển hình:

+ Tuy đã có công cụ hỗ trợ cài đặt, tuy nhiên, việc cài đặt Linux còn tương đối phức tạp và khó khăn. Khả năng tương thích của Linux với một số loại thiết bị phần cứng còn thấp do chưa có các trình điều khiển cho nhiều thiết bị,
+ Phần mềm ứng dụng chạy trên nền Linux tuy đã phong phú song so với một số hệ điều hành khác, đặc biệt là khi so sánh với MS Windows, thì vẫn còn có khoảng cách.

Với sự hỗ trợ của nhiều công ty tin học hàng đầu thế giới (IBM, SUẩ , HP ...) và sự tham gia phát triển của hàng vạn chuyên gia trên toàn thế giới thuộc cộng đồng Linux, các khó khăn của Linux chắc chắn sẽ nhanh chóng được khắc phục.

Chính vì lẽ đó đã hình thành một số nhà cung cấp Linux trên thế giới. Bảng dưới đây là tên của một số nhà cung cấp Linux có tiếng nhất và địa chỉ website của họ.

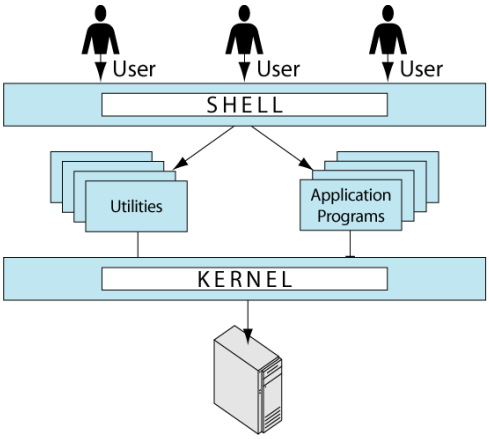
Đáng chú ý nhất là Red Hat Linux (tại Mỹ) và Red Flag Linux (tại Trung Quốc). Red Hat được coi là lâu đời và tin cậy, còn Red Flag là một công ty Linux của Trung quốc, có quan hệ với cộng đồng Linux Việt nam và chúng ta có thể học hỏi một cách trực tiếp kinh nghiệm cho quá trình đưa Linux vào Việt nam.

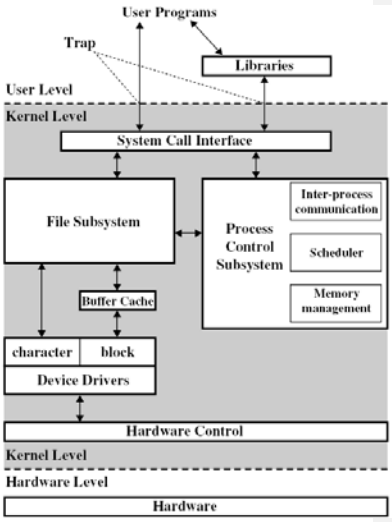
Tên công ty	***Địa chỉ website***
Caldera OpenLinux	www.caldera.com
Corel Linux	www.corel.com
Debian Gẩ U/Linux	www.debian.com
Linux Mandrake	www.mandrake.com
Red Hat Linux	www.redhat.com
Red Flag Linux	www.redflag-linux.com
Slackware Linux	www.slackware.com
SuSE Linux	www.suse.com
TurboLinux	www.turbolinux.com
	www.ubuntu.com

2.2 Các thành phần chính của hệ điều hành Linux

- Kernel (ẩ hân hệ điều hành).

- Các bộ điều khiển thiết bị.
- Lệnh và tiện ích.
- Shell.
- Windows & Graphic User Interface.





Hình1.2 Các thành phần chính của HĐH Unix

Kernel

Là thành phần chủ yếu hay trái tim của hệ điều hành. ả ó nắm nhiệm vụ điều khiển giao dịch giữa chương trình người sử dụng với các thiết bị phần cứng, xếp lịch các tiến trình để có thể thực hiện đa nhiệm, và nhiều tác vụ khác của hệ thống, và một tập các trình đơn nằm trong bộ nhớ, mọi tiến trình đều gọi chúng. ả hân hệ điều hành chịu trách nhiệm duy trì các đối tượng trừu tượng quan trọng của hệ điều hành, bao gồm bộ nhớ ảo và quá trình. Các mô đun chương trình trong nhân được đặc quyền trong hệ thống, bao gồm đặc quyền thường trực ở bộ nhớ trong.

ả hân (còn được gọi là hệ lõi) của Linux, là một bộ các môdun chương trình có vai trò điều khiển các thành phần của máy tính, phân phối các tài nguyên cho người dùng (các quá trình người dùng). ả hân chính là cầu nối giữa chương trình ứng dụng với phần cứng. ả gười dùng sử dụng bàn phím gõ nội dung yêu cầu của mình và yêu cầu đó được nhân gửi tới shell, Shell phân tích lệnh và gọi các chương trình tương ứng với lệnh để thực hiện.

Một trong những chức năng quan trọng nhất của nhân là giải quyết bài toán lập lịch, tức là hệ thống cần phân chia CPU cho nhiều quá trình hiện thời cùng tồn tại. Đối với Linux, số lượng quá trình có thể lên tới con số hàng nghìn. Với số lượng quá trình đồng thời nhiều như vậy, các thuật toán lập lịch cần phải đủ hiệu quả: Linux thường lập lịch theo chế độ Round Robin (RR) thực hiện việc luân chuyển CPU theo lượng tử thời gian.

Thành phần quan trọng thứ hai trong nhân là hệ thống các môđun chương trình (được gọi là lời gọi hệ thống) làm việc với hệ thống file. Linux có hai cách thức làm việc với các file: làm việc theo byte (kí tự) và làm việc theo khối. Một đặc điểm đáng chú ý là file trong Linux có thể được nhiều người cùng truy nhập tới nên các lời gọi hệ thống làm việc với file cần đảm bảo việc file được truy nhập theo quyền và được chia xẻ cho người dùng.

Các bộ điều khiển thiết bị

Uả IX thể hiện các thiết bị vật lý như các tập tin đặc biệt. Một tập tin đặc biệt sẽ có một điểm vào trong thư mục và có một tên tập tin. Do đó Unix cho phép người sử dụng định nghĩa tên thiết bị.

Các thiết bị được chia làm hai loại: ký tự và khối.

- Thiết bị ký tự đọc và ghi dòng các ký tự (ví dụ các thiết bị đầu cuối).
- Thiết bị khối đọc và ghi dữ liệu trong các khối có kích thước cố định (ví dụ ổ đĩa).

Thiết bị có thể đổi tên như đổi tên tập tin. Thư mục chứa các bộ điều khiển thiết bị là /dev.

Lệnh và tiện ích

Tiện ích hệ thống là các chương trình thi hành các nhiệm vụ quản lý riêng rẽ, chuyên biệt. Một số tiện ích hệ thống được gọi ra chỉ một lần để khởi động và cấu hình phương tiện hệ thống, một số tiện ích khác, theo thuật ngữ Uả IX được gọi là trình chạy ngầm (daemon), có thể chạy một cách thường xuyên (thường theo chu kỳ), điều khiển các bài toán như hưởng ứng các kết nối mạng mới đến, tiếp nhận yêu cầu logon, hoặc cập nhật các file log.

Tiện ích (hay lệnh) có sẵn trong hệ điều hành (dưới đây tiện ích được coi là lệnh thường trực). ả ội dung chính yếu của tài liệu này giới thiệu chi tiết về một số lệnh thông dụng nhất của Linux.

Các lệnh và tiện ích của Unix rất đa dạng.

- Một lệnh Uả IX có dạng: $lệnh [các chọn lựa] [các đối số] lệnh thường là chữ nhỏ.
- Unix phân biệt chữ lớn, nhỏ. **Ví dụ:** $ls -c /dev

Ta có thể chia lệnh thành các nhóm sau:

Các lệnh khởi tạo:

exit	thoát khỏi hệ thống (Bourne-Shell)
logout	thoát khỏi hệ thống C-Shell
id	chỉ danh của người sử dụng
logname	tên người sử dụng login
man	giúp đỡ
newgrp	chuyển người sử dụng sang một nhóm mới
psswd	thay đổi password của người sử dụng
set	xác định các biến môi trường
tty	đặt các thông số terminal
uname	tên của hệ thống (host)
who	cho biết những ai đang thâm nhập hệ thống

Trình báo màn hình:

echo	hiển thị dòng ký tự hay biến
setcolor	đặt màu nền và chữ của màn hình

Desktop:

bc	tính biểu thức số học
cal	máy tính cá nhân
date	hiển thị và đặt ngày
mail	gửi - nhận thư tín điện tử
mesg	cấm/cho phép hiển thị thông báo trên màn hình(bởi write/hello)
spell	kiểm tra lỗi chính tả

vi	soạn thảo văn bản
write/hello	cho phép gửi dòng thông báo đến những người sử dụng trong hệ thống.

Thư mục:

cd	đổi thư mục
copy	sao chép 2 thư mục
mkdir	tạo thư mục
rmdir	loại bỏ thư mục
pwd	trình bày thư mục hiện hành

Tập tin:

cas/more	trình bày nội dung tập tin
cp	sao chép một hay nhiều tập tin
find	tìm vị trí của tập tin
grep	tìm vị trí của chuỗi ký tự trong tập tin
ls, l, lf, lc	trình bày tên và thuộc tính của các tập tin trong thư mục
mv	chuyển/ đổi tên một tập tin
sort	sắp thứ tự nội dung tập tin
wc	đếm số từ trong tập tin

Quản lý tiến trình:

kill	hủy bỏ một quá trình
ps	trình bày tình trạng của các quá trình
sleep	ngưng hoạt động một thời gian

Kiểm soát chủ quyền:

chgrp	chuyển chủ quyền tập tin, thư mục từ một nhóm sang một nhóm khác
chmod	thay đổi quyền sở hữu của tập tin hay thư mục
chown	thay đổi người sở hữu tập tin hay thư mục

Kiểm soát in:

cancel	ngưng in
lp	in tài liệu ra máy in

lpstat trạng thái của hàng chờ in

Shell

Là bộ xử lý lệnh của người sử dụng hay nó đơn giản chỉ là một chương trình cho phép hệ thống hiểu các lệnh của người dùng.

Chức năng chính của Shell là:

- Sử lý tương tác: Khi Shell được sử dụng một cách tương tác, hệ thống đợi người dùng gõ vào một lệnh tại dấu nhắc lệnh. Lệnh có thể bao gồm các ký hiệu đặc biệt cho phép ta viết tắt các tên file hoặc tái định hướng nguồn vào và nguồn ra.
- Lập trình: Các shell cung cấp một bộ các lệnh đặc biệt (có sẵn), cho phép ta tạo ra các chương trình có tên *Shell Script*. Các Shell Script rất hữu ích khi sử dụng cho việc thực thi một chuỗi các lệnh riêng biệt giống như thực thi các file BATCH trong MS-Dos. Các script cũng có thể thực thi các lệnh lặp lại nhiều lần (trong vòng lặp) hoặc có điều kiện (if-else) giống như trong nhiều ngôn ngữ lập trình cao cấp khác.

Hiện nay người ta sử dụng ba loại shell, tùy theo loại mà có cú pháp khác nhau:

- Bourne-Shell : là shell cơ bản nhất, nhanh, hiệu quả, nhưng ít lệnh.
- C-Shell: là shell sử dụng cú pháp giống như C và nó thuận tiện hơn cho người sử dụng tương tác Bourne-Shell, nó giống như Bourne-Shell nhưng cung cấp thêm các cấu trúc điều khiển, history, bí danh.
- Korn-Shell: Kết hợp cả Bourne-Shell và C-Shell.

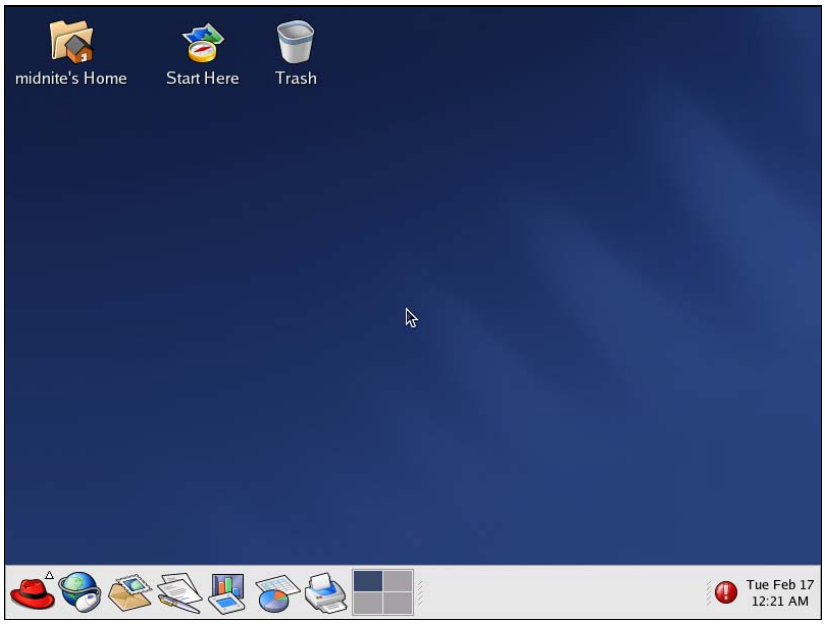
Mỗi người dùng khi đăng nhập hệ thống thì thường có một chương trình mặc định khởi động cùng, có thể nhận biết dạng Shell ta đang sử dụng là gì thông qua file /etc/passwd.

Tên chương trình	*Shell của ta là*
/bin/sh	Bourne - Shell
/bin/rsh	Bourne – Shell
/bin/jsh	Bourne – Shell
/bin/ksh	Korn-Shell
/usr/dt/bin/dtksh	Korn-Shell Desktop, một phiên bản chỉ dùng cho Solaris

/bin/rksh	Korn-Shell
/bin/csh	C Shell

Windows & Graphic User Interface:

Giao tiếp đồ hoạ và cửa sổ là một khả năng rất mạnh của hệ điều hành Linux, nó cho phép hệ điều hành giao tiếp thân thiện hơn với người sử dụng.



Hình 1.3 Giao diện Gnome của Linux

Tóm lại: Đứng về phía người sử dụng ta có thể hình dung hệ điều hành Linux như sau:

ả gười sử dụng - lệnh Linux - biên dịch Shell - Kernel - Máy tính (phần cứng).

CHƯƠNG 2: HỆ THỐNG FILE TRONG LINUX

1. Các kiểu file có trong Linux

Có rất nhiều file khác nhau trong Linux, nhưng bao giờ cũng tồn tại một số kiểu file cần thiết cho hệ điều hành và người dùng, dưới đây giới thiệu lại một số các kiểu file cơ bản.

- File người dùng (user data file): là các file tạo ra do hoạt động của người dùng khi kích hoạt các chương trình ứng dụng tương ứng. Ví dụ như các file thuần văn bản, các file cơ sở dữ liệu hay các file bảng tính.
- File hệ thống (system data file): là các file lưu trữ thông tin của hệ thống như: cấu hình cho khởi động, tài khoản của người dùng, thông tin thiết bị ... thường được cất trong các tệp dạng văn bản để người dùng có thể can thiệp, sửa đổi theo ý mình.
- File thực hiện hay thực thi (executable file): là các file chứa mã lệnh hay chỉ thị cho máy tính thực hiện. File thực hiện lưu trữ dưới dạng mã máy mà ta khó có thể tìm hiểu được ý nghĩa của nó, nhưng tồn tại một số công cụ để "hiểu" được các file đó. Khi dùng trình ứng dụng mc, file thực hiện được bắt đầu bởi dấu (*) và thường có màu xanh lục.
- Thư mục hay còn gọi là file bao hàm (directory): là file bao hàm các file khác và có cấu tạo hoàn toàn tương tự như file thông thường khác nên có thể gọi là file. Trong mc, file bao hàm thường có màu trắng và bắt đầu bằng dấu ngã (~) hoặc dấu chia (/). Ví dụ: /, /home, /bin, /usr, /usr/man, /dev ...
- File thiết bị (device file): là file mô tả thiết bị, dùng như là định danh để chỉ ra thiết bị cần thao tác. Theo quy ước, file thiết bị được lưu trữ trong thư mục /dev. Các file thiết bị hay gặp trong thư mục này là tty (teletype - thiết bị truyền thông), ttyS (teletype serial - thiết bị truyền thông nối tiếp), fd0, fd1, ... (floppy disk- thiết bị ổ đĩa mềm), hda1, hda2, ... hdb1, hdb2, ... (hardisk - thiết bị ổ cứng theo chuẩn IDE; a, b,... đánh số ổ đĩa vật lý; 1, 2, 3... đánh số ổ logic). Trong mc, file thiết bị có màu tím và bắt đầu bằng dấu cộng (+).
- File liên kết (linked file): là những file chứa tham chiếu đến các file khác trong hệ thống tệp tin của Linux. Tham chiếu này cho phép người dùng tìm

Formatted: Bullets and Numbering

nhanh tới file thay vì tới vị trí nguyên thủy của nó. Hơn nữa, người ta có thể gắn vào đó các thông tin phụ trợ làm cho file này có tính năng trội hơn so với tính năng nguyên thủy của nó. Ta thấy loại file này giống như khái niệm shortcut trong MS-Windows98.

Không giống một số hệ điều hành khác (như MS-DOS chẳng hạn), Linux quản lý thời gian của tệp tin qua các thông số thời gian truy nhập (accesed time), thời gian kiến tạo (created time) và thời gian sửa đổi (modified time).

2. Quy ước tên file trong Linux

Một đối tượng điển hình trong các hệ điều hành đó là ***file***. File là một tập hợp dữ liệu có tổ chức được hệ điều hành quản lý theo yêu cầu của người dùng. Cách tổ chức dữ liệu trong file thuộc về chủ của nó là người đã tạo ra file. File có thể là một văn bản (trường hợp đặc biệt là chương trình nguồn trên C, PASCAL, shell script ...), một chương trình ngôn ngữ máy, một tập hợp dữ liệu ...

Hệ điều hành quản lý file theo tên gọi của file (tên file) và một số thuộc tính liên quan đến file. Trước khi giới thiệu một số nội dung liên quan đến tên file và tên thư mục, chúng ta giới thiệu sơ bộ về khái niệm thư mục.

Để làm việc được với các file, hệ điều hành không chỉ quản lý nội dung file mà còn phải quản lý các thông tin liên quan đến các file. Thư mục (directory) là đối tượng được dùng để chứa thông tin về các file, hay nói theo một cách khác, thư mục chứa các file. Các thư mục cũng được hệ điều hành quản lý vì vậy, thư mục cũng được coi là file song trong một số trường hợp để phân biệt với "file" thư mục, chúng ta dùng thuật ngữ ***file thông thường***. Khác với file thông thường, hệ điều hành lại quan tâm đến nội dung của thư mục.

Formatted: Bullets and Numbering

Tên file trong Linux có thể dài tới 256 ký tự, bao gồm các chữ cái, chữ số, dấu gạch nối, gạch chân, dấu chấm. Tên thư mục/file trong Linux có thể có nhiều hơn một dấu chấm, ví dụ: *This_is.a.VERY_long.filename.* ả ếu trong tên file có dấu chấm "." thì xâu con của tên file từ dấu chấm cuối cùng được gọi là phần mở rộng của tên file (hoặc file). Ví dụ, tên file trên đây có phần mở rộng là *.filename*.

Chúng ta nên lưu ý rằng, không phải ký tự nào cũng có nghĩa. ả ếu có hai file chỉ khác nhau ở ký tự cuối cùng, thì đối với Linux, đó là hai file có thể trùng tên. Bởi lẽ, Linux chỉ lấy 32 hay 64 ký tự đầu tiên trong tên file mà thôi (tùy theo phiên

bản Linux), phần tên file còn lại dành cho chủ của file, Linux theo dõi thông tin, nhưng thường không xem các ký tự đứng sau ký tự thứ 33 hay 65 là quan trọng đối với nó.

- Xin nhắc lại lưu ý về phân biệt chữ hoa và chữ thường đối với tên thư mục/file, ví dụ hai file *FILENAME.tar.gz* và *filename.tar.gz* là hai file khác nhau.
- ả ếu trong tên thư mục/file có chứa khoảng trống, sẽ phải đặt tên thư mục/file vào trong cặp dấu nháy kép để sử dụng thư mục/file đó. Ví dụ, để tạo thư mục có tên là "*My document"* chẳng hạn, hãy đánh dòng lệnh sau:

```
# mkdir "My document"
```

- Một số ký tự sau không được sử dụng trong tên thư mục/file: !, *, $, &, # ... Formatted: Bullets and Numbering
- Khi sử dụng chương trình **mc**, việc hiển thị tên file sẽ bổ sung một kí tự theo nghĩa: dấu "*" cho file khả thi trong Linux, dấu "~" cho file sao lưu, dấu "." cho file ẩn, dấu "@" cho file liên kết...

Tập hợp tất cả các file có trong hệ điều hành được gọi là ***hệ thống file*** là một hệ thống thống nhất. Bởi chính từ cách thức sử dụng thư mục, hệ thống file được tổ chức lôgic theo dạng hình cây: Hệ thống file được xuất phát từ một thư mục gốc (được kí hiệu là "**/**") và cho phép tạo ra thư mục con trong một thư mục bất kỳ. Thông thường, khi khởi tạo Linux đã có ngay hệ thống file của nó.

3. Cấu trúc hệ thống file của Linux

Hệ thống file của linux gồm bốn thành phần chính là Boot block (dùng để khởi động hệ thống), Siêu khối (Super block), Danh sách inode và Vùng dữ liệu.

Block 0

Thường không được sử dụng và thường chứa mã để nạp HĐH (boot the computer). ả ó chứa một đoạn chương trình sẽ được đọc vào máy khi khởi động hệ điều hành Mặc dù Boot block chỉ cần thiết khi khởi động máy nhưng tương tự với Boot record của DOS, tất cả các hệ thống file Uả IX đều có một Boot block (block này có thể để trống).

Block 1:

Là Super Block (siêu khối), trình bày trạng thái của hệ thống File (số lượng I-node, số Disk Block, điểm bắt đầu của danh danh sách của khối đĩa trống (free disk blocks)). Là một dạng bản ghi mô tả tình trạng của hệ thống file. ả ó gồm các thông tin sau:

- Kích thước hệ thống file.
- Số khối còn trống trong hệ thống file.
- Danh sách khối trống trong hệ thống file.
- Chỉ số của khối tiếp theo trong danh sách khối trống.
- Kích thước của danh sách inode.
- Số inode còn trống trong hệ thống file.
- Danh sách inode còn trống trong hệ thống file.
- Chỉ số inode tiếp theo trong danh sách inode trống trong hệ thống file.
- Trường khoá của danh sách khối và inode trống.
- Cờ báo hiệu super block đã bị thay đổi.

I-nodes

Tương ứng bảng FAT trong MS-DOS, trình bày bên trong của một File được cho bởi một I-node, chứa đựng các thông tin mô tả về lưu trữ file trên đĩa và một số thông tin khác như: người chủ sở hữu, quyền truy nhập, thời gian truy nhập file. Mỗi I-node dài 64 byte và miêu tả chính xác một file. Inode là một bảng chứa các thông tin chi tiết về một file. Mỗi file đều được gắn với một inode qua số hiệu inode. Khi file được sử dụng bởi một tiến trình nào đó thì inode sẽ được đọc vào bộ nhớ và quản lý bởi kernel. Mỗi inode bao gồm các thông tin sau:

- ***Quyền sở hữu file:*** Quyền sở hữu được chia làm hai phần là người sở hữu file và nhóm người sở hữu. ả gười sở hữu thường là người tạo ra file đó. ả hóm người sở hữu file, trong Uả IX System V thì thường thuộc về nhóm của người tạo ra file đó, còn trong BSD Uả IX thì file thuộc về nhóm sở hữu thư mục mà file được tạo ra. Quyền sở hữu của người sử dụng và của nhóm đối với mỗi file có thể thay đổi được (ví dụ lệnh chown, chgrp của shell). Quyền sở hữu này cùng với quyền truy nhập của file sẽ quyết dịnh xem ai có thể truy nhập tới tập tin và có thể truy nhập như thế nào.
- ***Loại file:*** Khái niệm file của Uả IX có khác so với file trong DOS, ta có thể kể tới một số loại file sau.
    - *Kiểu file thường*: Đó là các file văn bản , các file nhị phân, file dữ liệu hay là các file chương trình...

- *Thư mục con*: Là những file tạo ra cấu trúc phân cấp cho hệ thống file gồm danh sách các file trong nó và có thể chứa cả các thư mục khác. ẩ ó có một vai trò quan trọng trong việc biến đổi tên file thành số hiệu inode. Thư mục là một file mà toàn bộ dữ liệu là chuỗi các phần tử (entry), mỗi phần tử chứa một số hiệu inode và tên file tương ứng trong thư mục. Đối với hệ Uẩ IX System V chỉ cho phép tên file tối đa dài 14 ký tự còn đối với các hệ khác chiều dài này có thể lớn hơn. Do thư mục là các file đặc biệt nên tuy các file có thể đọc dữ liệu trong thư mục như đối với các file thường nhưng kernel giành quyền ghi thư mục để đảm bảo tính chính xác của cấu trúc.
- *Kiểu file đặc biệt*: Đây là cơ chế mà Uẩ IX sử dụng để truy nhập tới các thiết bị vào ra. Mỗi thiết bị vào ra trong Uẩ IX đều được coi như là một file trong hệ thống file. Ta có thể truy nhập tới thiết bị vật lý thông qua việc truy nhập các file này. ẩ gười ta chia kiểu này làm hai loại dựa trên cách truy nhập tới chúng, đó là kiểu ký tự (character, ví dụ như file ứng với cổng nối tiếp) và kiểu khối (block, ví dụ như file ứng với ổ đĩa).
- *Kiểu file móc nối(symbolic link)*: Đây là file chứa đường dẫn tới một file khác. Cơ chế này cho phép ta truy nhập tới một tập tin bằng nhiều tên khác nhau. Thực chất của nó là định nghĩa một file với một tên khác.
- *Kiểu FIFO*: là một hàng đợi (queue) theo kiểu first-in-first-out hay còn được gọi là named pipe. FIFO được dùng để trao đổi dữ liệu giữa các tiến trình. Loại file này chỉ có trong hệ Uẩ IX System V mà không có trong BSD Uẩ IX.
- *Kiểu socket*: là một cơ chế tạo ra các đầu cuối (endpoint) cho phép các tiến trình liên hệ với nhau. Khái niệm socket sẽ được đề cập tới trong phần sau.

- ***Quyền truy nhập file:*** Hệ thống bảo vệ file theo 3 lớp người sử dụng là chủ sở hữu, nhóm sở hữu và các người sử dụng khác. Mỗi lớp người sử dụng đều có 3 quyền đọc, ghi, và thực hiện. Các quyền này được thiết lập tách biệt nhau. Do thư mục là một kiểu file đặc biệt nên quyền truy nhập tới thư mục có thay đổi. Quyền đọc cho phép tiến trình được đọc thư mục, quyền ghi cho phép tạo ra hoặc xoá bỏ các phần tử của thư mục (thông qua lệnh creat, mknod, link hay unlink), quyền thực hiện cho phép tiến trình tìm kiếm tên file trong thư mục.
- ***Thời gian:*** Lưu trữ thời gian mà file bị thay đổi gần nhất, thời gian file được truy cập gần nhất và thời gian inode bị thay đổi gần nhất.
- ***Số file liên kết:*** Thể hiện số file có trong cấu trúc cây thư mục.

- ***Bảng địa chỉ các khối dữ liệu:*** Mặc dù người sử dụng xử lý file như một chuỗi liên tiếp các byte nhưng trong kernel lưu trữ dữ liệu trên những khối không liên tiếp. inode phải xác định các khối chứa dữ liệu của file. Bảng này được mã hoá khá phức tạp để có thể chứa một số lượng địa chỉ thay đổi nhưng kích thước bảng lại không thay đổi.
- ***Kích thước file:*** Lưu giữ chính xác kích thước thực của file.

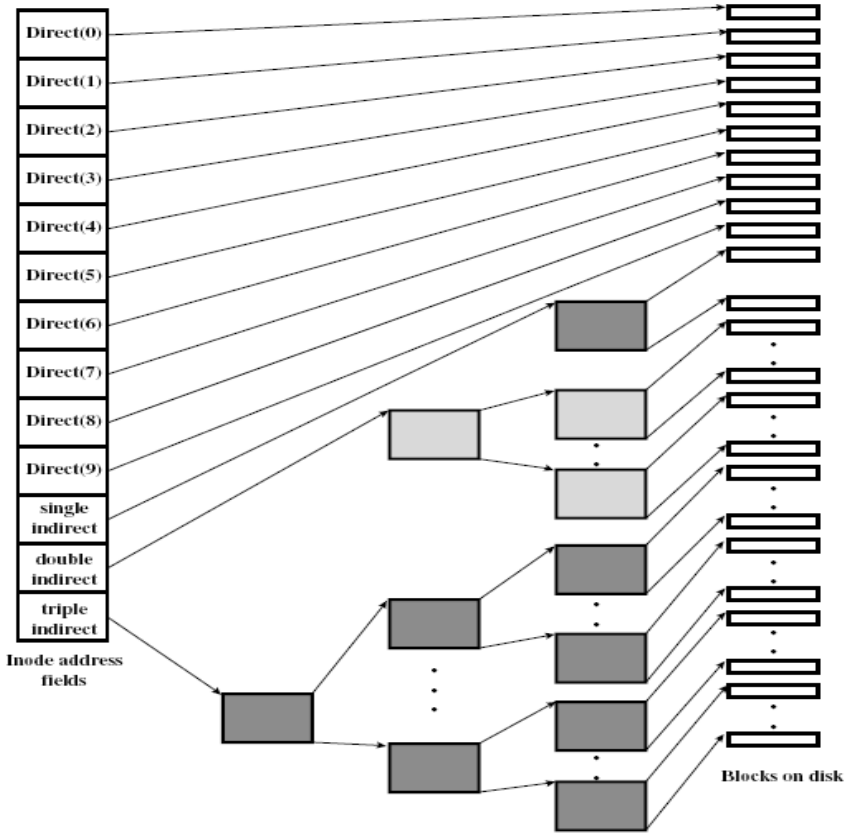
Chú ý : Inode hoàn toàn không lưu giữ tên file và không thể xác định đường dẫn tới file thông qua inode. Khi inode được đọc vào bộ nhớ, một số trường được thêm vào làm cho inode trong bộ nhớ (***in-core inode***) khác với inode trên đĩa.

- Trường trạng thái inode báo hiệu:
  - Inode bị khoá.
  - Có một tiến trình đang đợi cho đến khi inode được mở khoá.
  - In-core inode khác với inode trên đĩa do bị thay đổi.
  - File trong bộ nhớ đã thay đổi so với file trên đĩa.
  - File này là một điểm kết nốt với một hệ thống file khác (mount point).
- Số hiệu thiết bị logic của hệ thống file chứa file này.
- Số hiệu inode.
- Con trỏ tới in-core inode khác.
- Số đếm : Ghi nhận số file đang được mở.

Ghi chú: Danh sách inode đứng ngay sau super block. ẩ gười quản trị hệ thống sẽ quyết định kích thước của danh sách này khi thiết lập cấu hình hệ thống. Kernel của hệ điều hành tham chiếu tới vùng này bằng cách đánh chỉ số cho danh sách. Trong danh sách inode tồn tại một inode là inode gốc của hệ thống file (tương tự thư mục gốc trong hệ điều hành DOS). Inode này là điểm đầu tiên của cấu trúc thư mục trong hệ thống file và làm cho hệ thống file có thể truy nhập bình thường sau khi thực hiện lệnh "mount".

Khối dữ liệu (data block)

Tất cả các file và thư mục được lưu trữ tại đây. Hệ thống file trong Unix là một cấu trúc phân cấp có bảo mật cao. File có thể được tổ chức lưu trữ theo một vùng liên tục hay nhiều vùng liên tục. Bắt đầu từ sau danh sách inode cho tới khối cuối cùng của hệ thống file. Phần này chỉ chứa dữ liệu và thông tin quản trị hệ thống. Một khối dữ liệu chỉ được cấp phát cho một và chỉ một file duy nhất trong hệ thống file.



Hình 2.1 Inodes

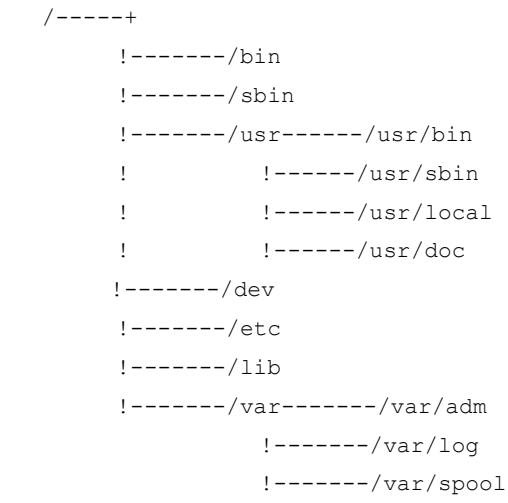
4. Cấu trúc cây thư mục của hệ thống file trong Linux

Đối với hệ điều hành linux, không có khái niệm các ổ đĩa khác nhau. Sau quá trình khởi động, toàn bộ các thư mục và tập tin được "gắn" (mount) lên và tạo thành một hệ thống tập tin thống nhất, bắt đầu từ gốc '/'.

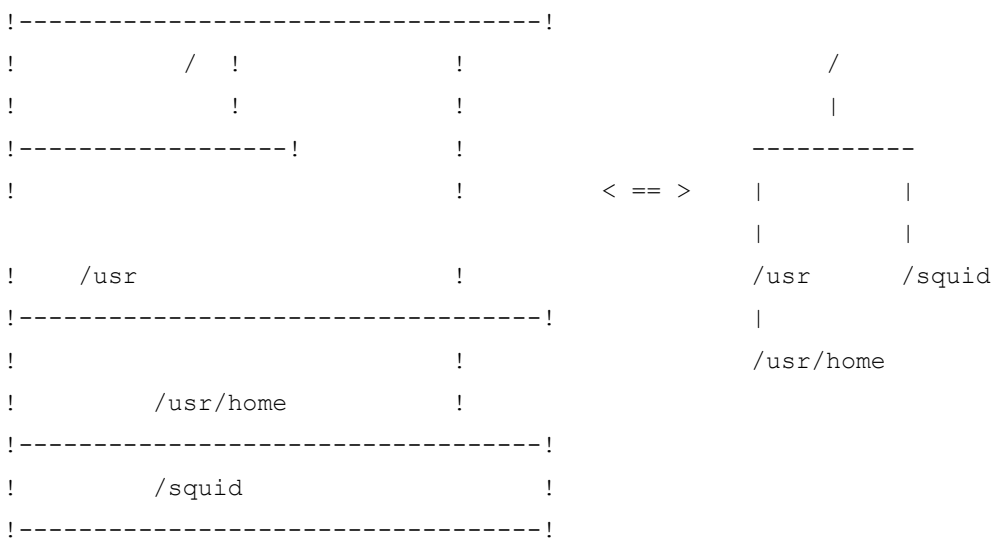
Hình dưới là cây thư mục của đa số các Linux. Với cây thư mục trên ta không thể nào biết được số lượng ổ đĩa cứng, các phân mảnh (partition) của mỗi đĩa và sự tương ứng giữa các phân mảnh và thư mục như thế nào.

Chúng ta có thể chia đĩa cứng thành nhiều phân mảnh (partition). Mỗi partition là một hệ thống tập tin độc lập. Sau đó, các hệ thống tập tin này được 'gắn ' (mount) vào hệ thống tập tin thống nhất của toàn hệ thống.

Chúng ta hoàn toàn có thể gắn thêm một đĩa cứng mới, format rồi mount vào hệ thống tập tin dưới tên một thư mục nào đó tại một điểm (mount point) nào đó.



Đối với các chương trình chạy trên Linux, không hề có khái niệm một thư mục nằm ở ổ đĩa nào hay partition nào. Hình sau đây cho thấy sự tương quan giữa vị trí vật lý trên đĩa và vị trí logic trong cây tập tin.



Thư mục **/usr/home** là thư mục con của **/usr** trong cây thư mục, nhưng trên đĩa vật lý, đây là hai phân mảnh (partition) cạnh nhau.

Một số thư mục quan trọng trong Unix/Linux

Thư mục gốc /

Đây là thư mục gốc chứa đựng tất cả các thư mục con có trong hệ thống.

Thư mục /root

ẩ hư đã được giới thiệu thư mục /root có thể được coi là "thư mục riêng" của người quản trị hệ thống. Thư mục này được sử dụng để lưu trữ các file tạm thời, nhân Linux và ảnh khởi động, các file nhị phân quan trọng (những file được sử dụng đến trước khi Linux có thể gắn kết đến phân vùng /user), các file đăng nhập quan trọng, bộ đệm in cho việc in ấn, hay vùng lưu tạm cho việc nhận và gửi email. ẩ ó cũng được sử dụng cho các vùng trống tạm thời khi thực hiện các thao tác quan trọng, ví dụ như khi xây dựng (build) một gói RPM từ các file RPM nguồn.

Thư mục /bin

Trong Linux, chương trình được coi là khả thi nếu nó có thể thực hiện được. Khi một chương trình được biên dịch, nó sẽ có dạng là file nhị phân. ẩ hư vậy, chương trình ứng dụng trong Linux là một file nhị phân khả thi. Vì vậy, những nhà phát triển Linux đã quyết định phải tổ chức một thư mục "binaries" để lưu trữ các chương trình khả thi có trên hệ thống, đó chính là thư mục /bin. Ban đầu, thư mục /bin (bin là viết tắt của từ binary) là nơi lưu trữ các file nhị phân khả thi. ẩ hưng theo thời gian, ngày càng có nhiều hơn các file khả thi có trong Linux, do đó, có thêm các thư mục như /sbin, /usr/bin được sử dụng để lưu trữ các file đó.

Thư mục /dev

Một phần không thể thiếu trong bất kỳ máy tính nào đó là các trình điều khiển thiết bị. Không có chúng, sẽ không thể có được bất kỳ thông tin nào trên màn hình của (các thông tin có được do trình điều khiển thiết bị hiển thị đưa ra). Cũng không thể nhập được thông tin (những thông tin do trình điều khiển thiết bị bàn phím đọc và chuyển tới hệ thống), và cũng không thể sử dụng đĩa mềm của (được quản lý bởi trình điều khiển đĩa mềm).

Tất cả các trình điều khiển thiết bị đều được lưu trữ trong thư mục /dev.

Thư mục /etc

Quản trị hệ thống trong Linux không phải là đơn giản, chẳng hạn như việc quản lý tài khoản người dùng, vấn đề bảo mật, trình điều khiển thiết bị, cấu hình phần cứng, v.v.. Để giảm bớt độ phức tạp, thư mục */etc* đã được thiết kế để lưu trữ tất cả các thông tin hay các file cấu hình hệ thống.

Thư mục /lib

Linux có một trung tâm lưu trữ các thư viện hàm và thủ tục, đó là thư mục /lib.

Thư mục /lost+found

Một file được khôi phục sau khi có bất kỳ một vấn đề hoặc gặp một lỗi về ghi đĩa trên hệ thống đều được lưu vào thư mục này.

Thư mục /mnt

Thư mục /mnt là nơi để kết nối các thiết bị (ví dụ đĩa cứng, đĩa mềm...) vào hệ thống file chính nhờ lệnh mount. Thông thường các thư mục con của /mnt chính là gốc của các hệ thống file được kết nối: /mnt/floppy: đĩa mềm, /mnt/hda1: vùng đầu tiên của đĩa cứng thứ nhất (hda), /mnt/hdb3: vùng thứ ba của đĩa cứng thứ 2 (hdb)..

Thư mục /tmp

Thư mục /tmp được rất nhiều chương trình trong Linux sử dụng như một nơi để lưu trữ các file tạm thời. Ví dụ, nếu đang soạn thảo một file, chương trình sẽ tạo ra một file là bản sao tạm thời (bản nháp) của file đó và lưu vào trong thư mục /tmp.

Việc soạn thảo thực hiện trực tiếp trên file tạm thời này và sau khi soạn thảo xong, file tạm thời sẽ được ghi đè lên file gốc. Cách thức như vậy bảo đảm sự an toàn đối với file cần soạn thảo.

Thư mục /usr

Thông thường thì thư mục /usr là trung tâm lưu trữ tất cả các lệnh hướng đến người dùng (user-related commands). Tuy nhiên, ngày nay thật khó xác định trong thư mục này có những thứ gì, bởi vì hầu hết các file nhị phân cần cho Linux đều được lưu trữ ở đây, trong đó đáng chú ý là thư mục con /usr/src bao gồm các thư mục con chứa các chương trình nguồn của nhân Linux.

Thư mục /home

Thư mục này chứa các thư mục cá nhân của người dùng: mỗi người dùng tương ứng với một thư mục con ở đây, tên người dùng được lấy làm tên của thư mục con.

Thư mục /var

Thư mục /var được sử dụng để lưu trữ các file chứa các thông tin luôn luôn thay đổi, bao gồm bộ đệm in, vùng lưu tạm thời cho việc nhận và gửi thư (mail), các khóa tiến trình, v.v..

Thư mục /boot

Là thư mục chứa nhân của hệ thống (Linux-*.*.), System.map (file ánh xạ đến các driver để nạp các hệ thống file khác), ảnh (image) của hệ thống file dùng cho initrd (ramdisk), trình điều khiển cho các thiết bị RAID (một thiết bị gồm một mảng các ổ đĩa cứng để tăng tốc độ và độ an toàn khi ghi dữ liệu), các bản sao lưu boot record của các phân vùng đĩa khác. Thư mục này cho phép khởi động và nạp lại bất kỳ trình điều khiển nào được yêu cầu để đọc các hệ thống file khác.

Thư mục /proc

Đây là thư mục dành cho nhân (**kernel**) của hệ điều hành và thực tế đây là một hệ thống file độc lập do nhân khởi tạo.

Thư mục /misc và thư mục /opt

Cho phép lưu trữ mọi đối tượng vào hai thư mục này.

Thư mục /sbin

Thư mục lưu giữ các file hệ thống thường tự động chạy.

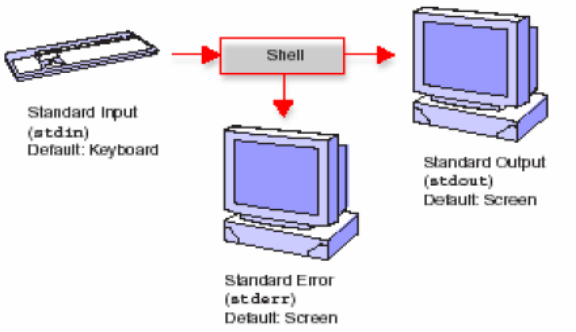
5. Các file chuẩn vào /ra trên Linux

Khi chạy chương trình Linux, nó giao tiếp với chúng ta qua việc hiển thị thông tin ra màn hình. Thông tin hiển thị màn hình có thể là dữ liệu của chương trình hay lỗi phát sinh khi có lỗi xảy ra. Chúng ta giao tiếp với chương trình qua các kí tự gõ vào bàn phím. Luồng dữ liệu vào từ bàn phím gọi là chuẩn input. Luồng dữ liệu ra màn hình gọi là chuẩn output còn luồng dữ liệu thông báo lỗi là chuẩn error.

Trong Linux các luồng giao tiếp chuẩn được xem như các file dữ liệu và được đánh số theo thứ tự, khi cho một file chạy, Shell tự động mở 3 file vào/ra chuẩn:

- Vào chuẩn (stdin) fd = 0.
- Ra chuẩn (stdout) fd = 1.
- Lỗi chuẩn(stderror) fd = 2.

Các số fd này được gọi là file descriptor.



Hình 2.2 Các chuẩn vào ra

Ví dụ về các file ra vào chuẩn: Sử dụng chương trình cat để soạn thảo, chúng ta gõ

```
$ cat  <enter>
du lieu vao tu ban phim <enter>
dong du lieu thu hai
```

Để kết thúc luồng dữ liệu vào chúng ta gõ **<Ctrl + d>**. Tất cả các dữ liệu chúng ta đưa vào từ bàn phím được xem là file input chuẩn. Dùng lệnh **ls** chúng ta sẽ nhận được dữ liệu ra màn hình, đó là file output chuẩn.

Một thông báo lỗi xuất hiện ở màn hình khi chúng ta gõ lệnh sai hoặc truy xuất vào các tập tin hay thư mục không có quyền chính là file error chuẩn. Ví dụ như chúng ta gõ lệnh listn thì sẽ xuất hiện lỗi invalid command.

Chuyển tiếp (redirection)

Chuyển tiếp là hình thức thay đổi luồng dữ liệu của các input, output chuẩn và error. Khi dùng chuyển tiếp, input chuẩn có thể lấy dữ liệu từ file thay vì bàn phím, output chuẩn hoặc error có thể chuyển vào tập tin hay ra máy in.

Có 3 loại chuyển hướng :

- **Input** redirection.
- **Output** redirection.
- **Error** redirection.

Input direction

Theo qui ước thì các lệnh lấy dữ liệu từ input chuẩn (bàn phím). Để lệnh lấy dữ liệu từ file chúng ta dùng ký hiệu < :

```
$lệnh < input
```

Ta có thể hình dung < chỉ hướng dữ liệu.

Ví dụ:

```
$cat < abc.txt  hoặc
$cat  0< abc.txt
```

Output redirection

Dữ liệu ra của các lệnh thông thường được hiển thị trên màn hình. để dữ liệu ra được đưa vào file chúng ta dùng dấu >

```
$lệnh > tên_file
```

Ví dụ Liệt kê nội dung thư mục và chuyển vào file : **$ls –l > tm.txt**

Để thêm vào dữ liệu có sẵn trên file, chúng ta dùng dấu >> thay cho dấu >

```
$lệnh >> tên_file hoặc $cat a.txt >> sum.txt
```

CHƯƠNG 3: THAO TÁC TRÊN HỆ THỐNG FILE CỦA UNIX

1. Quản lý quyền thâm nhập hệ thống file

Mỗi file và thư mục trong Linux đều có một chủ sở hữu và một nhóm sở hữu, cũng như một tập hợp các quyền thâm nhập (truy cập). Cho phép thay đổi các quyền thâm nhập và quyền sở hữu file và thư mục nhằm cung cấp thâm nhập nhiều hơn hay ít hơn.

Người sử dụng

Một người sử dụng được mô tả bằng các thông tin sau:

- Tên.
- [mật khẩu (nếu có].
- số nhận dạng (uid : user identify number).
- số của nhóm (gid : group identify number).
- [chú thích].
- thư mục tiếp nhận (HOME directory).
- [tên chương trình cho chạy lúc bắt đầu tên làm việc].

Các thông tin trên được chứa trong file /etc/passwd.

```
mail:x:8:12:mail:/var/spool/mail:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
duonglk:x:500:0:Le Khanh Duong:/home/duonglk:/bin/bash
anhth:x:17:100:Tran Hong Anh:/home/anhth:/bin/bash
```

Nhóm người dùng

Một nhóm người sử dụng là tập hợp của một số người sử dụng có thể dùng chung các file của nhau. Một nhóm người sử dụng được mô tả bằng các thông tin sau:

- tên của nhóm.
- [mật khẩu].
- số của nhóm (gid : group identify number).
- [danh sách những người khách (guest)].

Các thông tin trên được chứa trong file /etc/group.

Do Unix là một hệ điều hành đa người dùng và đa nhiệm, nhiều người cùng có thể sử dụng một máy Unix và một người có thể cho chạy nhiều chương trình khác nhau.

Có hai vấn đề lớn được đặt ra: quyền sở hữu các dữ liệu trên đĩa và phân chia tài nguyên hệ thống như CPU, RAM ... giữa các tiến trình. Chúng ta sẽ bàn về sở hữu các tập tin và các quyền truy xuất tập tin.

Tất cả các tập tin và thư mục của Linux đều có người sở hữu và quyền truy nhập. Ta có thể đổi các tính chất này cho phép nhiều hay ít quyền truy nhập hơn đối với một tập tin hay thư mục.

Quyền của tập tin còn cho phép xác định tập tin có là một chương trình (application) hay không (khác với MSDOS và MSWindows xác định tính chất này qua phần mở rộng của tên tập tin)

Thuộc tính thâm nhập file bao gồm các thuộc tính: *Đọc (R), Ghi (W), thực thi (X)*. ẩ hư vậy, một file có 9 thuộc tính thâm nhập ngoài ra có thêm thuộc tính chỉ định nó là file hay thư mục.

1	2	3	4	5	6	7	8	9	10

1 : kiểu file

2, 3, 4 : quyền thâm nhập của USER

5, 6, 7 : quyền thâm nhập của GROUP

8, 9, 10: quyền thâm nhập của OTHER

Có một số kiểu file trong Linux. Ký tự đầu tiên mô tả kiểu file và quyền thâm nhập sẽ cho biết file thuộc kiểu nào (chữ cái đó được gọi là chữ cái biểu diễn).

Chữ cái biểu diễn	*Kiểu file*
d	Thư mục (directory)
b	File kiểu khối (block-type special file)
c	File kiểu ký tự (character-type special file)
l	Liên kết tượng trưng (symbolic link)
p	File đường ống (pipe)
s	Socket
-	File bình thường (regular file)

Trong mỗi nhóm quyền thâm nhập có 3 thuộc tính: (R) được đọc, (W) được ghi, (X) được thực thi, (-) rỗng.

```
- r w -r--r--1 van_a group 166 Oct 4 08:02 thu.txt
-               :    chỉ rằng đây là File
r w -           :    USER có quyền đọc ghi
r -  -          :    GROUP có quyền đọc
r - -           :    OTHER có quyền đọc
1               :    số liên kết
van_a           :    tên người sở hữu
group           :    tên nhóm sử dụng
```

```
166             :     độ dài file
Oct 4 08: 02    :     thời gian tạo file
Thu.txt         :     tên file
```

Để hiểu được chính xác quyền thâm nhập có ý nghĩa như thế nào đối với hệ thống máy tính, phải nhớ rằng *Linux xem mọi thứ đều là file*. ả ếu cài đặt một ứng dụng, nó cũng sẽ được xem như mọi chương trình khác, trừ một điều: hệ thống nhận biết rằng một ứng dụng là một chương trình khả thi, tức là nó có thể chạy được. Một bức thư gửi cho mẹ là một dạng file văn bản bình thường, nhưng nếu thông báo cho hệ thống biết đó là một chương trình khả thi, hệ thống sẽ cố để chạy chương trình (và tất nhiên là lỗi).

Quyền đọc: cho phép người dùng có thể xem nội dung của file với rất nhiều chương trình khác nhau, nhưng họ sẽ không thể thay đổi, sửa chữa hoặc xóa bất kỳ thông tin nào trong đó. Tuy nhiên, họ có thể sao chép file đó thành file của họ và sửa chữa file bản sao.

Quyền ghi: là quyền thâm nhập tiếp theo. ả gười sử dụng với quyền ghi khi truy nhập vào file có thể thêm thông tin vào file. ả ếu có quyền ghi và quyền đọc đối với một file, có thể soạn thảo lại file đó - quyền đọc cho phép xem nội dung, và quyền ghi cho phép thay đổi nội dung file. ả ếu chỉ có quyền ghi, sẽ thêm được thông tin vào file, nhưng lại không thể xem được nội dung của file.

Quyền thực hiện hay thực thi: quyền này cho phép người dùng có thể chạy được file, nếu đó là một chương trình khả thi. Quyền thực hiện độc lập với các quyền truy nhập khác, vì thế hoàn toàn có thể có một chương trình với quyền đọc và quyền thực hiện, nhưng không có quyền ghi. Cũng có trường hợp một chương trình chỉ có quyền thực hiện, có nghĩa là người dùng có thể chạy ứng dụng, nhưng họ không thể xem được cách nó làm việc hay sao chép nó.

Quyền thâm nhập	*Ý nghĩa*
---	Không cho phép một quyền truy nhập nào
r--	Chỉ được quyền đọc
r-x	Quyền đọc và thực hiện (cho chương trình và shell script)
rw-	Quyền đọc và ghi
rwx	Cho phép tất cả các quyền truy nhập (cho chương trình)

Song song với cách ký hiệu miêu tả bằng ký tự như ở trên, quyền thâm nhập tập tin còn có thể cho dưới dạng chữ số hệ 8. Đối với file ***thu.txt*** ở trên có quyền là 644.

Điều quan trọng là phải hiểu cách ký hiệu bằng số vì nó liên quan đến việc thay đổi các quyền sau này. Các số có thể nhận tất cả các giá trị từ 0 đến 7.

Số đầu tiên miêu tả quyền của USER, số thứ hai cho GROUP và số thứ ba cho OTHER.

Mỗi số là tổng của các quyền theo quy tắc sau :

read permission (QUYỀẩ ĐỌC)	4
Write permission (QUYỀẩ GHI)	2
Execute permission (QUYỀẩ THỰC THI)	1

Vì vậy, một tập tin với quyền 751 có nghĩa là USER có quyền read, write, và execute bằng 4+2+1=7, GROUP có quyền read và execute bằng 4+1=5, và OTHER có quyền execute bằng 1.

ẩ ếu ta xem kỹ, ta sẽ thấy mọi số từ 0 đến 7 đều tương ứng với một tổ hợp duy nhất các quyền thâm nhập tập tin.

Quyền	*Chữ số hệ 8*	*Quyền*	*Chữ số hệ 8*
Chỉ đọc	4	Chỉ đọc và ghi	6
Chỉ ghi	2	Chỉ đọc và thực hiện	5
Chỉ thực hiện	1	Chỉ ghi và thực hiện	3
Không có quyền nào	0	Đọc, ghi và thực hiện	7

ẩ ếu ta quen với hệ nhị phân, hãy suy nghĩ bằng hệ thống nhị phân. Khi đó, rwx sẽ như số nhị phân 3 bit. ẩ ếu quyền được cho, số nhị phân tương ứng sẽ bằng 1, ngược lại, nó sẽ bằng 0. Ví dụ r-x sẽ là số nhị phân 101, và theo hệ thập phân sẽ là 4+0+1, hay 5. --x sẽ tương ứng 001, hay 0+0+1 = 1 …

2. Nhóm lệnh quản lý quyền thâm nhập file

ẩ hóm lệnh chown, chgrp và chmod được sử dụng rất phổ biến, cho phép thay quyền thâm nhập của tập tin hay thư mục. Chỉ có chủ sở hữu và superuser mới có quyền thực hiện các lệnh này.

2.1 Lệnh chmod

Cho phép thay đổi quyền thâm nhập các file và thư mục. Có thể chạy lệnh theo 2 cách:

Theo thông số tuyệt đối

```
chmod      mode       tên_file
```

trong đó thông số mode là một số cơ số 8 (octal)

```
r w x           r - x           r - -
1 1 1           1 0 1           1 0 0
7               5               4
$chmod 754 tên_file
```

Cách thay đổi tuyệt đối này có một số ưu điểm vì nó là cách định quyền tuyệt đối, kết quả cuối cùng không phụ thuộc vào quyền thâm nhập trước đó của tập tin. Đồng thời, dễ nói "thay quyền tập tin thành bảy-năm-năm" thì dễ hơn là "thay quyền tập tin thành đọc-viết-thực hiện, đọc-thực hiện, đọc-thực hiện"

Dùng các ký hiệu tượng trưng

Ta cũng có thể thay đổi quyền truy nhập một cách tương đối và dễ nhớ. Để chỉ ra nhóm quyền nào cần thay đổi, ta có thể sử dụng u (user), g (group), o (other), hay a (all). Tiếp theo đó là dấu + để thêm quyền và – để bớt quyền. Cuối cùng là bản thân các quyền viết tắt bởi r,w,x.

Ví dụ như để bổ sung quyền thực hiện cho group và other, ta nhập vào dòng lệnh:

```
chmod           who   [operation]     [right]     filename
who        :    u có nghĩa user
                g group
                o other
                a all
operation:
                + thêm quyền
                - bớt quyền
                = gán giá trị khác
right:
                r reading
                w writing
                x execution
                s đặt suid hoặc guid
```

Ví dụ: **$ chmod go+x tenfile**

Đây là cách thay đổi tương đối vì kết quả cuối cùng phụ thuộc vào quyền đã có trước đó mà lệnh này không liên quan đến. Trên quan điểm bảo mãt hệ thống, cách thay đổi tuyệt đối dẫn đến ít sai sót hơn.

Thay đổi quyền thâm nhập của một thư mục cũng được thực hiện giống như đối với một tập tin. Chú ý là nếu ta không có quyền thực hiện (execute) đối với một thư mục, ta không thể thay đổi thư mục *cd* vào thư mục đó. Mọi người sử dụng có quyền viết vào thư mục đều có quyền xóa tập tin trong thư mục đó, không phụ thuộc vào quyền của người đó đối với tập tin.

Vì vậy, đa số các thư mục có quyền **drwxr-xr-x**. ẩ hư vậy chỉ có người sở hữu của thư mục mới có quyền tạo và xóa tập tin trong thư mục. ẩ goài ra, thư mục còn có một quyền đặc biệt, đó là cho phép mọi người đều có quyền tạo tập tin trong thư mục, mọi người đều có quyền thay đổi nội dung tập tin trong thư mục, nhưng chỉ có người tạo ra mới có quyền xóa tập tin. Đó là sticky bit (bít đính kèm) cho thư mục. Thư mục /tmp thường có sticky bit bật lên

```
drwxrwxrw   7 root     root        16384 Oct 21 15:33 tmp
```

2.2 Lệnh chown

Để thay đổi quyền sở hữu đối với một file, hãy sử dụng lệnh **chown** với cú pháp như sau:

```
chown [tùy chọn] [chủ][.nhóm] <file ...>
```

Lệnh này cho phép thay chủ sở hữu file. ẩ ếu chỉ có tham số về chủ, thì người dùng chủ sẽ có quyền sở hữu file và nhóm sở hữu không thay đổi.

ẩ ếu theo sau tên người chủ là dấu "." và tên của một nhóm thì nhóm đó sẽ nhóm sở hữu file.

ẩ ếu chỉ có dấu "." và nhóm mà không có tên người chủ thì chỉ có quyền sở hữu nhóm của file thay đổi, lúc này, lệnh chown có tác dụng giống như lệnh chgrp (lệnh chgrp được trình bày dưới đây).

Các tùy chọn của lệnh *chown*:

- c, --changes : hiển thị dòng thông báo chỉ với các file mà lệnh làm thay đổi sở hữu (số thông báo hiện ra có thể ít hơn trường hợp -v, -verbosr).
- f, --silent, --quiet : bỏ qua hầu hết các thông báo lỗi.
- R, --recursive : thực hiện đổi quyền sở hữu đối với thư mục và file theo đệ quy.
- v, --verbose : hiển thị dòng thông báo với mọi file liên quan mà chown tác động tới (có hoặc không thay đổi sở hữu).
- - help : đưa ra trang trợ giúp và thoát.

Formatted: Bullets and Numbering

Ví dụ, thư mục vidu có thông tin về các quyền truy nhập như sau:

```
drwxr-xr-x 11 duonglk root 4000 Oct 21 2008 vidu
```

ẩ gười sở hữu hiện tại thư mục vidu là người dùng duonglk. Để người dùng anhth là chủ sở hữu thư mục trên, hãy gõ lệnh: ***# chown anhth vidu***

Khi chuyển quyền sở hữu file cho một người khác, người chủ cũ mất quyền sở hữu file đó.

2.3 Lệnh chgrp

Các file (và người dùng) còn thuộc vào các nhóm, đây là phương thức truy nhập file thuận tiện cho nhiều người dùng nhưng không phải tất cả người dùng trên hệ thống. Khi đăng

nhập, mặc định sẽ là thành viên của một nhóm được thiết lập khi người dùng root tạo tài khoản người dùng. Cho phép một người dùng thuộc nhiều nhóm khác nhau, nhưng mỗi lần đăng nhập chỉ là thành viên của một nhóm.

Để thay đổi quyền sở hữu nhóm đối với một hoặc nhiều file, hãy sử dụng lệnh chgrp với cú pháp như sau:

```
chgrp [tùy-chọn] {nhóm|--reference=nhómR} <file...>
```

Lệnh này cho phép thay thuộc tính nhóm sở hữu của file theo tên nhóm được chỉ ra trực tiếp theo tham số nhóm hoặc gián tiếp qua thuộc tính nhóm của file có tên là nhómR.

Các tùy chọn của lệnh là (một số tương tự như ở lệnh chown):

- c, --changes : hiển thị dòng thông báo chỉ với các file mà lệnh làm thay đổi sở hữu (số thông báo hiện ra có thể ít hơn trường hợp -v, -verbosr).
- f, --silent, --quiet : bỏ qua hầu hết các thông báo lỗi.
- R, --recursive : thực hiện đổi quyền sở hữu đối với thư mục và file theo đệ quy.
- v, --verbose : hiển thị dòng thông báo với mọi file liên quan mà chgrp tác động tới (có hoặc không thay đổi sở hữu).
- - help : hiển thị trang trợ giúp và thoát

Tham số --reference=nhómR cho thấy cách gián tiếp thay nhóm chủ của file theo nhóm chủ của một file khác (tên là nhómR) là cách thức được ưa chuộng hơn.

Ví dụ: Cho phép thay đổi nhóm sở hữu.

```
$echo Hello > file1
$chmod 700 file1
$ls -l file1
-rwx------ 1 user1 stagiair 6 Apr 5 14:06 file1
$cat file1
Hello
$chgrp animator file1
$ls -l file1
-rwx------ 1 user1 animator 6 Apr 5 14:06 file1
$chown user2 file1
$ls -l file1
-rwx------ 1 user2 animator 6 Apr 5 14:06 file1
$cat file1
cat: cannot open file1
```

Formatted: Bullets and Numbering

3. Các lệnh thao tác trên thư mục

3.1 Thay đổi thư mục làm việc hiện thời với lệnh cd

Cú pháp lệnh: **cd**

Chuyển đến thư mục /usr/include : $cd /usr/include

Chuyển trở lại thư mục "home": $cd

Chuyển đến thư mục cha: $cd..

3.2 Xem nội dung thư mục với lệnh ls

Sử dụng lệnh *ls* và một số các tùy chọn của nó là có thể biết được mọi thông tin về một thư mục.

Cú pháp lệnh: **# ls [tùy-chọn] [file]**

Lệnh này đưa ra danh sách các file liên quan đến tham số ***file*** trong lệnh. Trường hợp phổ biến tham số file là một thư mục, tuy nhiên trong một số trường hợp khác, tham số ***file*** xác định nhóm (khi sử dụng các mô tả nhóm *, ? và cặp [và])**;** nếu không có tham số ***file***, mặc định danh sách các file có trong thư mục hiện thời sẽ được hiển thị.

Các tùy chọn của lệnh:

- a : liệt kê tất cả các file, bao gồm cả file ẩn.
- l : đưa ra thông tin đầy đủ nhất về các file và thư mục.
- s : chỉ ra kích thước của file, tính theo khối (1 khối = 1204 byte).
- F : xác định kiểu file (/ = thư mục, * = chương trình khả thi).
- m : liệt kê các file được ngăn cách nhau bởi dấu ",".
- C : đưa ra danh sách các file và thư mục theo dạng cột (hai thư mục gần nhau được xếp vào một cột).
- 1 : hiển thị mỗi file hoặc thư mục trên một dòng.
- t : sắp xếp các file và thư mục trong danh sách theo thứ tự về thời gian được sửa đổi gần đây nhất.
- x : đưa ra danh sách các file và thư mục theo dạng cột (hai thư mục gần nhau được xếp trên hai dòng đầu của hai cột kề nhau).
- r : sắp xếp danh sách hiển thị theo thứ tự ngược lại.
- R : liệt kê lần lượt các thư mục và nội dung của các thư mục.

Ví dụ: khi gõ lệnh *ls [is]** cho danh sách các file và thư mục con có tên bắt đầu bằng hoặc chữ cái *i* hoặc chữ cái *s* có trong thư mục hiện thời:

```
id*  sed*  sh@   sha256sum*  shred*  sln*    stat*  sulogin@
install*   seq*  sha1sum*  sha384sum*  shuf*   sort*   stty*  sum*
```

Formatted: Bullets and Numbering

```
ipmask*  setterm*  sha224sum* sha512sum*  sleep*  split*  su*  sync*
```

3.3 Tạo thư mục với lệnh mkdir

Lệnh *mkdir* tạo một thư mục, cú pháp: **mkdir [tùy-chọn] <thư-mục>**

Lệnh này cho phép tạo một thư mục mới nếu thư mục đó chưa thực sự tồn tại. Để tạo một thư mục, cần đặc tả tên và vị trí của nó trên hệ thống file (vị trí mặc định là thư mục hiện thời). ả ếu thư mục đã tồn tại, hệ thống sẽ thông báo cho biết.

Các tùy chọn:

- m, --mode=Mod : thiết lập quyền truy nhập Mod như trong lệnh chmod nhưng không cho quyền rwxrwxrwx.
- p, --parents : tạo các thư mục cần thiết mà không thông báo lỗi khi nó đã tồn tại.
- - verbose : hiển thị các thông báo cho mỗi thư mục được tạo.
- - help : đưa ra trang trợ giúp và thoát.

ả ếu muốn tạo thư mục có khoảng cách giữa các từ ta phải sử dụng dấu " ". ả ếu muốn tạo thư mục My Documents ta sử dụng lệnh: *mkdir " My Documents"*

Ví dụ: nếu muốn tạo thư mục test trong thư mục home, hãy gõ lệnh sau: *mkdir /home/test*

3.4 Xóa thư mục với lệnh rmdir

Lệnh rmdir được dùng để xóa bỏ một thư mục.

Cú pháp lệnh: **rmdir [tùy-chọn[<thư-mục>**

Có thể xóa bỏ bất kỳ thư mục nào nếu có quyền đó. Lưu ý rằng, thư mục chỉ bị xóa khi nó "rỗng", tức là không tồn tại file hay thư mục con nào trong đó.

Không có cách gì khôi phục lại các thư mục đã bị xóa, vì thế hãy suy nghĩ cẩn thận trước khi quyết định xóa một thư mục.

Các tùy chọn của lệnh:

- - ignore-fail-on-non-empty : bỏ qua các lỗi nếu xóa một thư mục không rỗng.
- p, --parents : xóa bỏ một thư mục, sau đó lần lượt xóa bỏ tiếp các thư mục có trên đường dẫn chứa thư mục vừa xóa. Ví dụ, dòng lệnh rmdir -p /a/b/c sẽ tương đương với ba dòng lệnh rmdir /a/b/c, rmdir /a/b, rmdir /a (với điều kiện các thư mục là rỗng).
- - verbose : đưa ra thông báo khi xóa một thư mục.
- - help : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# rmdir -p /test/test1/test2
rmdir: /: No such file or directory
```

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Dòng lệnh trên sẽ lần lượt xóa ba thư mục test2, test1, test và hiển thị thông báo trên màn hình kết quả của lệnh.

3.5 Xem đường dẫn thư mục hiện thời với lệnh pwd

Cú pháp lệnh: **pwd**

Lệnh này cho biết hiện người dùng đang ở trong thư mục nào và hiện ra theo dạng một đường dẫn tuyệt đối.

Ví dụ: gõ lệnh pwd tại dấu nhắc lệnh sau khi người dùng duonglk vừa đăng nhập thì màn hình hiển thị như sau:

```
# pwd
/home/duonglk
```

3.6 Lệnh đổi tên thư mục với lệnh mv

Cú pháp lệnh: **mv <tên-cũ> <tên-mới>**

Lệnh này cho phép đổi tên một thư mục từ tên-cũ thành tên-mới.

Ví dụ: **# mv Tongket thongke** sẽ đổi tên thư mục *Tongket* thành *thongke* .

ả ếu sử dụng lệnh *mv* để đổi tên một thư mục với một cái tên đã được đặt cho một file thì lệnh sẽ gặp lỗi. ả ếu tên mới trùng với tên một thư mục đang tồn tại thì nội dung của thư mục được đổi tên sẽ ghi đè lên nội dung của thư mục trùng tên.

4. Các lệnh thao tác trên file

4.1 Tạo file với lệnh touch

Lệnh touch có nhiều chức năng, trong đó một chức năng là giúp tạo file mới trên hệ thống: *touch* rất hữu ích cho việc tổ chức một tập hợp các file mới.

Cú pháp lệnh: **touch <file>**

Thực chất lệnh này có tác dụng dùng để cập nhật thời gian truy nhập và sửa chữa lần cuối của một file. Vì lý do này, các file được tạo bằng lệnh touch đều được sắp xếp theo thời gian sửa đổi. ả ếu sử dụng lệnh touch đối với một file chưa tồn tại, chương trình sẽ tạo ra file đó. Sử dụng bất kỳ trình soạn thảo nào để soạn thảo file mới.

Ví dụ: dùng lệnh touch để tạo file newfile: *# touch newfile*

4.2 Tạo file với lệnh cat

Lệnh *cat* tuy đơn giản nhưng rất hữu dụng trong Linux. Chúng ta có thể sử dụng lệnh này để lấy thông tin từ đầu vào (bàn phím...) rồi kết xuất ra file hoặc các nguồn khác, hay để xem nội dung của một file ... Phần này trình bày tác dụng của lệnh cat đối với việc tạo file.

Cú pháp lệnh: **cat > filename**

Theo ngầm định, lệnh này cho phép lấy thông tin đầu vào từ bàn phím rồi xuất ra màn hình. Soạn thảo nội dung của một file bằng lệnh cat tức là đã đổi hướng đầu ra của lệnh từ màn hình vào một file. ẩ gười dùng gõ nội dung của file ngay tại dấu nhắc màn hình và gõ *CTRL+d* để kết thúc việc soạn thảo.

ẩ hược điểm của cách tạo file này là nó không cho phép sửa lỗi, ví dụ nếu muốn sửa một lỗi chính tả trên một dòng, chỉ có cách là xóa đến vị trí của lỗi và gõ lại nội dung vừa bị xóa.

Ví dụ: tạo file newfile trong thư mục /home/vd bằng lệnh cat.

```
# cat > /home/vd/newfile
This is a example of cat command
^D
```

Sau khi soạn thảo xong, gõ Enter và CTRL+d để trở về dấu nhắc lệnh, nếu không gõ Enter thì phải gõ CTRL+d hai lần. Khi sử dụng lệnh này, nếu file chưa tồn tại thì sẽ tạo file mới, nếu file đó đã tồn tại thì sẽ xóa file cũ và tạo file mới. Có thể sử dụng luôn lệnh *cat* để xem nội dung của file vừa soạn thảo:

```
# cat /home/vd/newfile
This is a example of cat command
```

Để thêm nội dung vào phần cuối của file có sẵn dùng lệnh: **cat >> filename.**

Để tổng hợp hai tập tin thành một ta sử dụng cú pháp lệnh sau: **$cat file1 file2 > file3**

4.3 Xem nội dung các file lớn với lệnh more

Lệnh *cat* cho phép xem nội dung của một file, nhưng nếu file quá lớn, nội dung file sẽ trôi trên màn hình và chỉ có thể nhìn thấy phần cuối của file. Linux có một lệnh cho phép có thể xem nội dung của một file lớn theo từng trang màn hình, đó là lệnh more.

Cú pháp lệnh: **more [-tùy chọn] [-số] [+/xâumẫu] [+dòng-số] [file ...]**

Các tùy chọn:

- _____số: xác định số dòng nội dung của file được hiển thị (số).
- _____d: trên màn hình sẽ hiển thị các thông báo giúp người dùng cách sử dụng đối với lệnh more, ví như [Press space to continue, "q" to quit .], hay hiển thị [Press "h" for instructions .] thay thế cho tiếng chuông cảnh báo khi bấm sai một phím.
- _____l: *more* thường xem ^L là một ký tự đặc biệt, nếu không có tùy chọn này, lệnh sẽ dừng tại dòng đầu tiên có chứa ^L và hiển thị % nội dung đã xem được (^L không bị mất), nhấn phím space (hoặc enter) để tiếp tục. ẩ ếu có tùy chọn -l, nội dung của file sẽ được hiển thị như bình thường nhưng ở một khuôn dạng khác, tức là dấu ^L sẽ mất và trước dòng có chứa ^L sẽ có thêm một dòng trống.

Formatted: Bullets and Numbering

- _____p: không cuộn màn hình, thay vào đó là xóa những gì có trên màn hình và hiển thị tiếp nội dung file.
- _____c: không cuộn màn hình, thay vào đó xóa màn hình và hiển thị nội dung file bắt đầu từ đỉnh màn hình.
- _____s: xóa bớt các dòng trống liền nhau trong nội dung file chỉ giữ lại một dòng.
- _____u: bỏ qua dấu gạch chân.
- +/xâumẫu : tùy chọn +/xâumẫu chỉ ra một chuỗi sẽ được tìm kiếm trước khi hiển thị mỗi file.
- +dòng-số : bắt đầu hiển thị từ dòng thứ dòng-số.

Ví dụ:

```
# more -d vdmore
total 1424
drwxr-xr-x 6 root root 4096 Oct 31 2000 AfterStep-1.8.0
drwxr-xr-x 2 root root 4096 Oct 31 2000 AnotherLevel
drwxr-xr-x 2 root root 4096 Oct 31 2000 ElectricFence
drwxr-xr-x 2 root root 4096 Oct 31 2000 GXedit-1.23
drwxr-xr-x 3 root root 4096 Oct 31 2000 HTML
drwxr-xr-x 3 root root 4096 Oct 31 2000 ImageMagick
drwxr-xr-x 6 root root 4096 Oct 31 2000 LDP
drwxr-xr-x 3 root root 4096 Oct 31 2000 ORBit-0.5.0
drwxr-xr-x 2 root root 4096 Oct 31 2000 SVGATextMode
drwxr-xr-x 2 root root 4096 Oct 31 2000 SysVinit-2.78
drwxr-xr-x 2 root root 4096 Oct 31 2000 WindowMaker
--More--(9%) [ Press space to continue, "q" to quit .]
```

Đối với lệnh more, có thể sử dụng một số các phím tắt để thực hiện một số các thao tác đơn giản trong khi đang thực hiện lệnh. Bảng dưới đây liệt kê các phím tắt đó:

Phím tắt	*Chức năng*
[Space]	ấ hấn phím space để hiển thị màn hình tiếp theo
n	Hiển thị n dòng tiếp theo
[Enter]	Hiển thị dòng tiếp theo
h	Hiển thị danh sách các phím tắt
d hoặc CTRL+D	Cuộn màn hình (mặc định là 11 dòng)
q hoặc CTRL+Q	Thoát khỏi lệnh more
s	Bỏ qua n dòng (mặc định là 1)

f	Bỏ qua k màn hình tiếp theo (mặc định là 1)
b hoặc CTRL+B	Trở lại k màn hình trước (mặc định là 1)
=	Hiển thị số dòng hiện thời
:n	xem k file tiếp theo
:p	Trở lại k file trước
v	Chạy chương trình soạn thảo vi tại dòng hiện thời
CTRL+L	Vẽ lại màn hình
:f	Hiển thị tên file hiện thời và số dòng
.	Lặp lại lệnh trước

4.4 Thêm số thứ tự của các dòng trong file với lệnh nl

ả hư đã biết lệnh *cat* với tham số -n sẽ đánh số thứ tự của các dòng trong file, tuy nhiên Linux còn cho phép dùng lệnh nl để thực hiện công việc như vậy.

Cú pháp lệnh: **nl [tùy-chọn] <file>**

Lệnh này sẽ đưa nội dung file ra thiết bị ra chuẩn, với số thứ tự của dòng được thêm vào. ả ếu không có file (tên file), hoặc khi file là dấu "-", thì đọc nội dung từ thiết bị vào chuẩn.

Các tuỳ chọn:

- b, --body-numbering=STYLE: sử dụng kiểu STYLE cho việc đánh thứ tự các dòng trong nội dung file. Có các kiểu STYLE sau:
  - a : đánh số tất cả các dòng kể cả dòng trống;
  - t : chỉ đánh số các dòng không trống;
  - n : không đánh số dòng.
- d, --section-delimiter=CC : sử dụng CC để đánh số trang logic (CC là hai ký tự xác định phạm vi cho việc phân trang logic).
- f, --footer-numbering=STYLE : sử dụng kiểu STYLE để đánh số các dòng trong nội dung file (một câu có thể có hai dòng ...).
- h, --header-numbering=STYLE : sử dụng kiểu STYLE để đánh số các dòng trong nội dung file.
- i, --page-increment=số : đánh số thứ tự của dòng theo cấp số cộng có công sai là số.
- l, --join-blank-lines=số :nhóm số dòng trống vào thành một dòng trống.
- n, --number-format=khuôn: chèn số dòng theo khuôn (khuôn: ln - căn trái, không có số 0 ở đầu; rn - căn phải, không có số 0 ở đầu; rz - căn phải và có số 0 ở đầu).

Formatted: Bullets and Numbering

- _____p, --no-renumber : không thiết lập lại số dòng tại mỗi trang logic.
- _____s, --number-separator=xâu : thêm chuỗi xâu vào sau số thứ tự của dòng.
- _____v, --first-page=số : số dòng đầu tiên trên mỗi trang logic.
- _____w, --number-width=số : hiển thị số thứ tự của dòng trên cột thứ số.
- _____- help : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# cat > hello
noi dung trong file hello
noi dung trong file hello
^D
# nl --body-numbering=a --number-format=rz hello
000001 noi dung trong file hello
000002 noi dung trong file hello
```

Lệnh trong ví dụ trên cho thêm số thứ tự của các câu trong file *hello* theo dạng: đánh số thứ tự tất cả các dòng, kể cả dòng trống, các số thứ tự được căn phải và có số 0 ở đầu (lưu ý rằng có dòng trong file được hiện ra thành hai dòng trên giấy).

4.5 Xem nội dung file với lệnh head

Các đoạn trước cho biết cách thức xem nội dung của một file nhờ lệnh *cat* hay *more*. Trong Linux cũng có các lệnh khác cho nhiều cách thức để xem nội dung của một file. Trước hết, chúng ta hãy làm quen với lệnh *head*.

Cú pháp lệnh: **head [tùy-chọn] [filename]...**

Lệnh này mặc định sẽ đưa ra màn hình 10 dòng đầu tiên của mỗi file. ả ếu có nhiều hơn một file, thì lần lượt tên của file và 10 dòng nội dung đầu tiên sẽ được hiển thị. ả ếu không có tham số filename, hoặc filename là dấu "-", thì ngầm định sẽ đọc từ thiết bị vào chuẩn.

Các tuỳ chọn:

- _____c, --bytes=cỡ : hiển thị cỡ (số nguyên) ký tự đầu tiên trong nội dung file (cỡ có thể nhận giá trị là b cho 512, k cho 1K, m cho 1 Meg)
- _____n, --lines=n : hiển thị n (số nguyên) dòng thay cho 10 dòng ngầm định.
- _____q, --quiet, --silent : không đưa ra tên file ở dòng đầu.
- _____v, --verbose : luôn đưa ra tên file ở dòng đầu.
- _____- help : hiển thị trang trợ giúp và thoát.

Formatted: Bullets and Numbering

4.6 Xem nội dung file với lệnh tail

Lệnh thứ hai cho phép xem qua nội dung của file là lệnh *tail*.

Cú pháp lệnh: **tail [tùy-chọn] [file]...**

Lệnh tail ngầm định đưa ra màn hình 10 dòng cuối trong nội dung của các file. ả ếu có nhiều hơn một file, thì lần lượt tên của file và 10 dòng cuối sẽ được hiển thị. ả ếu không có tham số file, hoặc file là dấu "-" thì ngầm định sẽ đọc từ thiết bị vào chuẩn.

Các tùy chọn:

- _____- retry : cố gắng mở một file khó truy nhập khi bắt đầu thực hiện lệnh tail.
- _____c, --bytes=n : hiển thị n (số) ký tự sau cùng.
- _____f, --follow[={name | descritptor}] : sau khi hiện nội dung file sẽ hiện thông tin về file: -f, --follow, và --follow=descriptor là như nhau.
- _____n, --lines=n : hiển thị n (số) dòng cuối cùng của file thay cho 10 dòng ngầm định.
- _____- max-unchanged-stats=n : hiển thị tài liệu về file (ngầm định n là 5).
- _____- max-consecutive-size-changes=n : hiển thị tài liệu về file (ngầm định n là 200).
- _____- pid=PID : kết hợp với tùy chọn -f, chấm dứt sau khi quá trình có chỉ số = PID lỗi.
- _____q, --quiet, --silent : không đưa ra tên file ở dòng đầu trong nội dung được hiển thị.
- _____s, --sleep-interval=k : kết hợp với tùy chọn -f, dừng k giây giữa các hoạt động.
- _____v, --verbose : luôn hiển thị tên của file.
- _____- help : hiển thị trang trợ giúp và thoát.

4.7 Sử dụng lệnh file để xác định kiểu file

Cú pháp lệnh file: **file [tùy-chọn] [-f file] [-m <file-ảnh>...] <file>...**

Lệnh file cho phép xác định và in ra kiểu thông tin chứa trong file. Lệnh file sẽ lần lượt kiểm tra từ kiểu file hệ thống, kiểu file magic (ví dụ file mô tả thiết bị) rồi đến kiểu file văn bản thông thường.

ả ếu file được kiểm tra thỏa mãn một trong ba kiểu file trên thì kiểu file sẽ được in ra theo các dạng cơ bản sau:

- _____text: dạng file văn bản thông thường, chỉ chứa các mã ký tự ASCII.
- _____executable: dạng file nhị phân khả thi.
- _____data: thường là dạng file chứa mã nhị phân và không thể in ra được.

Một số tuỳ chọn sau đây:

- _____b : cho phép chỉ đưa ra kiểu file mà không đưa kèm theo tên file.
- _____f tên-file : cho phép hiển thị kiểu của các file có tên trùng với nội dung trên mỗi dòng trong file tên-file. Để kiểm tra trên thiết bị vào chuẩn, sử dụng dấu "-".
- _____z : xem kiểu của file nén.

Ghi chú: ả hớ rằng kết quả của lệnh *file* không phải lúc nào cũng chính xác tuyệt đối.

4.8 Lệnh wc dùng để đếm số ký tự, số từ, hay số dòng trong một file

Cú pháp lệnh: **wc [tùy-chọn] [file]...**

Lệnh hiện ra số lượng dòng, số lượng từ, số lượng ký tự có trong mỗi file, và một dòng tính tổng nếu có nhiều hơn một file được chỉ ra. ả ếu không có tùy chọn nào thì mặc định đưa ra cả số dòng, số từ và số ký tự. ả gầm định khi không có tên file trong lệnh thì sẽ đọc và đếm trên thiết bị vào chuẩn.

Các tuỳ chọn:

- _____c, --byte, --chars : đưa ra số ký tự trong file.
- _____l, --lines : đưa ra số dòng trong file.
- _____L, --max-line-length : đưa ra chiều dài của dòng dài nhất trong file.
- _____w, --words : đưa ra số từ trong file.
- _____- help : hiển thị trang trợ giúp và thoát.

Khi gõ lệnh *wc* mà không có một tham số nào, mặc định sẽ soạn thảo trực tiếp nội dung trên thiết bị vào chuẩn. Dùng CTRL+d để kết thúc việc soạn thảo, kết quả sẽ hiển thị lên màn hình.

- Bằng cách kết hợp lệnh *wc* với một số lệnh khác, có thể có nhiều cách để biết được những thông tin cần thiết.

Kết hợp với lệnh *ls* để xác định số file có trong một thư mục: **# ls | wc -l**

Kết hợp với lệnh *cat* để biết số tài khoản cá nhân có trên máy của người dùng:

```
# cat /etc/passwd | wc -l
```

4.9 So sánh nội dung hai file sử dụng lệnh diff

Việc tìm ra sự khác nhau giữa hai file đôi khi là rất cần thiết. Linux có một lệnh có tác dụng như vậy, đó là lệnh diff .

Cú pháp: **diff [tuỳ-chọn] <file1> <file2>**

Trong trường hợp đơn giản, lệnh *diff* sẽ so sánh nội dung của hai file. ả ếu file1 là một thư mục còn file2 là một file bình thường, diff sẽ so sánh file có tên trùng với file2 trong thư mục file1 với file2.

ả ếu cả file1 và file2 đều là thư mục, *diff* sẽ thực hiện sự so sánh lần lượt các file trong cả hai thư mục theo thứ tự từ a-z (sự so sánh này sẽ không đệ qui nếu tuỳ chọn *-r* hoặc *--recursive* không được đưa ra). Tất nhiên so sánh giữa hai thư mục không thể chính xác như khi so sánh hai file.

Các tuỳ chọn:

- _____a: xem tất cả các file ở dạng văn bản và so sánh theo từng dòng.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

- b: bỏ qua sự thay đổi về số lượng của ký tự trống.
- B: bỏ qua mọi sự thay đổi mà chỉ chèn hoặc xoá các dòng trống.
- brief: chỉ thông báo khi có sự khác nhau mà không đưa ra chi tiết nội dung khác nhau.
- d: tìm ra sự khác biệt nhỏ (tuỳ chọn này có thể làm chậm tốc độ làm việc của lệnh diff).
- exclude-from=file: khi so sánh thư mục, bỏ qua các file và các thư mục con có tên phù hợp với mẫu có trong file.
- i: so sánh không biệt chữ hoa chữ thường.
- r: thực hiện so sánh đệ qui trên thư mục.
- s: thông báo khi hai file là giống nhau.
- y: hiển thị hai file cạnh nhau để dễ phân biệt sự khác nhau.

4.10 Xóa file với lệnh rm

Lệnh *rm* là lệnh rất "nguy hiểm" vì trong Linux không có lệnh khôi phục lại những gì đã xóa, vì thế hãy cẩn trọng khi sử dụng lệnh này. Lệnh rm cho phép xóa bỏ một file hoặc nhiều file.

Cú pháp lệnh: **rm [tùy-chọn] <file> ...**

Các tùy chọn:

- d, --directory : loại bỏ liên kết của thư mục, kể cả thư mục không rỗng. Chỉ có siêu người dùng mới được phép dùng tùy chọn này.
- f, --force : bỏ qua các file (xác định qua tham số file) không tồn tại mà không cần nhắc nhở.
- i, --interactive : nhắc nhở trước khi xóa bỏ một file.
- r, -R, --recursive : xóa bỏ nội dung của thư mục một cách đệ quy.
- v, --verbose : đưa ra các thông báo về quá trình xóa file.
- help : hiển thị trang trợ giúp và thoát.

Lệnh *rm* cho phép xóa nhiều file cùng một lúc bằng cách chỉ ra tên của các file cần xóa trong dòng lệnh (hoặc dùng kí kiệu mô tả nhóm). Dùng lệnh # *rm bak/*.h* xóa mọi file với tên có hai kí hiệu cuối cùng là ".h" trong thư mục con bak.

4.11 Sao chép tập tin với lệnh cp

Lệnh cp có hai dạng như sau:

cp [tùy-chọn] <file-nguồn> ... <file-đích>

Formatted: Bullets and Numbering

cp [tùy-chọn] --target-directory=<thư-mục> <file-nguồn>...

Lệnh này cho phép sao file-nguồn thành file-đích hoặc sao chép từ nhiều file-nguồn vào một thư mục đích (tham số <file-đích> hay <thư-mục>). Dạng thứ hai là một cách viết khác đổi thứ tự hai tham số vị trí.

Các tùy chọn:

- a, --archive : giống như -dpR (tổ hợp ba tham số -d, -p, -R, như dưới đây).
- b, --backup[=COẩ TROL] : tạo file lưu cho mỗi file đích nếu như nó đang tồn tại.
- d, --no-dereference : duy trì các liên kết.
- f, --force : ghi đè file đích đang tồn tại mà không nhắc nhở.
- i, --interactive : có thông báo nhắc nhở trước khi ghi đè.
- l, --link : chỉ tạo liên kết giữa file-đích từ file-nguồn mà không sao chép.
- p, --preserve : duy trì các thuộc tính của file-nguồn sang file-đích.
- r : cho phép sao chép một cách đệ quy file thông thường.
- R : cho phép sao chép một cách đệ quy thư mục.
- s, --symbolic-link : tạo liên kết tượng trưng thay cho việc sao chép các file.
- S, --suffix=<hậu-tố> : bỏ qua các hậu tố thông thường (hoặc được chỉ ra).
- u, --update : chỉ sao chép khi file nguồn mới hơn file đích hoặc khi file đích chưa có.
- v, --verbose : đưa ra thông báo về quá trình sao chép.
- - help : hiển thị trang trợ giúp và thoát.

File đích được tạo ra có cùng kích thước và các quyền truy nhập như file nguồn, tuy nhiên file đích có thời gian tạo lập là thời điểm thực hiện lệnh nên các thuộc tính thời gian sẽ khác.

Ví dụ:

```
# cp /home/ftp/vd /home/test/vd1
```

ẩ ếu ở vị trí đích, mô tả đầy đủ tên file đích thì nội dung file nguồn sẽ được sao chép sang file đích. Trong trường hợp chỉ đưa ra vị trí file đích được đặt trong thư mục nào thì tên của file nguồn sẽ là tên của file đích.

```
# cp /home/ftp/vd /home/test/
```

Trong ví dụ này, tên file đích sẽ là *vd* nghĩa là tạo một file mới */home/test/vd*.

ẩ ếu sử dụng lệnh này để sao một thư mục, sẽ có một thông báo được đưa ra cho biết nguồn là một thư mục và vì vậy không thể dùng lệnh *cp* để sao chép.

```
# cp . newdir
cp: .: omitting directory
```


Formatted: Bullets and Numbering

Ví dụ: về việc lệnh *cp* cho phép sao nhiều file cùng một lúc vào một thư mục.

```
# cp vd vd1 newdir
# pwd
/newdir
# ls -l
total 8
-rw-r--r-- 1 root ftp 15 Nov 14 11:00 vd
-rw-r--r-- 1 root ftp 12 Nov 14 11:00 vd1
```

Đối với nhiều lệnh làm việc với file, khi gõ lệnh có thể sử dụng kí hiệu mô tả nhóm để xác định một nhóm file làm cho tăng hiệu lực của các lệnh đó. Ví dụ, lệnh: **# cp * bak** thực hiện việc sao chép mọi file có trong thư mục hiện thời sang thư mục con của nó có tên là *bak*.

Dùng lệnh: **# cp /usr/src/linux-2.2.14/include/linux/*.h bak** cho phép sao chép mọi file với tên có hai kí hiệu cuối cùng là *".h"* sang thư mục con *bak*.

Chính vì lí do nói trên, dù trong nhiều lệnh tuy không nói đến việc sử dụng kí hiệu mô tả nhóm file nhưng chúng ta có thể áp dụng chúng nếu điều đó không trái với suy luận thông thường. Do những tình huống như thế là quá phong phú cho nên không thể giới thiệu hết trong tài liệu. Chúng ta chú ý một giải pháp là mỗi khi sử dụng một lệnh nào đó, nên thử nghiệm cách thức hiệu quả này.

4.12 Đổi tên file với lệnh mv

Cú pháp lệnh đổi tên file: mv <tên-cũ> <tên-mới>

Lệnh này cho phép đổi tên file từ tên cũ thành tên mới.

Ví dụ:

```
# mv vd newfile
```

Lệnh này sẽ đổi tên file *vd* thành *newfile*. Trong trường hợp file *newfile* đã tồn tại, nội dung của file *vd* sẽ ghi đè lên nội dung của file *newfile*

4.13 Lệnh uniq loại bỏ những dòng không quan trọng trong file

Trong một số trường hợp khi xem nội dung một file, chúng ta thấy có một số các thông tin bị trùng lặp, ví dụ các dòng trống hoặc các dòng chứa nội dung giống nhau. Để đồng thời làm gọn và thu nhỏ kích thước của file, có thể sử dụng lệnh *uniq* để liệt kê ra nội dung file sau khi đã loại bỏ các dòng trùng lặp.

Cú pháp lệnh: **uniq [tùy-chọn] [input] [output]**

Lệnh *uniq* sẽ loại bỏ các dòng trùng lặp kề nhau từ input (thiết bị vào chuẩn) và chỉ giữ lại một dòng duy nhất trong số các dòng trùng lặp rồi đưa ra output (thiết bị ra chuẩn).

Các tuỳ chọn:

- _____c, --count : đếm và hiển thị số lần xuất hiện của các dòng trong file.
- _____d : hiển thị lên màn hình dòng bị trùng lặp.
- _____u : hiển thị nội dung file sau khi xóa bỏ toàn bộ các dòng bị trùng lặp không giữ lại một dòng nào.
- _____i : hiển thị nội dung file sau khi xóa bỏ các dòng trùng lặp và chỉ giữ lại duy nhất một dòng có nội dung bị trùng lặp.
- _____D : hiển thị tất cả các dòng trùng lặp trên màn hình.

ẩ ếu sử dụng lệnh uniq trên một file không có các dòng trùng lặp thì lệnh không có tác dụng.

Ví dụ: người dùng sử dụng lệnh *cat* để xem nội dung file *vduniq*

```
# cat vduniq
Gnome có hai phương pháp để thoát ra ngoài.
Gnome có hai phương pháp để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel, trước hết phải
thêm nút này vào Panel.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Trong file *vduniq* có hai dòng bị trùng lặp và kề nhau là dòng thứ 1 và 2.

```
Gnome có hai phương pháp để thoát ra ngoài.
Gnome có hai phương pháp để thoát ra ngoài.
```

và dòng thứ 5 và 6

```
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Dùng lệnh *uniq* để loại bỏ dòng trùng lặp:

```
# uniq vduniq
Gnome có hai phương pháp để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel, trước hết phải
thêm nút này vào Panel.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Dòng cuối cùng trong file vduniq có nội dung trùng với dòng thứ 5, nhưng sau lệnh uniq, nó không bị xóa vì không kề với dòng có nội dung trùng lặp.

4.14 Sắp xếp nội dung file với lệnh sort

Là lệnh đọc các thông tin và sắp xếp chúng theo thứ tự trong bảng chữ cái hoặc theo thứ tự được quy định theo các tùy chọn của lệnh.

Cú pháp lệnh: **sort [tùy-chọn] [file] ...**

Hiển thị nội dung sau khi sắp xếp của một hoặc nhiều file ra thiết bị ra chuẩn là tác dụng của lệnh sort. ả gầm định sắp xếp theo thứ tự từ điển của các dòng có trong các file (từng chữ cái theo bảng chữ hệ thống (chẳng hạn ASCII) và kể từ vị trí đầu tiên trong các dòng).

Các tùy chọn:

+<số1> [-<số2>] : Hai giá trị số1 và số2 xác định "khóa" sắp xếp của các dòng, thực chất lấy xâu con từ vị trí số1 tới vị trí số2 của các dòng để so sánh lấy thứ tự sắp xếp các dòng. ả ếu số2 không có thì coi là hết các dòng; nếu số2 nhỏ hơn số1 thì bỏ qua lựa chọn này. Chú ý, nếu có số2 thì phải cách số1 ít nhất một dấu cách.

- b : bỏ qua các dấu cách đứng trước trong phạm vi sắp xếp.
- c : kiểm tra nếu file đã sắp xếp thì thôi không sắp xếp nữa.
- d : xem như chỉ có các ký tự [a-zA-Z0-9] trong khóa sắp xếp, các dòng có các kí tự đặc biệt (dấu cách, ? ...) được đưa lên đầu.
- f : sắp xếp không phân biệt chữ hoa chữ thường.
- n : sắp xếp theo kích thước của file.
- r : chuyển đổi thứ tự sắp xếp hiện thời.

Ví dụ: muốn sắp xếp file vdsort

```
# cat vdsort
trước hết phải thêm nút này vào Panel.
21434
bạn xác nhận là có thực sự muốn thoát hay không.
menu chính, chọn mục Logout ở đáy menu.
Bạn có thể sử dụng mục Logout từ menu chính
Gnome có hai phương pháp để thoát ra ngoài.
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
57879
Lựa chọn YES hoặc NO để kết thúc phiên làm việc với Gnome.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
# sort -f vdsort
21434
```

```
57879
Bạn có thể sử dụng mục Logout từ menu chính
bạn xác nhận là có thực sự muốn thoát hay không.
Gnome có hai phương pháp để thoát ra ngoài.
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
Lựa chọn YES hoặc NO để kết thúc phiên làm việc với Gnome.
menu chính, chọn mục Logout ở đáy menu.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
trước hết phải thêm nút này vào Panel.
```

Có thể kết hợp lệnh sort với các lệnh khác, ví dụ: **# ls -s | sort –n**

Lệnh này cho thứ tự sắp xếp của các file theo kích thước trong thư mục hiện thời

4.15 Tìm theo nội dung file bằng lệnh grep

Lệnh *grep* cũng như lệnh *ls* là hai lệnh rất quan trọng trong Linux. Lệnh này có hai tác dụng cơ bản, tác dụng thứ nhất là lọc đầu ra của một lệnh khác.

Cú pháp là: **<lệnh> | grep <mẫu lọc>**

tác dụng thứ hai, là tìm dòng chứa mẫu đã định trong file được chỉ ra.

Cú pháp lệnh grep: **grep [tùy-chọn] <mẫu-lọc> [file]**

Lệnh *grep* hiển thị tất cả các dòng có chứa mẫu-lọc trong file được chỉ ra (hoặc từ thiết bị vào chuẩn nếu không có file hoặc file có dạng là dấu "-")

Các tùy chọn:

- G, --basic-regexp : xem mẫu lọc như một biểu thức thông thường. Điều này là ngầm định.
- E, --extended-regexp : xem mẫu lọc như một biểu thức mở rộng.
- F, --fixed-strings : xem mẫu như một danh sách các xâu cố định, được phân ra bởi các dòng mới. ả goài lệnh grep còn có hai lệnh là egrep và fgrep. egrep tương tự như lệnh grep -E, fgrep tương tự với lệnh grep -F .

Lệnh grep còn có các tùy chọn sau:

- A ả UM, --after-context=ả UM : đưa ra ả UM dòng nội dung tiếp theo sau dòng có chứa mẫu.
- B ả UM, --before-context=ả UM : đưa ra ả UM dòng nội dung trước dòng có chứa mẫu.
- C [ả UM], --context[=ả UM] : hiển thị ả UM dòng (mặc định là 2 dòng) nội dung.

- _____ả UM : giống --context=ả UM đưa ra các dòng nội dung trước và sau dòng có chứa mẫu. Tuy nhiên, grep sẽ không đưa ra dòng nào nhiều hơn một lần.
- _____b, --byte-offset : hiển thị địa chỉ tương đối trong file đầu vào trước mỗi dòng được đưa ra
- _____c, --count : đếm số dòng tương ứng chứa mẫu trong file đầu vào thay cho việc hiển thị các dòng chứa mẫu.
- _____d ACTIOả , --directories=ACTIOả : nếu đầu vào là một thư mục, sử dụng ACTIOả để xử lý nó. Mặc định, ACTIOả là read, tức là sẽ đọc nội dung thư mục như một file thông thường. ả ếu ACTIOả là skip, thư mục sẽ bị bỏ qua. ả ếu ACTIOả là recurse, grep sẽ đọc nội dung của tất cả các file bên trong thư mục (đệ quy); tùy chọn này tương đương với tùy chọn -r.
- _____f file, --file=file : lấy các mẫu từ file, một mẫu trên một dòng. File trống chứa đựng các mẫu rỗng, và các dòng đưa ra cũng là các dòng trống.
- _____H, --with-file : đưa ra tên file trên mỗi dòng chứa mẫu tương ứng.
- _____h, --no-filename : không hiển thị tên file kèm theo dòng chứa mẫu trong trường hợp tìm nhiều file.
- _____i : hiển thị các dòng chứa mẫu không phân biệt chữ hoa chữ thường.
- _____l : đưa ra tên các file trùng với mẫu lọc.
- _____n, --line-number : thêm số thứ tự của dòng chứa mẫu trong file.
- _____r, --recursive : đọc tất cả các file có trong thư mục (đệ quy).
- _____s, --no-messages : bỏ qua các thông báo lỗi file không đọc được hoặc không tồn tại.
- _____v, --invert-match : hiển thị các dòng không chứa mẫu.
- _____w, --word-regexp : chỉ hiển thị những dòng có chứa mẫu lọc là một từ trọn vẹn.
- _____x, --line-regexp : chỉ hiển thị những dòng mà nội dung trùng hoàn toàn với mẫu lọc.

Cũng có thể sử dụng các ký hiệu biểu diễn thông thường (regular - expression) trong mẫu lọc để đưa ra được nhiều cách tìm kiếm file khác nhau.

ả goài các tùy chọn khác nhau, lệnh *grep* còn có hai dạng nữa trên Linux. Hai dạng đó là *egrep* - sử dụng với các mẫu lọc phức tạp, và *fgrep* - sử dụng để tìm nhiều mẫu lọc cùng một lúc. Thỉnh thoảng một biểu thức đơn giản không thể xác định được đối tượng cần tìm, ví dụ, như đang cần tìm các dòng có một hoặc hai mẫu lọc. ả những lúc đó, lệnh egrep tỏ ra rất có ích.

Formatted: Bullets and Numbering

egrep - expression grep - có rất nhiều các ký hiệu biểu diễn mạnh hơn grep. Dưới đây là các ký hiệu hay dùng:

Ký hiệu	*Ý nghĩa*
c	- thay thế cho ký tự c
\c	- hiển thị c như là một ký tự bình thường nếu c là một ký tự điều khiển
^	- bắt đầu một dòng
$	- kết thúc dòng
.	- thay cho một ký tự đơn
[xy]	- chọn một ký tự trong tập hợp các ký tự được đưa ra
[^xy]	- chọn một ký tự không thuộc tập hợp các ký tự được đưa ra
c*	- thay cho một mẫu có hoặc không chứa ký tự c
c+	- thay cho một mẫu có chứa một hoặc nhiều hơn ký tự c
c?	- thay cho một mẫu không có hoặc chỉ có chứa duy nhất một ký tự c
a | b	- hoặc là a hoặc là b
(a)	- a một biểu thức

Ví dụ:

giả sử bây giờ muốn tìm các dòng có chứa một hoặc nhiều hơn ký tự b trên file passwk với lệnh egrep.

```
# egrep 'b+' /etc/passwd | head
```

cho ra các dòng kết quả sau:

```
root : x : 0 : 0 : root : /root : /bin/bash
bin : x : 1 : 1 : bin : /bin :
daemon : x : 2 : 2 : daemon : /sbin :
sync : x : 5 : 0 : sync : /sbin : /bin/sync
shutdown : x : 6 : 0 : shutdown : /sbin : /sbin/shutdown
halt : x : 7 : 0 : halt : /sbin : /sbin/halt
gopher : x : 13 : 30 : gopher : /usr/lib/gopher-data :
nobody : x : 99 : 99 : Nobody : / :
xfs : x : 43 : 43 : X Font Server : /etc/X11/fs : /bin/false
named : x : 25 : 25 : Named : /var/named : /bin/false
```

Bất kỳ lúc nào muốn tìm các dòng có chứa nhiều hơn một mẫu lọc, egrep là lệnh tốt nhất để sử dụng.

Có những lúc cần phải tìm nhiều mẫu lọc trong một lúc. Ví dụ, có một file chứa rất nhiều mẫu lọc và muốn sử dụng một lệnh trong Linux để tìm các dòng có chứa các mẫu đó. Lệnh fgrep sẽ làm được điều này.

Formatted: Bullets and Numbering

Ví dụ: file *thu* có nội dung như sau:

```
# cat thu
/dev/hda4: Linux/i386 ext2 filesystem
/dev/hda5: Linux/i386 swap file
/dev/hda8: Linux/i386 swap file
/dev/hda9: empty
/dev/hda10: empty
thutest
toithutest
```

và file *mauloc* có nội dung là:

```
# cat mauloc
empty
test
```

Bây giờ muốn sử dụng nội dung file *mauloc* làm mẫu lọc để tìm các câu trong file *thu*, hãy gõ lệnh:

```
# fgrep -i -f mauloc thu
/dev/hda9: empty
/dev/hda10: empty
thutest
toithutest
```

Một số ví dụ sử dụng lệnh grep

Với file data file có nội dung sau: # cat datafile

```
northwest  NW Charles Main       3.0  .98  3  34
western    WE Sharon Gray        5.3  .97  5  23
southwest  SW Lewis Dalsass      2.7  .8   2  18
southern   SO Suan Chin          5.1  .95  4  15
southeast  SE Patricia Hemenway  4.0  .7   4  17
eastern    EA TB Savage          4.4  .84  5  20
northeast  NE AM Main Jr         5.1  .94  3  13
north      NO Margot Weber       4.5  .89  5  9
central    CT Ann Stephens       5.7  .94  5  13


# grep NW datafile
northwest NW  Charles Main   3.0   .98    3   34
# grep '^n' datafile
northwest NW  Charles Main   3.0   .98    3    34
northeast NE  AM Main Jr.    5.1   .94    3    13
```

```
north     NO  Margot Weber   4.5   .89    5     9
# grep '4$' datafile
northwest NW  Charles Main   3.0   .98    3     34
# grep TB Savage datafile
grep: Savage: No such file or directory
datafile: eastern  EA   TB Savage 4.4   .84     5     20
# grep 'TB Savage' datafile
eastern  EA        TB Savage      4.4   .84    5   20
# grep '5\..' datafile
western   WE   Sharon Gray         5.3   .97   5    23
southern  SO   Suan Chin           5.1   .95   4    15
northeast NE   AM Main Jr.         5.1   .94   3    13
central    CT    Ann Stephens          5.7    .94   5    13
# grep '\.5' datafile
north      NO  Margot Weber      4.5   .89   5     9
# grep '^[we]' datafile
western   WE   Sharon Gray         5.3   .97    5   23
# grep '[^0-9]' datafile
northwest  NW      Charles Main  3.0    .98    3      34
western     WE    Sharon Gray    5.3    .97    5      23
southwest   SW    Lewis Dalsass  2.7    .8     2      18
southern    SO    Suan Chin        5.1    .95    4      15
southeast   SE    Patricia Hemenway 4.0    .7     4      17
eastern     EA    TB Savage         4.4    .84    5      20
northeast   NE    AM Main Jr.       5.1    .94    3      13
north       NO    Margot Weber     4.5    .89    5      9
central     CT    Ann Stephens     5.7    .94    5      13
eastern     EA    TB Savage        4.4   .84     5     20
# grep '[A-Z][A-Z] [A-Z]' datafile
eastern     EA   TB Savage         4.4   .84     5    20
northeast   NE   AM Main Jr.       5.1   .94     3    13
# grep 'ss* ' datafile
northwest   NW   Charles Main      3.0   .98     3     34
southwest   SW   Lewis Dalsass     2.7   .8      2     18
# grep '[a-z]\{9\}' datafile
northwest   NW   Charles Main    3.0   .98    3     34
southwest   SW   Lewis Dalsass     2.7   .8     2     18
southeast   SE   Patricia Hemenway 4.0   .7     4     17
northeast   NE   AM Main Jr.       5.1   .94    3     13
# grep '\(3\)\.[0-9].*\1    *\1' datafile
```

```
northwest   NW       Charles Main   3.0   .98     3     34
# grep '\<north' datafile
northwest   NW       Charles Main  3.0   .98     3    34
northeast   NE       AM Main Jr.   5.1   .94     3    13
north       NO     Margot Weber    4.5   .89     5     9
# grep '\<north\>' datafile
north      NO    Margot Weber    4.5   .89     5     9
# grep '\<[a-z].*n\>' datafile
northwest  NW        Charles Main     3.0   .98     3    34
western    WE        Sharon Gray      5.3   .97     5    23
southern   SO        Suan Chin     5.1   .95     4    15
eastern    EA        TB Savage     4.4   .84     5    20
northeast  NE        AM Main Jr.       5.1   .94     3    13
central    CT        Ann Stephens  5.7   .94     5    13
#grep -n '^south' datafile
3:southwest SW        Lewis Dalsass      2.7   .8    2   18
4:southern  SO        Suan Chin          5.1   .95  4    15
5:southeast SE        Patricia Hemenway  4.0   .7    4   17
# grep -i 'pat' datafile
southeast   SE        Patricia Hemenway  4.0   .7    4   17
# grep -v 'Suan Chin' datafile
northwest   NW      Charles Main         3.0   .98   3  34
western     WE      Sharon Gray          5.3   .97   5  23
southwest   SW      Lewis Dalsass        2.7   .8    2  18
southeast   SE      Patricia Hemenway    4.0   .7    4  17
eastern     EA      TB Savage            4.4   .84   5  20
northeast   NE      AM Main Jr.          5.1   .94   3  13
north       NO      Margot Weber         4.5   .89   5  9
central     CT      Ann Stephens         5.7   .94   5  13
# grep -l 'SE'  *     (tìm file chứa xâu "SE")
datafile
datebook
# grep  -w 'north' datafile
north      NO       Margot Weber        4.5   .89     5     9
# echo $LOGNAME
lewis
# grep -i "$LOGNAME" datafile
southwest  SW       Lewis Dalsass     2.7   .8    2     18
```

4.16 Tìm theo các đặc tính của file với lệnh find

Các đoạn trên đây đã giới thiệu cách thức tìm file theo nội dung với các lệnh grep, egrep và fgrep. Linux còn cho phép người dùng sử dụng một cách thức khác đầy năng lực, đó là sử dụng lệnh *find*, lệnh tìm file theo các thuộc tính của file. Lệnh này có một sự khác biệt so với các lệnh khác, đó là các tùy chọn của lệnh là một từ chứ không phải một ký tự. Điều kiện cần đối với lệnh này là chỉ ra được điểm bắt đầu của việc tìm kiếm trong hệ thống file và những quy tắc cần tuân theo của việc tìm kiếm.

Cú pháp của lệnh find: **find [đường-dẫn] [biểu-thức]**

Lệnh *find* thực hiện việc tìm kiếm file trên cây thư mục theo biểu thức được đưa ra. Mặc định đường dẫn là thư mục hiện thời, biểu thức là -print.

- Các toán tử:

(EXPR); ! EXPR hoặc -not EXPR; EXPR1 -a EXPR2 hoặc EXPR1 -and EXPR2; EXPR1 -o EXPR2 hoặc EXPR1 -or EXPR2; và EXPR1, EXPR2

- Các tùy chọn lệnh: tất cả các tùy chọn này luôn trả về giá trị true và được đặt ở đầu biểu thức:
  - daystart : đo thời gian (-amin, -atime, -cmin, -ctime, -mmin, -mtime).
  - depth : thực hiện tìm kiếm từ nội dung bên trong thư mục trước (mặc định việc tìm kiếm được thực hiện bắt đầu tại gốc cây thư mục có chứa file cần tìm).
  - follow : (tùy chọn này chỉ áp dụng cho thư mục) nếu có tùy chọn này thì các liên kết tượng trưng có trong một thư mục liên kết sẽ được chỉ ra.
  - help, --help : hiển thị kết quả của lệnh find và thoát. các test
  - amin n : tìm file được truy nhập n phút trước.
  - atime n : tìm file được truy nhập n*24 giờ trước.
  - cmin n : trạng thái của file được thay đổi n phút trước đây.
  - ctime n : trạng thái của file được thay đổi n*24 giờ trước đây.
  - empty : file rỗng và hoặc là thư mục hoặc là file bình thường.
  - fstype kiểu : file thuộc hệ thống file với kiểu.
  - gid n : chỉ số nhóm của file là n.
  - group nhóm : file thuộc quyền sở hữu của nhóm.
  - links n : file có n liên kết.
  - mmin n : dữ liệu của file được sửa lần cuối vào n phút trước đây.
  - mtime n : dữ liệu của file được sửa vào n*24 giờ trước đây.
  - name mẫu : tìm kiếm file có tên là mẫu. Trong tên file có thể chứa cả các ký tự đại diện như dấu "*", "?"...

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

- _____type kiểu : tìm các file thuộc kiểu với kiểu nhận các giá trị:
  - b: đặc biệt theo khối
  - c: đặc biệt theo ký tự
  - d: thư mục
  - p: pipe
  - f: file bình thường
  - l: liên kết tượng trưng
  - s: socket
- _____uid n: chỉ số người sở hữu file là n.
- _____user tên-người: file được sở hữu bởi người dùng tên-người.

Các hành động:

- _____exec lệnh : tùy chọn này cho phép kết hợp lệnh find với một lệnh khác để có được thông tin nhiều hơn về các thư mục có chứa file cần tìm. Tùy chọn exec phải sử dụng dấu {} - nó sẽ thay thế cho tên file tương ứng, và dấu '\' tại cuối dòng lệnh, (phải có khoảng trống giữa {} và '\'). Kết thúc lệnh là dấu ';'
- _____fprint file : hiển thị đầy đủ tên file vào trong file. ả ếu file không tồn tại thì sẽ được tạo ra, nếu đã tồn tại thì sẽ bị thay thế nội dung.
- _____print : hiển thị đầy đủ tên file trên thiết bị ra chuẩn.
- _____ls : hiển thị file hiện thời theo khuôn dạng: liệt kê danh sách đầy đủ kèm cả số thư mục, chỉ số của mỗi file, với kích thước file được tính theo khối (block).

Ví dụ:

```
# find -name 'what*'
./usr/bin/whatis
./usr/bin/whatnow
./usr/doc/AfterStep-1.8.0/TODO/1.0ỴtoỴ1.5/whatsnew
./usr/doc/gnome-libs-devel-1.0.55/devel-docs/gnome-dev-info/gnome-dev-
info/what.html
./usr/doc/gnome-libs-devel-1.0.55/devel-docs/gnome-dev-info/gnome-dev-
info/whatis.html
# find . -type f -exec grep -l -i mapping {} \ ;
./OWL/WordMap/msw-to-txt.c
./.elm/aliases.text
./Mail/mark
./News/usenet.alt
./bin/my.new.cmd: Permission denied
./src/fixit.c
```

```
./temp/attach.msg
```

4.17 Nén và sao lưu các file

Sao lưu các file (lệnh tar)

Dữ liệu rất có giá trị, sẽ mất nhiều thời gian và công sức nếu phải tạo lại, thậm chí có lúc cũng không thể nào tạo lại được. Vì vậy, Linux đưa ra các cách thức để người dùng bảo vệ dữ liệu của mình.

Có bốn nguyên nhân cơ bản khiến dữ liệu có thể bị mất: lỗi phần cứng, lỗi phần mềm, lỗi do con người hoặc do thiên tai.

Sao lưu là cách để bảo vệ dữ liệu một cách kinh tế nhất. Bằng cách sao lưu dữ liệu, sẽ không có vấn đề gì xảy ra nếu dữ liệu trên hệ thống bị mất.

Một vấn đề rất quan trọng trong việc sao lưu đó là lựa chọn phương tiện sao lưu, cần phải quan tâm đến giá cả, độ tin cậy, tốc độ, ích lợi cũng như tính khả dụng của các phương tiện sao lưu.

Có rất nhiều các công cụ có thể được sử dụng để sao lưu. Các công cụ truyền thống là *tar, cpio* và *dump* (công cụ trong tài liệu này là tar). ấ goài ra còn rất nhiều các công cụ khác có thể lựa chọn tùy theo phương tiện sao lưu có trong hệ thống.

Có hai kiểu sao lưu là sao lưu theo kiểu toàn bộ (full backup) và sao lưu theo kiểu tăng dần (incremental backup). Sao lưu toàn bộ thực hiện việc sao mọi thứ trên hệ thống file, bao gồm tất cả các file. Sao lưu tăng dần chỉ sao lưu những file được thay đổi hoặc được tạo ra kể từ đợt sao lưu cuối cùng.

Việc sao lưu toàn bộ có thể được thực hiện dễ dàng với lệnh *tar* .

Cú pháp: **tar [tùy-chọn] [<file>, ...] [<thư-mục>, ...]**

Lệnh (chương trình) *tar* được thiết kế để tạo lập một file lưu trữ duy nhất. Với *tar*, có thể kết hợp nhiều file thành một file duy nhất có kích thước lớn hơn, điều này sẽ giúp cho việc di chuyển file hoặc sao lưu băng từ trở nên dễ dàng hơn nhiều.

Lệnh *tar* có các lựa chọn:

- c, --create : tạo file lưu trữ mới.
- d, --diff, --compare : tìm ra sự khác nhau giữa file lưu trữ và file hệ thống được lưu trữ.
- - delete : xóa từ file lưu trữ (không sử dụng cho băng từ).
- r, --append : chèn thêm file vào cuối file lưu trữ.

Formatted: Bullets and Numbering

- t, --list : liệt kê nội dung của một file lưu trữ.
- u, --update : chỉ thêm vào file lưu trữ các file mới hơn các file đã có.
- x, --extract, --get : tách các file ra khỏi file lưu trữ.
- C, --directory tên-thư-mục : thay đổi đến thư mục có tên là tên-thư-mục.
- - checkpoint : đưa ra tên thư mục khi đọc file lưu trữ.
- f, --file [HOSTả AME:]file : tùy chọn này xác định tên file lưu trữ hoặc thiết bị lưu trữ là file (nếu không có tùy chọn này, mặc định nơi lưu trữ là /dev/rmt0).
- h, --dereference : không hiện các file liên kết mà hiện các file mà chúng trỏ tới.
- k, --keep-old-files : giữ nguyên các file lưu trữ đang tồn tại mà không ghi đè file lưu trữ mới lên chúng.
- K, --starting-file file : bắt đầu tại file trong file lưu trữ.
- l, --one-file-system : tạo file lưu trữ trên hệ thống file cục bộ.
- M, --multi-volume : tùy chọn này được sử dụng khi dung lượng của file cần sao lưu là lớn và không chứa hết trong một đơn vị lưu trữ vật lý.
- ả , --after-date DATE, --newer DATE : chỉ lưu trữ các file mới hơn các file được lưu trữ trong ngày DATE.
- - remove-files : xóa file gốc sau khi đã sao lưu chúng vào trong file lưu trữ.
- - totals : đưa ra tổng số byte được tạo bởi tùy chọn --create.
- v, --verbose : hiển thị danh sách các file đã được xử lý.

Ví dụ:

```
# tar --create --file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
```

Lệnh trên tạo một file sao lưu của thư mục */usr/src* trong thư mục */dev/ftape*, (dòng thông báo ở trên cho biết rằng *tar* sẽ chuyển cả dấu / vào trong file sao lưu).

ả ếu việc sao lưu không thể thực hiện gọn vào trong một băng từ, lúc đó hãy sử dụng tùy chọn -M:

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
```

Chú ý rằng phải định dạng đĩa mềm trước khi thực hiện việc sao lưu, có thể sử dụng một thiết bị đầu cuối khác để thực hiện việc định dạng đĩa khi *tar* yêu cầu một đĩa mềm mới. Sau khi thực hiện việc sao lưu, có thể kiểm tra kết quả của công việc bằng tùy chọn - - *compare*:

```
# tar --compare --verbose -f /dev/ftape
usr/src/
```

```
usr/src/Linux
usr/src/Linux-1.2.10-includes/
...
```

Để sử dụng kiểu sao lưu tăng dần, hãy sử dụng tùy chọn -ẩ :

```
# tar --create --newer '8 Sep 1995' --file /dev/ftape /usr/src --
verbose
tar: Removing leading / from absolute path names in the archive
usr/src/
usr/src/Linux-1.2.10-includes/
usr/src/Linux-1.2.10-includes/include/
usr/src/Linux-1.2.10-includes/include/Linux/
usr/src/Linux-1.2.10-includes/include/Linux/modules/
usr/src/Linux-1.2.10-includes/include/asm-generic/
usr/src/Linux-1.2.10-includes/include/asm-i386/
usr/src/Linux-1.2.10-includes/include/asm-mips/
usr/src/Linux-1.2.10-includes/include/asm-alpha/
usr/src/Linux-1.2.10-includes/include/asm-m68k/
usr/src/Linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
```

Lưu ý rằng, *tar* không thể thông báo được khi các thông tin trong inode của một file bị thay đổi, ví dụ như thay đổi quyền truy nhập của file, hay thay đổi tên file chẳng hạn. Để biết được những thông tin thay đổi sẽ cần dùng đến lệnh find và so sánh với trạng thái hiện thời của file hệ thống với danh sách các file được sao lưu từ trước.

Nén dữ liệu với gzip

Việc sao lưu rất có ích nhưng đồng thời nó cũng chiếm rất nhiều không gian cần thiết để sao lưu. Để giảm không gian lưu trữ cần thiết, có thể thực hiện việc nén dữ liệu trước khi sao lưu, sau đó thực hiện việc giải nén để nhận lại nội dung trước khi nén.

Trong Linux có khá nhiều cách để nén dữ liệu, tài liệu này giới thiệu hai phương cách phổ biến là *gzip* và *compress*.

ả én, giải nén và xem nội dung các file với lệnh *gzip, gunzip* và *zcat.*

Cú pháp các lệnh này như sau:

gzip [tùy-chọn] [-S suffix] [< file>]

gunzip [tùy-chọn] [-S suffix] [<file>]

zcat [tùy-chọn] [<file>]

Lệnh *gzip* sẽ làm giảm kích thước của file và khi sử dụng lệnh này, file gốc sẽ bị thay thế bởi file nén với phần mở rộng là *.gz*, các thông tin khác liên quan đến file không thay đổi. ả ếu

không có tên file nào được chỉ ra thì thông tin từ thiết bị vào chuẩn sẽ được nén và gửi ra thiết bị ra chuẩn. Trong một vài trường hợp, lệnh này sẽ bỏ qua liên kết tượng trưng.

ẩ ếu tên file nén quá dài so với tên file gốc, *gzip* sẽ cắt bỏ bớt, *gzip* sẽ chỉ cắt phần tên file vượt quá 3 ký tự (các phần được ngăn cách với nhau bởi dấu chấm). ẩ ếu tên file gồm nhiều phần nhỏ thì phần dài nhất sẽ bị cắt bỏ. Ví dụ, tên file là *gzip.msdos.exe*, khi được nén sẽ có tên là *gzip.msd.exe.gz*.

File được nén có thể được khôi phục trở lại dạng nguyên thể với lệnh *gzip -d* hoặc *gunzip*. Với lệnh *gzip* có thể giải nén một hoặc nhiều file có phần mở rộng là *.gz, -gz, .z, -z, _z* hoặc *.Z ... gunzip* dùng để giải nén các file nén bằng lệnh *gzip, zip, compress, compress -H*.

Lệnh zcat được sử dụng khi muốn xem nội dung một file nén trên thiết bị ra chuẩn.

Các tùy chọn:

- _____c, --stdout --to-stdout : đưa ra trên thiết bị ra chuẩn; giữ nguyên file gốc không có sự thay đổi. ẩ ếu có nhiều hơn một file đầu vào, đầu ra sẽ tuần tự là các file được nén một cách độc lập.
- _____d, --decompress --uncompress : giải nén.
- _____f, --force : thực hiện nén hoặc giải nén thậm chí file có nhiều liên kết hoặc file tương ứng thực sự đã tồn tại, hay dữ liệu nén được đọc hoặc ghi trên thiết bị đầu cuối.
- _____h, --help : hiển thị màn hình trợ giúp và thoát.
- _____l, --list : hiển thị những thông tin sau đối với một file được nén:
  - compressed size: kích thước của file nén
  - uncompressed size: kích thước của file được giải nén
  - ratio: tỷ lệ nén (0.0% nếu không biết)
  - uncompressed_name: tên của file được giải nén

ẩ ếu kết hợp với tùy chọn --*verbose*, các thông tin sau sẽ được hiển thị:

- method: phương thức nén
- crc: CRC 32-bit cho dữ liệu được giải nén
- date & time: thời gian các file được giải nén

ẩ ếu kết hợp với tùy chọn --*name*, tên file được giải nén, thời gian giải nén được lưu trữ trong file nén.

ẩ ếu kết hợp với tùy chọn --*verbose*, tổng kích thước và tỷ lệ nén của tất cả các file sẽ được hiển thị.

ẩ ếu kết hợp với tùy chọn --*quiet*, tiêu đề và tổng số dòng của các file nén không được hiển thị.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

- _____n, --no-name : khi nén, tùy chọn này sẽ không lưu trữ tên file gốc và thời gian nén, (tên file gốc sẽ luôn được lưu nếu khi nén tên của nó bị cắt bỏ). Khi giải nén, tùy chọn này sẽ không khôi phục lại tên file gốc cũng như thời gian thực hiện việc nén. Tùy chọn này được ngầm định.
- _____ẩ , --name : tùy chọn này ngược với tùy chọn trên (-n), nó hữu ích trên hệ thống có sự giới hạn về độ dài tên file hay khi thời điểm nén bị mất sau khi chuyển đổi file.
- _____-q, --quiet : bỏ qua mọi cảnh báo.
- _____r, --recursive : nén thư mục.
- _____S .suf, --suffix .suf : sử dụng phần mở rộng .suf thay cho .gz. Bất kỳ phần mở rộng nào cũng có thể được đưa ra, nhưng các phần mở rộng khác .z và .gz sẽ bị ngăn chặn để tránh sự lộn xộn khi các file được chuyển đến hệ thống khác.
- _____t, --test : tùy chọn này được sử dụng để kiểm tra tính toàn vẹn của file được nén
- _____v, --verbose : hiển thị phần trăm thu gọn đối với mỗi file được nén hoặc giải nén
- _____#, --fast, --best : điều chỉnh tốc độ của việc nén bằng cách sử dụng dấu #, nếu -# là -1 hoặc --fast thì sử dụng phương thức nén nhanh nhất (less compression), nếu là -9 hoặc --best thì sẽ dùng phương thức nén chậm nhất (best compression). ẩ gầm định mức nén là -6 (đây là phương thức nén theo tốc độ nén cao).

Ví dụ:

```
# ls /home/test
Desktop data dictionary newt-0.50.8 rpm save vd1
# gzip /home/test/vd1
# ls /home/test
Desktop data dictionary newt-0.50.8 rpm save vd1.gz
# zcat /home/test/vd1
PID TTY TIME CMD
973 pts/0 00:00:00 bash
996 pts/0 00:00:00 man
1008 pts/0 00:00:00 sh
1010 pts/0 00:00:00 less
1142 pts/0 00:00:00 cat
1152 pts/0 00:00:00 cat
1181 pts/0 00:00:00 man
1183 pts/0 00:00:00 sh
1185 pts/0 00:00:00 less
```

Nén, giải nén và xem file với các lệnh compress, uncompress, zcat

Cú pháp các lệnh như sau:

Formatted: Bullets and Numbering

compress [tùy-chọn] [<file>]

uncompress [tùy-chọn] [<file>]

zcat [tùy-chọn] [<file>]

Lệnh *compress* làm giảm kích thước của file và khi sử dụng lệnh này, file gốc sẽ bị thay thế bởi file nén với phần mở rộng là *.Z*, các thông tin khác liên quan đến file không thay đổi. ả ếu không có tên file nào được chỉ ra, thông tin từ thiết bị vào chuẩn sẽ được nén và gửi ra thiết bị ra chuẩn. Lệnh *compress* chỉ sử dụng cho các file thông thường. Trong một vài trường hợp, nó sẽ bỏ qua liên kết tượng trưng. ả ếu một file có nhiều liên kết cứng, *compress* bỏ qua việc nén file đó trừ khi có tùy chọn *-f*.

Các tùy chọn:

- _____f : nếu tùy chọn này không được đưa ra và compress chạy trong chế độ nền trước, người dùng sẽ được nhắc khi các file đã thực sự tồn tại và có thể bị ghi đè. Các file được nén có thể được khôi phục lại nhờ việc sử dụng lệnh *uncompress*.
- _____c : tùy chọn này sẽ thực hiện việc nén hoặc giải nén rồi đưa ra thiết bị ra chuẩn, không có file nào bị thay đổi.

Lệnh *zcat* tương đương với *uncompress -c*. zcat thực hiện việc giải nén hoặc là các file được liệt kê trong dòng lệnh hoặc từ thiết bị vào chuẩn để đưa ra dữ liệu được giải nén trên thiết bị ra chuẩn.

- _____r : nếu tùy chọn này được đưa ra, compress sẽ thực hiện việc nén các thư mục.
- _____v : hiển thị tỷ lệ giảm kích thước cho mỗi file được nén.

4.18 Liên kết (link) tập tin

Trong Linux có 2 hình thức liên kết hoàn toàn khác nhau, đó là *hard link* và *soft link* (hay *symbolic link*).

Hard link cho phép tạo một tên mới cho tập tin. Các tên này có vai trò hoàn toàn như nhau và tập tin chỉ bị hoàn toàn xóa bỏ khi hard link cuối cùng của nó bị xóa. Lệnh ls –l cho phép hiển thị số hard link đến tập tin.

Symbolic link có chức năng giống như shortcut của MS Windows. Khi ta đọc/ghi soft link, ta đọc/ghi tập tin; khi ta xóa symbolic link, ta chỉ xóa symbolic link và tập tin được giữ nguyên. Link được tạo bởi lệnh ln . Tự chọn ln –s cho phép tạo symbolic link. Ví dụ

```
[tnminh@pascal tnminh]$ls -l
-rw-------   1 tnminh   pkt 517 Oct 27 12:00 mbox
drwxr-xr-x   2 tnminh   pkt 4096 Aug 31 17:50 security
[tnminh@pascal tnminh]$ln –s mbox mybox
[tnminh@pascal tnminh]$ln –s security securproj
```

```
[tnminh@pascal tnminh]$ln –l
-rw-------   1 tnminh   pkt 517 Oct 27 12:00      mbox
lrwxrwxrwx   1 tnminh   pkt 4 Oct 27 17:57   mymail -> mbox
lrwxrwxrwx   1 tnminh   pkt 8 Oct 27 17:57   secrproj -> security
drwxr-xr-x   2 tnminh   pkt 4096 Aug 31 17:50 security
[tnminh@pascal tnminh]$
```

Bạn đọc có thể thấy khá rõ kết quả của symbolic link qua thí dụ trên.

Symbolic link rất có nhiều ứng dụng. Ví dụ như một tập tin XXX của một chương trình YYY nằm trong thư mục /var/ZZZ. ả ếu phân mảnh của /var/ZZZ bị quá đầy, ta có thể "sơ tán" XXX qua một thư mục khác thuộc phân mảnh khác và tạo một link thế vào đó mà chương trình YYY vẫn không hề "hay biết" vì nó vẫn truy cập đến /var/ZZZ/XXX như thường lệ.

5. Các lệnh và tiện ích hệ thống

5.1 Các lệnh đăng nhập và thoát khỏi hệ thống

Đăng nhập

Sau khi hệ thống Linux (lấy Red Hat 6.2 làm ví dụ) khởi động xong, trên màn hình xuất hiện những dòng sau:

```
Ret Hat Linux release 6.2 (Zoot)
Kernel 2.2.14-5.0 on an i686
May1 login:
```

Chúng ta có thể thay đổi các dòng hiển thị như trình bày trên đây bằng cách sửa đổi file */etc/rc.d/rc.local* như sau:

Thay đoạn chương trình sau:

```
echo "" > /etc/issue
echo "$R" >> /etc/issue
echo "Kernel $(uname -r) on $a $SMP$(uname -m)" >> /etc/issue
cp -f /etc/issue /etc/issue.net
echo >> /etc/issue
```

thành

```
echo "" > /etc/issue
echo "Thông báo muốn hiển thị" >> /etc/issue
```

Ví dụ: sửa thành:

```
echo "" > /etc/issue
echo "This is my computer" >> /etc/issue
```

thì trên màn hình đăng nhập sẽ có dạng sau:

```
This is my computer
hostname login:
```

Dòng thứ nhất và dòng thứ hai cho biết loại phiên bản Linux, phiên bản của nhân và kiến trúc phần cứng có trên máy, dòng thứ ba là dấu nhắc đăng nhập để người dùng thực hiện việc đăng nhập. Chú ý là các dòng trên đây có thể thay đổi chút ít tùy thuộc vào phiên bản Linux.

Tại dấu nhắc đăng nhập, hãy nhập tên người dùng (còn gọi là tên đăng nhập): đây là tên kí hiệu đã cung cấp cho Linux nhằm nhận diện một người dùng cụ thể. Tên đăng nhập ứng với mỗi người dùng trên hệ thống là duy nhất, kèm theo một mật khẩu đăng nhập.

```
May1 login: root
Password:
```

Khi nhập xong tên đăng nhập, hệ thống sẽ hiện ra thông báo hỏi mật khẩu và di chuyển con trỏ xuống dòng tiếp theo để người dùng nhập mật khẩu. Mật khẩu khi được nhập sẽ không hiển thị trên màn hình và chính điều đó giúp tránh khỏi sự "nhòm ngó" của người khác.

ẩ ếu nhập sai tên đăng nhập hoặc mật khẩu, hệ thống sẽ đưa ra một thông báo lỗi:

```
May1  login: root
Password:
Login incorrect
Máy1 login:
```

ẩ ếu đăng nhập thành công, người dùng sẽ nhìn thấy một số thông tin về hệ thống, một vài tin tức cho người dùng... Lúc đó, dấu nhắc shell xuất hiện để người dùng bắt đầu phiên làm việc của mình.

```
May1 login: root
Password:
Last login: Fri Oct 27 14:16:09 on tty2
Root[may1 /root]#
```

Dãy kí tự trong dòng cuối cùng chính là dấu nhắc shell. Trong dấu nhắc này, root là tên người dùng đăng nhập, may1 là tên máy và /root tên thư mục hiện thời (vì đây là người dùng root). Khi dấu nhắc shell xuất hiện trên màn hình thì điều đó có nghĩa là hệ điều hành đã sẵn sàng tiếp nhận một yêu cầu mới của người dùng.

Dấu nhắc shell có thể khác với trình bày trên đây, nhưng có thể hiểu nó là chuỗi kí tự bắt đầu một dòng có chứa trỏ chuột và luôn xuất hiện mỗi khi hệ điều hành hoàn thành một công việc nào đó.

Thoái khỏi hệ thống

Để kết thúc phiên làm việc người dùng cần thực hiện thủ tục ra khỏi hệ thống. Có rất nhiều cách cho phép thoát khỏi hệ thống, ở đây chúng ta xem xét một số cách thông dụng nhất.

- Cách đơn giản nhất để đảm bảo thoát khỏi hệ thống đúng đắn là nhấn tổ hợp phím CTRL+ALT+DEL. Khi đó, trên màn hình sẽ hiển thị một số thông báo của hệ thống và cuối cùng là thông báo thoát trước khi tắt máy.

Cần chú ý là: ả ếu đang làm việc trong môi trường X Window System, hãy nhấn tổ hợp phím CTRL+ALT+BACKSPACE trước rồi sau đó hãy nhấn CTRL+ALT+DEL.

Hoặc sử dụng lệnh shutdown: **shutdown [tùy-chọn] <time> [cảnh-báo]**

Lệnh này cho phép dừng tất cả các dịch vụ đang chạy trên hệ thống.

Các tùy chọn của lệnh này như sau:

- _____k : không thực sự shutdown mà chỉ cảnh báo.
- _____r : khởi động lại ngay sau khi shutdown.
- _____h : tắt máy thực sự sau khi shutdown.
- _____f : khởi động lại nhanh và bỏ qua việc kiểm tra đĩa.
- _____F : khởi động lại và thực hiện việc kiểm tra đĩa.
- _____c : bỏ qua không chạy lệnh shutdown. Trong tùy chọn này không thể đưa ra tham số thời gian nhưng có thể đưa ra thông báo giải thích trên dòng lệnh gửi cho tất cả các người dùng.
- _____t số-giây : qui định init(8) chờ khoảng thời gian số-giây tạm dừng giữa quá trình gửi cảnh báo và tín hiệu kill, trước khi chuyển sang một mức chạy khác.

và hai tham số vị trí còn lại:

- *time :* đặt thời điểm shutdown. Tham số time có hai dạng. Dạng tuyệt đối là gg:pp (gg: giờ trong ngày, pp: phút) thì hệ thống sẽ shutdown khi đồng hồ máy trùng với giá trị tham số. Dạng tương đối là +<số> là hẹn sau thời khoảng <số> phút sẽ shutdown; coi shutdown lập tức tương đương với +0.
- *cảnh-báo :* thông báo gửi đến tất cả người dùng trên hệ thống. Khi lệnh thực hiện tất cả các máy người dùng đều nhận được cảnh báo.

Ví dụ: khi người dùng gõ lệnh: **shutdown +1** Sau mot phut nua he thong se shutdown!
trên màn hình của tất cả người dùng xuất hiện thông báo "Sau mot phut nua he thong se shutdown! " và sau một phút thì hệ thống shutdown thực sự.

Cách thứ ba là sử dụng lệnh halt với cú pháp như sau: **halt [tùy-chọn]** Lệnh này tắt hẳn máy.

Các tuỳ chọn của lệnh halt:

- _____w : không thực sự tắt máy nhưng vẫn ghi các thông tin lên file /var/log/wtmp (đây là file lưu trữ danh sách các người dùng đăng nhập thành công vào hệ thống).

- _____d : không ghi thông tin lên file /var/log/wtmp. Tùy chọn -n có ý nghĩa tương tự song không tiến hành việc đồng bộ hóa.
- _____f : thực hiện tắt máy ngay mà không thực hiện lần lượt việc dừng các dịch vụ có trên hệ thống.
- _____i : chỉ thực hiện dừng tất cả các dịch vụ mạng trước khi tắt máy.

Chúng ta cần nhớ rằng, nếu thoát khỏi hệ thống không đúng cách thì dẫn đến hậu quả là một số file hay toàn bộ hệ thống file có thể bị hư hỏng. Có thể sử dụng lệnh *exit* để trở về dấu nhắc đăng nhập hoặc kết thúc phiên làm việc bằng lệnh *logout*.

Khởi động lại hệ thống

ể goài việc thoát khỏi hệ thống nhờ các cách thức trên đây (ấn tổ hợp ba phím Ctrl+Alt+Del, dùng lệnh *shutdown* hoặc lệnh *halt*), khi cần thiết (chẳng hạn, gặp phải tình huống một trình ứng dụng chạy quẩn) có thể khởi động lại hệ thống nhờ lệnh *reboot*.

Cú pháp lệnh reboot: **reboot [tùy-chọn]**

Lệnh này cho phép khởi động lại hệ thống. ể ói chung thì chỉ siêu người dùng mới được phép sử dụng lệnh reboot, tuy nhiên, nếu hệ thống chỉ có duy nhất một người dùng đang làm việc thì lệnh reboot vẫn được thực hiện song hệ thống đòi hỏi việc xác nhận mật khẩu.

Các tùy chọn của lệnh reboot như sau là *-w, -d, -n, -f, -i* có ý nghĩa tương tự như trong lệnh halt.

5.2 Lệnh thay đổi mật khẩu passwd

Mật khẩu là vấn đề rất quan trọng trong các hệ thống đa người dùng và để đảm bảo tính bảo mật tối đa, cần thiết phải chú ý tới việc thay đổi mật khẩu. Thậm chí trong trường hợp hệ thống chỉ có một người sử dụng thì việc thay đổi mật khẩu vẫn là rất cần thiết.

Mật khẩu là một xâu kí tự đi kèm với tên người dùng để đảm bảo cho phép một người vào làm việc trong hệ thống với quyền hạn đã được quy định. Trong quá trình đăng nhập, người dùng phải gõ đúng tên và mật khẩu, trong đó gõ mật khẩu là công việc bắt buộc phải thực hiện. Tên người dùng có thể được công khai song mật khẩu thì tuyệt đối phải được đảm bảo bí mật.

Việc đăng ký tên và mật khẩu của siêu người dùng được tiến hành trong quá trình khởi tạo hệ điều hành Linux. Việc đăng ký tên và mật khẩu của một người dùng thông thường được tiến hành khi một người dùng mới đăng ký tham gia sử dụng hệ thống. Thông thường siêu người dùng cung cấp tên và mật khẩu cho người dùng mới (có thể do người dùng đề nghị) và dùng lệnh *adduser* (hoặc lệnh *useradd*) để đăng ký tên và mật khẩu đó với hệ thống. Sau đó, người dùng mới nhất thiết cần thay đổi mật khẩu để bảo đảm việc giữ bí mật cá nhân tuyệt đối.

Lệnh *passwd* cho phép thay đổi mật khẩu ứng với tên đăng nhập người dùng.

Cú pháp lệnh *passwd*: **passwd [tùy-chọn] [tên-người-dùng]** với các tùy chọn như sau:

- _____k : thay đổi mật khẩu người dùng. Lệnh đòi hỏi phải xác nhận quyền bằng việc gõ mật khẩu đang dùng trước khi thay đổi mật khẩu. Cho phép người dùng thay đổi mật khẩu của mình độc lập với siêu người dùng.
- _____f : đặt mật khẩu mới cho người dùng song không cần tiến hành việc kiểm tra mật khẩu đang dùng. Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- _____l : khóa một tài khoản người dùng. Việc khóa tài khoản thực chất là việc dịch bản mã hóa mật khẩu thành một xâu ký tự vô nghĩa bắt đầu bởi kí hiệu "!". Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- _____stdin : việc nhập mật khẩu người dùng chỉ được tiến hành từ thiết bị vào chuẩn không thể tiến hành từ đường dẫn (pipe). ẩ ếu không có tham số này cho phép nhập mật khẩu cả từ thiết bị vào chuẩn hoặc từ đường dẫn.
- _____u : mở khóa (tháo bỏ khóa) một tài khoản (đối ngẫu với tham số -l). Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- _____d : xóa bỏ mật khẩu của người dùng. Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- _____S : hiển thị thông tin ngắn gọn về trạng thái mật khẩu của người dùng được đưa ra. Chỉ siêu người dùng mới có quyền sử dụng tham số này.

ẩ ếu tên-người-dùng không có trong lệnh thì ngầm định là chính người dùng đã gõ lệnh này. Ví dụ khi người dùng user1 gõ lệnh: **# passwd user1** hệ thống thông báo:

```
Changing password for user user1
New UNIX password:
```

để người dùng nhập mật khẩu mới của mình vào. Sau khi người dùng gõ xong mật khẩu mới, hệ thống cho ra thông báo:

```
BAD PASSWORD: it is derived from your password entry
Retype new UNIX password:
```

để người dùng khẳng định một lần nữa mật khẩu vừa gõ dòng trên (nhớ phải gõ lại đúng hệt như lần trước). Không nên quá phân vân vì thông báo ở dòng phía trên vì hầu hết khi gõ mật khẩu mới luôn gặp những thông báo kiểu đại loại như vậy, chẳng hạn như:

```
BAD PASSWORD: it is too simplistic/systematic
```

Và sau khi chúng ta khẳng định lại mật khẩu mới, hệ thống cho ra thông báo:

```
Passwd: all authentication tokens updated successfully.
```

cho biết việc thay đổi mật khẩu thành công và dấu nhắc shell lại hiện ra.

Khi siêu người dùng gõ lệnh:

```
# passwd -S root
```

sẽ hiện ra thông báo

```
Changing password for user root
Password set, MD5 encryption
```

cho biết thuật toán mã hóa mật khẩu mà Linux sử dụng là một thuật toán hàm băm có tên là MD5.

Chú ý: Có một lời khuyên đối với người dùng là nên chọn mật khẩu không quá đơn giản quá (nhằm tránh người khác dễ dò tìm ra) hoặc không quá phức tạp (tránh khó khăn cho chính người dùng khi phải ghi nhớ và gõ mật khẩu). Đặc biệt không nên sử dụng họ tên, ngày sinh, số điện thoại ... của bản thân hoặc người thân làm mật khẩu vì đây là một trong những trường hợp mật khẩu đơn giản nhất.

ẩ ếu thông báo mật khẩu quá đơn giản được lặp đi lặp lại một vài lần và không có thông báo mật khẩu mới thành công đã quay về dấu nhắc shell thì nên gõ lại lệnh và chọn một mật khẩu mới phức tạp hơn đôi chút.

Lệnh xem, thiết đặt ngày, giờ hiện tại và xem lịch trên hệ thống

5.4 Lệnh date xem, thiết đặt ngày, giờ

Lệnh *date* cho phép có thể xem hoặc thiết đặt lại ngày giờ trên hệ thống.

Cú pháp của lệnh gồm hai dạng, dạng xem thông tin về ngày, giờ và dạng thiết đặt lại ngày giờ cho hệ thống:

date [tùy-chọn] [+định-dạng]

date [tùy-chọn] [MMDDhhmm[[CC[YY]]-ss]]

Các tùy-chọn như sau:

- _____d, --date=xâu-văn-bản : hiển thị thời gian dưới dạng xâu-văn-bản, mà không lấy "thời gian hiện tại của hệ thống" như theo ngầm định; xâu-văn-bản được đặt trong hai dấu nháy đơn hoặc hai dấu nháy kép.
- _____f, --file=file-văn-bản : giống như một tham số --date nhưng ứng với nhiều ngày cần xem: mỗi dòng của file-văn-bản có vai trò như một xâu-văn-bản trong trường hợp tham số --date.
- _____I, --iso-8601[=mô-tả] : hiển thị ngày giờ theo chuẩn ISO-8601 (ví dụ: 2000-11-8).
- _____I tương đương với tham số --iso-8601='date'. Với --iso-8601: nếu mô-tả là 'date' (hoặc không có) thì hiển thị ngày, nếu mô-tả là 'hours' hiển thị ngày+giờ, nếu mô-tả là 'minutes': ngày+giờ+phút; nếu mô-tả là 'seconds': ngày + giờ + phút + giây.
- _____r, --reference= file : hiển thị thời gian sửa đổi file lần gần đây nhất.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

- _____R, --rfc-822 : hiển thị ngày theo RFC-822 (ví dụ: Wed, 8 ả ov 2000 09:21:46 -0500).
- _____s, --set=xâu-văn-bản : thiết đặt lại thời gian theo kiểu xâu-văn-bản.
- _____u, --utc, --universal : hiển thị hoặc thiết đặt thời gian theo UTC (ví dụ: Wed ả ov 8 14:29:12 UTC 2000).
- _____- help : hiển thị thông tin trợ giúp và thoát.

Trong dạng lệnh *date* cho xem thông tin ngày, giờ thì tham số định-dạng điều khiển cách hiển thị thông tin kết quả. Định-dạng là dãy có từ một đến nhiều cặp gồm hai kí tự, trong mỗi cặp kí tự đầu tiên là % còn kí tự thứ hai mô tả định dạng.

Do số lượng định dạng là rất nhiều vì vậy chúng ta chỉ xem xét một số định dạng điển hình (để xem đầy đủ các định dạng, sử dụng lệnh man date).

Dưới đây là một số định dạng điển hình:

- **%%** : Hiện ra chính kí tự %.
- **%a** : Hiện ra thông tin tên ngày trong tuần viết tắt theo ngôn ngữ bản địa.
- **%A** : Hiện ra thông tin tên ngày trong tuần viết đầy đủ theo ngôn ngữ bản địa.
- **%b** : Hiện ra thông tin tên tháng viết tắt theo ngôn ngữ bản địa.
- **%B** : Hiện ra thông tin tên tháng viết đầy đủ theo ngôn ngữ bản địa.

Formatted: Bullets and Numbering

Trong dạng lệnh *date* cho phép thiết đặt lại ngày giờ cho hệ thống thì tham số [MMDDhhmm[[CC[YY] [.ss]] mô tả ngày, giờ mới cần thiết đặt, trong đó:

MM: hai số chỉ tháng,

DD: hai số chỉ ngày trong tháng,

hh: hai số chỉ giờ trong ngày,

mm: hai số chỉ phút,

CC: hai số chỉ thế kỉ,

YY: hai số chỉ năm trong thế kỉ.

Các dòng ngay dưới đây trình bày một số ví dụ sử dụng lệnh *date*, mỗi ví dụ được cho tương ứng với một cặp hai dòng, trong đó dòng trên mô tả lệnh được gõ còn dòng dưới là thông báo của Linux.

```
# date
Wed Jan 3 23:58:50 ICT 2001
# date -d='01/01/2000'
Sat Jan 1 00:00:00 ICT 2000
# date -iso-8601='seconds'
2000-12-01T00:36:41-0500
```

```
# date -d='01/01/2001'
Mon Jan 1 00:00:00 ICT 2001
# date 010323502001.50
Wed Jan 3 23:50:50 ICT 2001
# date +%a%A
Wed Wednesday
# date +%a%A%b%B
Wed Wednesday Jan January
# date +%D%%j
01/05/01%005
```

5.5 Lệnh xem lịch cal

Lệnh cal cho phép xem lịch trên hệ thống.

Cú pháp như sau: **cal [tùy-chọn] [<tháng> [<năm>]]** nếu không có tham số, lịch của tháng hiện thời sẽ được hiển thị.

Các tùy chọn là:

- m : chọn ngày Thứ hai là ngày đầu tiên trong tuần (mặc định là ngày Chủ nhật).
- j : hiển thị số ngày trong tháng dưới dạng số ngày trong năm (ví dụ: ngày 1/11/2000 sẽ được hiển thị dưới dạng là ngày thứ 306 trong năm 2000, số ngày bắt đầu được tính từ ngày 1/1).
- y : hiển thị lịch của năm hiện thời.

Formatted: Bullets and Numbering

Ví dụ: # cal 1 2001

```
January 2001
     Su    sMo   Tu    We    Th    Fr    Sa
           1     2     3     4     5     6
     7     8     9     10    11    12    13
     14    15    16    17    18    19    20
     21    22    23    24    25    26    27
     28    29    30    31
```

Khi nhập dòng lệnh trên, trên màn hình sẽ hiển thị lịch của tháng 1 năm 2001, mặc định chọn ngày chủ nhật là ngày bắt đầu của tuần. Dưới đây là ví dụ hiển thị số ngày trong tháng 3 dưới dạng số ngày trong năm 2001.

```
# cal -j 3 2001
 March 2001
     Su    Mo    Tu    We    Th    Fr    Sa
                             60    61    62
     63    64    65    66    67    68    69
     70    71    72    73    74    75    76
```

```
77    78    79    80    81    82    83
84    85    86    87    88    89    90
```

5.6 Xem thông tin hệ thống uname

Lệnh uname cho phép xem thông tin hệ thống với cú pháp là: **uname [tùy-chọn]**

ẩ ếu không có tuỳ chọn thì hiện tên hệ điều hành.

Các tùy chọn là:

- _____a, --all : hiện tất cả các thông tin.
- _____m, --machine : kiểu kiến trúc của bộ xử lý (i386, i486, i586, i686...).
- _____n, --nodename : hiện tên của máy.
- _____r, --release : hiện nhân của hệ điều hành.
- _____s, --sysname : hiện tên hệ điều hành.
- _____p, --processor : hiện kiểu bộ xử lý của máy chủ.

Ví dụ: # uname -a thì màn hình sẽ hiện ra như sau:

```
Linux linuxsrv.linuxvn.net 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000
i686 unknown
```

Thông tin hiện ra có tất cả 6 trường là:

Tên hệ điều hành: Linux

Tên máy: linuxsrv.linuxvn.net

Tên nhân của hệ điều hành: 2.2.14-5.0

ẩ gày sản xuất: #1 Tue Mar 7 21:07:39 EST 2000

Kiểu kiến trúc bộ xử lý: i686

Kiểu bộ xử lý của máy chủ: unknown

Ví dụ : nếu gõ lệnh: # uname -spr thì màn hình sẽ hiện ra như sau:

```
Linux 2.2.14-5.0 unknown
```

là tên hệ điều hành, tên nhân và kiểu bộ xử lý của máy chủ.

5.7 Thay đổi nội dung dấu nhắc shell

Trong Linux có hai loại dấu nhắc: dấu nhắc cấp một (dấu nhắc shell) xuất hiện khi nhập lệnh và dấu nhắc cấp hai (dấu nhắc nhập liệu) xuất hiện khi lệnh cần có dữ liệu được nhập từ bàn phím và tương ứng với hai biến nhắc tên là PS1 và PS2.

PS1 là biến hệ thống tương ứng với dấu nhắc cấp 1: Giá trị của PS1 chính là nội dung hiển thị của dấu nhắc shell. Để nhận biết thông tin hệ thống hiện tại, một nhu cầu đặt ra là cần thay đổi giá trị của các biến hệ thống PS1 và PS2.

Formatted: Bullets and Numbering

Linux cho phép thay đổi giá trị của biến hệ thống PS1 bằng lệnh gán trị mới cho nó. Lệnh này có dạng: **# PS1='<dãy kí tự>'**

ẩ ăm (5) kí tự đầu tiên của lệnh gán trên đây (PS1=') phải được viết liên tiếp nhau. Dãy kí tự nằm giữa cặp hai dấu nháy đơn (có thể sử dụng cặp hai dấu kép ") và không được phép chứa dấu nháy. Dãy kí tự này bao gồm các cặp kí tự điều khiển và các kí tự khác, cho phép có thể có dấu cách. Cặp kí tự điều khiển gồm hai kí tự, kí tự đầu tiên là dấu sổ xuôi "\" còn kí tự thứ hai nhận một trong các trường hợp liệt kê trong bảng dưới đây. Bảng dưới đây giới thiệu một số cặp ký tự điều khiển có thể được sử dụng khi muốn thay đổi dấu nhắc lệnh:

Cặp ký tự điều khiển	*Ý nghĩa*
\!	Hiển thị thứ tự của lệnh trong lịch sử
\#	Hiển thị thứ tự của lệnh
\$	Hiển thị dấu đô-la ($). Đối với siêu người dùng (super user), thì hiển thị dấu số hiệu (#)
\\	Hiển thị dấu sổ (\)
\d	Hiển thị ngày hiện tại
\h	Hiển thị tên máy (hostname)
\n	Ký hiệu xuống dòng
\s	Hiển thị tên hệ shell
\t	Hiển thị giờ hiện tại
\u	Hiển thị tên người dùng
\W	Hiển thị tên thực sự của thư mục hiện thời (ví dụ thư mục hiện thời là /mnt/hda1 thì tên thực sự của nó là /hda1)
\w	Hiển thị tên đầy đủ của thư mục hiện thời (ví dụ /mnt/hda1)

Ví dụ: hiện thời dấu nhắc shell có dạng: root[may1 /hda1]#

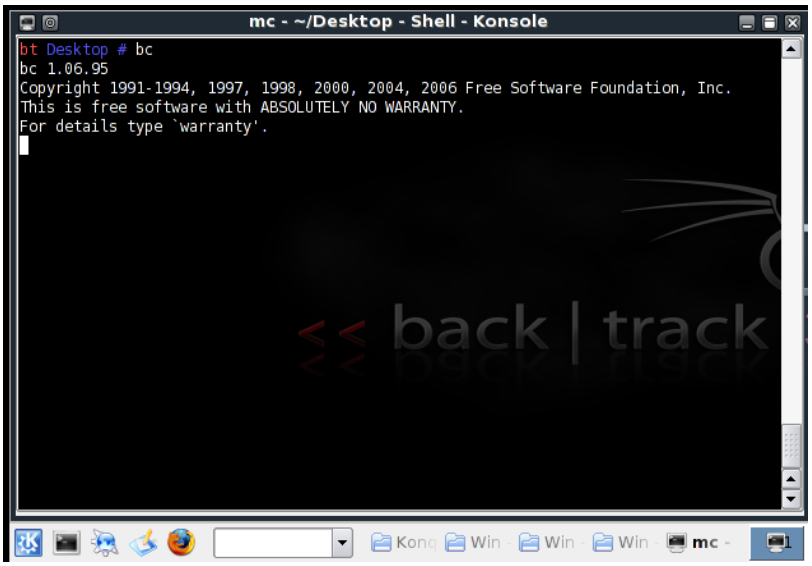
Sau khi gõ lệnh: **root@may1 /hda1]# PS1='[\h@\u \w : \d]\$'**

thì dấu nhắc shell được thay đổi là: **[may1@root /mnt/hda1 : Fri Oct 27]#** ngoài việc đổi thứ tự giữa tên người dùng và máy còn cho chúng ta biết thêm về ngày hệ thống quản lý và tên đầy đủ của thư mục hiện thời.

Linux cung cấp cách thức hoàn toàn tương tự như đối với biến PS1 để thay đổi giá trị biến hệ thống PS2 tương ứng với dấu nhắc cấp hai.

5.8 Lệnh gọi ngôn ngữ tính toán số học

Linux cung cấp một ngôn ngữ tính toán với độ chính xác tùy ý thông qua lệnh *bc*. Khi yêu cầu lệnh này, người dùng được cung cấp một ngôn ngữ tính toán (và cho phép lập trình tính toán có dạng ngôn ngữ lập trình C) hoạt động theo thông dịch.



Trong ngôn ngữ lập trình được cung cấp (tạm thời gọi là ngôn ngữ *bc*), tồn tại rất nhiều công cụ hỗ trợ tính toán và lập trình tính toán: kiểu phép toán số học phong phú, phép toán so sánh, một số hàm chuẩn, biến chuẩn, cấu trúc điều khiển, cách thức định nghĩa hàm, cách thức thay đổi độ chính xác, đặt lời chú thích ... Chỉ cần sử dụng một phần nhỏ tác động của lệnh *bc*, chúng ta đã có một "máy tính số bấm tay" hiệu quả.

Cú pháp lệnh bc: **bc [tùy-chọn] [file...]** với các tuỳ chọn sau đây:

- -l, --mathlib: thực hiện phép tính theo chuẩn thư viện toán học (ví dụ: 5/5=1.00000000000000000000).
- -w, --warn : khi thực hiện phép tính không tuân theo chuẩn POSIX (POSIX là một chuẩn trong Linux) thì một cảnh báo xuất hiện.
- -s, --standard : thực hiện phép tính chính xác theo chuẩn của ngôn ngữ POSIX bc.
- -q, --quiet : không hiện ra lời giới thiệu về phần mềm Gả U khi dùng bc.

Formatted: Bullets and Numbering

Tham số file là tên file chứa chương trình viết trên ngôn ngữ *bc*, khi lệnh *bc* thực hiện sẽ tự động chạy các file chương trình này (ả ếu có nhiều tham số thì có nghĩa sẽ chạy nhiều chương trình liên tiếp nhau).

Ví dụ: Từ dấu nhắc lệnh gõ: **# bc** sẽ xuất hiện dấu nhắc, sau đó ta gõ biểu thức vào:

```
(4+5)*(12-10) ↵
18
```

```
1000000000000*1000000000000↵
100000000000000000000000
```

để ấn định số chữ số thập phân dùng lệnh scale = n

```
scale=3 ↵
1/6 ↵
.166
```

Ví dụ: Lập trình trong bc:

```
define giaithua(n)
{
      if (n<=1) return (1);
      else return (gt(n-1)*n);
}
gt(5)
120
```

để chuyển sang các cơ số khác nhau dùng lệnh ibase và obase

ibase=cơ số Định dạng cơ số đầu vào

obase=cơ số Định dạng cơ số đầu ra

ibase và obase ngầm định là cơ số 10.

```
ibase=16↵
FF↵
255
obase=2↵
FF↵
11111111
ibase↵
obase↵
```

Để kết thúc bc gõ CTRL + D.

Chú ý: ả gôn ngữ lập trình tính toán *bc* là một ngôn ngữ rất mạnh có nội dung hết sức phong phú cho nên trong khuôn khổ của tài liệu này không thể mô tả hết các nội dung của ngôn ngữ đó được. Chúng ta cần sử dụng lệnh *man bc* để nhận được thông tin đầy đủ về lệnh *bc* và ngôn ngữ tính toán *bc.*

Ở đây trình bày sơ bộ một số yếu tố cơ bản nhất của ngôn ngữ đó (*bt* là viết tắt của biểu thức, *b* là viết tắt của biến).

Formatted: Bullets and Numbering

Các phép tính: - bt: lấy đối; ++ b, --b, b ++, b --: phép toán tăng, giảm b; các phép toán hai ngôi cộng +, trừ -, nhân *, chia /, lấy phần dư %, lũy thừa nguyên bậc ^; gán =; gán sau khi thao tác <thao tác>=; các phép toán so sánh <, <=, >, >=, bằng ==, khác != ...
Phép so sánh cho 1 nếu đúng, cho 0 nếu sai.

Bốn biến chuẩn là *scale* số lượng chữ số phần thập phân, *last* giá trị tính toán cuối cùng; ibase cơ số hệ đếm đối với *input* và *obase* là cơ số hệ đếm với *output* (ngầm định hai biến này có giá trị 10).

Các hàm chuẩn sin s (bt); cosin c (bt); arctg a (bt); lôgarit tự nhiên l (bt); mũ cơ số tự nhiên e (bt); hàm Bessel bậc nguyên n của bt là j (n, bt).

5.9 Tiện ích mc

Tiện ích mc trong Linux cũng giống như ả C Command của MS-DOS.

ả gười sử dụng hệ điều hành MS-DOS đều biết tính năng tiện ích ả orton Commander (ả C) rất mạnh trong quản lý, điều khiển các thao tác về file, thư mục, đĩa cũng như là môi trường trực quan trong chế độ văn bản (text). Dù trong hệ điều hành Windows sau này đã có sự hỗ trợ của tiện ích Explorer nhưng không vì thế mà vai trò của ả C giảm đi: ả hiều người dùng vẫn thích dùng ả C trong các thao tác với file và thư mục. Linux cũng có một tiện ích mang tên Midnight Commander (viết tắt là MC) có chức năng và giao diện gần giống với ả C của MS-DOS và sử dụng MC trong Linux tương tự như sử dụng ả C trong MS-DOS.

Khởi động MC

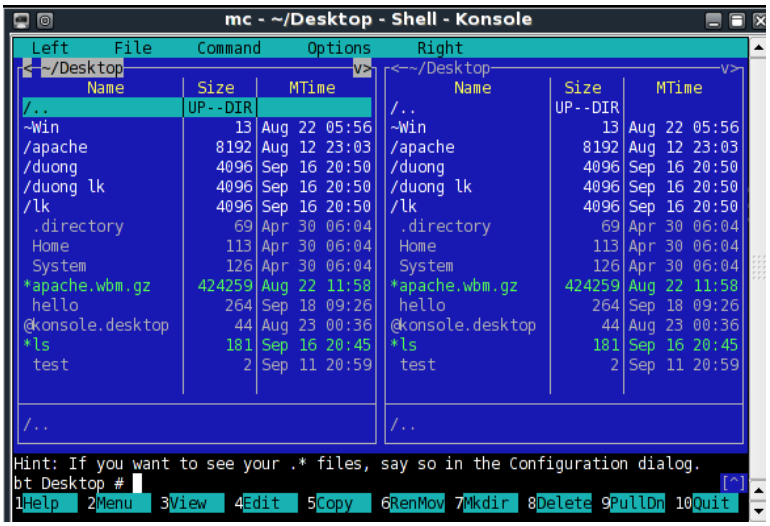
Lệnh khởi động MC: **# mc [Tùy-chọn]**

Có một số tuỳ chọn khi dùng tiện ích này theo một số dạng thông dụng sau:

-a	Không sử dụng các ký tự đồ hoạ để vẽ các đường thẳng khung.
-b	Khởi động trong chế độ màn hình đen trắng.
-c	Khởi động trong chế độ màn hình màu.
-d	Không hỗ trợ chuột
-P	Với tham số này, Midnight Commander sẽ tự động chuyển thư mục hiện hành tới thư mục đang làm việc. ả hư vậy, sau khi kết thúc, thư mục hiện hành sẽ là thư mục cuối cùng thao tác.
-v file	Sử dụng chức năng View của MC để xem nội dung của file được chỉ ra.
-V	Cho biết phiên bản chương trình đang sử dụng.

ả ếu chỉ ra đường dẫn (path), đường dẫn đầu tiên là thư mục được hiển thị trong panel chọn (selected panel), đường dẫn thứ hai được hiển thị panel còn lại.

Giao diện của MC



Giao diện của MC được chia ra làm bốn phần. Phần lớn màn hình là không gian hiển thị của hai panel. Panel là một khung cửa sổ hiển thị các file thư mục cùng các thuộc tính của nó hoặc một số nội dung khác. Theo mặc định, dòng thứ hai từ dưới lên sẽ là dòng lệnh còn dòng dưới cùng hiển thị các phím chức năng. Dòng đầu tiên trên đỉnh màn hình là thực đơn ngang (menu bar) của MC. Thanh thực đơn này có thể không xuất hiện nhưng nếu kích hoạt bằng cả hai chuột tại dòng đầu tiên hoặc nhấn phím <F9> thì nó sẽ hiện ra và được kích hoạt.

Midnight Commander cho phép hiển thị cùng một lúc cả hai panel. Một trong hai panel là panel hiện hành (panel chọn). Thanh sáng chọn nằm trên panel hiện hành. Hầu hết các thao tác đều diễn ra trên Panel này. Một số các thao tác khác về file như Rename hay Copy sẽ mặc định sử dụng thư mục ở Panel còn lại làm thư mục đích. Tuy nhiên ta vẫn có thể sửa được thư mục này trước khi thao tác vì các thao tác này đầu tiên bao giờ cũng yêu cầu nhập đường dẫn.

Trên panel sẽ hiển thị hầu hết các file và thư mục con của thư mục hiện hành. Midnight Commander có cơ chế hiển thị các kiểu file khác nhau bằng các ký hiệu và màu sắc khác nhau, ví dụ như các file biểu tượng liên kết sẽ có ký hiệu '@' ở đầu, các file thiết bị sẽ có màu đỏ tím, các file đường ống có màu đen, các thư mục có ký hiệu '/' ở đầu, các thư mục liên kết có ký hiệu '~'...

Cho phép thi hành một lệnh hệ thống từ MC bằng cách gõ chúng lên màn hình. Tất cả những gì có gõ vào đều được hiển thị ở dòng lệnh phía dưới trừ một số ký tự điều khiển và khi nhấn Enter, Midnight Commander sẽ thi hành lệnh gõ vào.

Dùng chuột trong MC

Midnight Commander sẽ hỗ trợ chuột trong trường hợp không gọi với tham số **-d**. Khi kích chuột vào một file trên Panel, file đó sẽ được chọn, có nghĩa là thanh sáng chọn sẽ nằm tại vị trí file đó và panel chứa file đó sẽ trở thành panel hiện hành. Còn nếu kích chuột phải vào một file, file đó sẽ được đánh dấu hoặc xoá dấu tuỳ thuộc vào trạng thái kích trước đó.
ẩ ếu kích đôi chuột tại một file, file đó sẽ được thi hành nếu đó là file thi hành được (executable program) hoặc nếu có một chương trình đặc trưng cho riêng phần mở rộng đó thì chương trình đặc trưng này sẽ được thực hiện.

ẩ gười dùng cũng có thể thực hiện các lệnh của các phím chức năng bằng cách nháy chuột lên phím chức năng đó.

ẩ ếu kích chuột tại dòng đầu tiên trên khung panel, toàn bộ panel sẽ bị kéo lên. Tương tự kích chuột tại dòng cuối cùng trên khung panel, toàn bộ panel sẽ bị kéo xuống.

Có thể bỏ qua các thao tác chuột của MC và sử dụng các thao tác chuột chuẩn bằng cách giữ phím <Shift>

Các thao tác bàn phím

Một số thao tác của Midnight Commander cho phép sử dụng nhanh bằng cách gõ các phím tắt (hot key). Để tương thích với một số hệ thống khác, trong các bảng dưới đây về Midnight Commander, viết tắt phím CTRL là "C", phím ALT là "M" (Meta), phím SHIFT là "S".

Các ký hiệu tổ hợp phím có dạng như sau:

C-<chr>	Có nghĩa là giữ phím CTRL trong khi gõ phím <char>. Ví dụ C -f có nghĩa là giữ CTRL và nhấn <f>.
C-<chr1><char2>	Có nghĩa là giữ phím CTRL trong khi gõ phím <char1> sau đó nhả tất cả ra và gõ phím <char2>.
M-<chr>	Có nghĩa là giữ phím ALT trong khi gõ phím <char>. ẩ ếu không có hiệu lực thì có thể thực hiện bằng cách gõ phím <Esc> nhả ra rồi gõ phím <char>.
S-<chr>	Có nghĩa là giữ phím SHIFT trong khi gõ phím <char>.

Sau đây là chức năng một số phím thông dụng. Các phím thực hiện lệnh:

Enter	ẩ ếu có dòng lệnh, lệnh đó sẽ được thi hành. Còn nếu không thì sẽ tuỳ vào vị trí của thanh sáng trên panel hiện hành là file hay thư

	mục mà hoặc việc chuyển đổi thư mục hoặc thi hành file hay thi hành một chương trình tương ứng sẽ diễn ra.
C-l	Cập nhật lại các thông tin trên Panel.

Các phím thao tác trên dòng lệnh:

M-Enter hay C-Enter	chép tên file ở vị trí thanh sáng chọn xuống dòng lệnh
M-Tab	hoàn thành tên file, lệnh, biến, tên người dùng hoặc tên máy giúp
C-x t, C-x C-t	sao các file được đánh dấu (mặc định là file hiện thời) trên panel chọn (C-x t) hoặc trên panel kia (C-x C-t) xuống dòng lệnh
C-x p, C-x C-p	đưa tên đường dẫn hiện thời trên panel chọn (C-x p) hoặc trên panel kia (C-x C-p) xuống dòng lệnh
M-p, M-n	sử dụng để hiện lại trên dòng lệnh các lệnh đã được gọi trước đó. M-p sẽ hiện lại dòng lệnh được thi hành gần nhất, M-n hiện lại lệnh được gọi trước lệnh đó
C-a	đưa dấu nhắc trỏ về đầu dòng
C-e	đưa dấu nhắc trỏ về cuối dòng
C-b, Left	đưa dấu nhắc trỏ di chuyển sang trái một ký tự
C-f, Right	đưa dấu nhắc trỏ di chuyển sang phải một ký tự
M-f	đưa dấu nhắc trỏ đến từ tiếp theo
M-b	đưa dấu nhắc trỏ ngược lại một từ
C-h, Space	xoá ký tự trước đó
C-d, Delete	xoá ký tự tại vị trí dấu nhắc trỏ
C-@	đánh dấu để cắt
C-k	xoá các ký tự từ vị trí dấu nhắc trỏ đến cuối dòng
M-C-h, M-Backspace	xoá ngược lại một từ

Các phím thao tác trên panel:

Up,Down, PgUp, PgDown, Home, End	sử dụng các phím này để di chuyển trong một panel
b, C-b, C-h, Backspace, Delete	di chuyển ngược lại một trang màn hình

Space	di chuyển tiếp một trang màn hình
u, d	di chuyển lên/ xuống 1/2 trang màn hình
g, G	di chuyển đến điểm đầu hoặc cuối của một màn hình
Tab, C-i	hoán đổi panel hiện hành. Thanh sáng chọn sẽ chuyển từ panel cũ sang panel hiện hành
Insert, C-t	chọn đánh dấu một file hoặc thư mục
M-g, M-h, M-j	lần lượt chọn file đầu tiên, file giữa và file cuối trên panel hiển thị
	tìm kiếm file trong thư mục. Khi kích hoạt chế độ này, những ký tự gõ vào sẽ được thêm vào xâu tìm kiếm thay vì hiển thị trên dòng lệnh. ẩ ếu tuỳ chọn Show mini-status trong option được đặt thì xâu tìm kiếm sẽ được hiển thị ở dòng trạng thái.
C-s, M-s	Khi gõ các kí tự, thanh sáng chọn sẽ di chuyển đến file đầu tiên có những ký tự đầu giống những ký tự gõ vào. Sử dụng phím Backspace hoặc Del để hiệu chỉnh sai sót. ẩ ếu nhấn C-s lần nữa, việc tìm kiếm sẽ được tiếp tục
M-t	chuyển đổi kiểu hiển thị thông tin về file hoặc thư mục
C-\	thay đổi thư mục hiện thời
+	sử dụng dấu cộng để lựa chọn đánh dấu một nhóm file. Có thể sử dụng các ký tự đại diện như '*', '?'... để biểu diễn các file sẽ chọn
ữ	sử dụng dấu trừ để xoá đánh dấu một nhóm file. Có thể sử dụng các ký tự dại diện như '*', '?' để biểu diễn các file sẽ xoá
*	sử dụng dấu * để đánh dấu hoặc xoá đánh dấu tất cả các file trong panel
M-o	một panel sẽ hiển thị nội dung thư mục hiện thời hoặc thư mục cha của thư mục hiện thời của panel kia
M-y	di chuyển đến thư mục lúc trước đã được sử dụng
M-u	di chuyển đến thư mục tiếp theo đã được sử dụng

Thực đơn thanh ngang (menu bar)

Thực đơn thanh ngang trong Midnight Commander được hiển thị ở dòng đầu tiên trên màn hình. Mỗi khi nhấn <F9> hoặc kích chuột tại dòng dầu tiên trên màn hình thực đơn ngang sẽ được kích hoạt. Thực đơn ngang của MC có năm mục "Left", "File", "Command", "Option" và "Right".

Thực đơn Left và Right giúp ta thiết lập cũng như thay đổi kiểu hiển thị của hai panel left và right. Các thực đơn mức con của chúng gồm:

Listing Mode ...		thực đơn này được dùng khi muốn thiết lập kiểu hiển thị của các file. Có bốn kiểu hiển thị: Full - hiển thị thông tin về tên , kích thước, và thời gian sử đổi của file; Brief - chỉ hiển thị tên của file; Long - hiển thị thông tin đầy đủ về file (tương tự lệnh ls -l); User - hiển thị các thông tin do tự chọn về file;
Quick view	C-x q	xem nhanh nội dung của một file
Info	C-x i	xem các thông tin về một thư mục hoặc file
Tree		hiển thị dưới dạng cây thư mục
Sort order...		thực hiện sắp xếp nội dung hiển thị theo tên, theo tên mở rộng, thời gian sửa chữa, thời gian truy nhập, thời gian thay đổi, kích thước, inode
Filter ...		thực hiện việc lọc file theo tên
ẩ etwork link ...		thực hiện liên kết đến một máy tính
FTP link ...		thực hiện việc lấy các file trên các máy từ xa
Rescan	C-r	quét lại

Thực đơn File chứa một danh sách các lệnh mà có thể thi hành trên các file đã được đánh dấu hoặc file tại vị trí thanh chọn. Các thực đơn mức con:

User menu	F2	thực đơn dành cho người dùng
View	F3	xem nội dung của file hiện thời
View file ...		mở và xem nội dung của một file bất kì
Filtered view	M-!	thực hiện một lệnh lọc với tham số là tên file và hiển thị nội dung của file đó
Edit	F4	soạn thảo file hiện thời với trình soạn thảo mặc định trên hệ thống
Copy	F5	thực hiện copy
cHmod	C-x c	thay đổi quyền truy nhập đối với một thư mục hay một file

Link	C-x l	tạo một liên kết cứng đến file hiện thời
Symlink	C-x s	tạo một liên kết tượng trưng đến file hiện thời
edit sYimlink	C-x C-s	hiệu chỉnh lại một liên kết tượng trưng
chOwn	C-x o	thay đổi quyền sở hữu đối với thư mục hay file
Advanced chown		thay đổi quyền sở hữu cũng như quyền truy nhập của file hay thư mục
Rename/Move	F6	thực hiện việc đổi tên hay di chuyển đối với một file
Mkdir	F7	tạo một thư mục
Delete	F8	xoá một hoặc nhiều file
Quick cd	M-c	chuyển nhanh đến một thư mục
select Group	M-+	thực hiện việc chọn một nhóm các file
Unselect group	M-ữ	ngược với lệnh trên
reverse selecTion	M-*	chọn các file trong thư mục hiện thời
Exit	F10	thoát khỏi MC

Thực đơn Command cũng chứa một danh sách các lệnh.

Directory tree		hiển thị thư mục dưới dạng cây thư mục
Find file	M-?	tìm một file
Swap panels	C-u	thực hiện tráo đổi nội dung giữa hai panel hiển thị
Switch panels on/of	C-o	đưa ra lệnh shell được thực hiện lần cuối (chỉ sử dụng trên xterm, trên console SCO và Linux)
Compare directories	C-x d	thực hiện so sánh thư mục hiện tại trên panel chọn với các thư mục khác
Command history		đưa ra danh sách các lệnh đã thực hiện
Directory hotlist	C-\	thay đổi thư mục hiện thời
External panelize	C-x !	thực hiện một lệnh trong MC và hiển thị kết quả trên panel chọn (ví dụ: nếu muốn trên panel chọn hiển thị tất cả các file liên kết trong thư mục hiện thời, hãy chọn mục thực đơn này và nhập lệnh find . -type l -print sẽ thấy kết quả thật tuyệt vời)
Show directory size		hiển thị kích thước của thư mục
Command history		hiển thị danh sách các lệnh đã thực hiện
Directory hotlist	C-\	chuyển nhanh đến một thư mục
Background	C-x j	thực hiện một số lệnh liên quan đến các quá trình nền

Extension file edit	cho phép hiệu chỉnh file ~/.mc/ext để xác định chương trình sẽ thực hiện khi xem, soạn thảo hay làm bất cứ điều gì trên các file có tên mở rộng

Thực đơn Options cho phép thiết lập, huỷ bỏ một số tuỳ chọn có liên quan đến hoạt động của chương trình MC.

Configuration ...	thiết lập các tuỳ chọn cấu hình cho MC
Lay-out ...	xác lập cách hiển thị của MC trên màn hình
Confirmation ...	thiết lập các hộp thoại xác nhận khi thực hiện một thao tác nào đó
Display bits ...	thiết lập cách hiển thị của các ký tự
Learn keys ...	xác định các phím không được kích hoạt
Virtual FS ...	thiết lập hệ thống file ảo
Save setup	ghi mọi sự thiết lập được thay đổi

Các phím chức năng

Các phím chức năng của Midnight Commander được hiển thị tại dòng cuối cùng của màn hình. Có thể thực hiện các chức năng đó bằng cách kích chuột lên nhãn của các chức năng tương ứng hoặc nhấn trên bàn phím chức năng đó.

F1	hiển thị trang trợ giúp
F2	đưa ra thực đơn người dùng
F3	xem nội dung một file
F4	soạn thảo nội dung một file
F5	thực hiện sao chép file
F6	thực hiện di chuyển hoặc đổi tên file
F7	tạo thư mục mới
F8	xoá thư mục hoặc file
F9	đưa trỏ soạn thảo lên thanh thực đơn nằm ngang
F10	thoát khỏi MC

Bộ soạn thảo của Midnight Commander

Midnight Commander cung cấp một bộ soạn thảo khá tiện dụng trong việc soạn thảo các văn bản ASCII. Bộ soạn thảo này có giao diện và thao tác khá giống với tiện ích Edit của DOS hay ả cEdit của ả orton Commander. Để hiệu chỉnh một số file văn bản, hãy di chuyển thanh

sáng chọn đến vị trí file đó rồi nhấn F4, nội dung của file đó sẽ hiện ra trong vùng soạn thảo. Sau khi hiệu chỉnh xong, nhấn F2 để ghi lại. Bộ soạn thảo này có một thực đơn ngang cung cấp các chức năng đầy đủ như một bộ soạn thảo thông thường.

ẩ ếu đã từng là người dùng DOS và mới dùng Linux thì nên dùng bộ soạn thảo này để hiệu chỉnh và soạn thảo văn bản thay vì bộ soạn thảo Vim. Sau đây là bảng liệt kê các phím chức năng cũng như các mức thực đơn trong bộ soạn thảo này:

Thực đơn File các thao tác liên quan đến file.

Open/load	C-o	mở hoặc nạp một file
ẩ ew	C-n	tậo một file mới
Save	F2	ghi nội dung file đã được soạn thảo
Save as ...	F12	tạo một file khác tên nhưng có nội dung trùng với nội dung file hiện thời
Insert file ...	F15	chèn nội dung một file vào file hiện thời
Copy to file ...	C-f	sao đoạn văn bản được đánh dấu đến một file khác
About ..		thông tin về bộ soạn thảo
Quit	F10	thoát khỏi bộ soạn thảo

Thực đơn Edit: các thao tác liên quan đến việc soạn thảo nội dung file.

Toggle Mark	F3	thực hiện đánh dấu một đoạn văn bản
Mark Columns	S-F3	đánh dấu theo cột
Toggle Ins/overw	Ins	chuyển đổi giữa hai chế độ chèn/đè
Copy	F5	thực hiện sao chép file
Move	F6	thực hiện di chuyển file
Delete	F8	xoá file
Undo	C-u	trở về trạng thái trước khi thực hiện một sự thay đổi
Beginning	C-PgUp	di chuyển đến đầu màn hình
End	C-PgDn	di chuyển đến cuối màn hình

Thực đơn Sear/Repl: các thao tác liên quan đến việc tìm kiếm và thay thế

Search ..	F7	thực hiện tìm kiếm một xâu văn bản
Search again	F17	tìm kiếm tiếp
Replace ...	F4	tìm và thay thế xâu văn bản

Thực đơn Command: các lệnh có thể được thực hiện trong khi soạn thảo.

Goto line ...	M-l	di chuyển trỏ soạn thảo đến một dòng
Insert Literal ...	C-q	chèn vào trước dấu nhắc trỏ một ký tự
Refresh screen	C-l	làm tươi lại màn hình
Insert Date/time		chèn ngày giờ hiện tại vào vị trí dấu nắhc trỏ
Format paragraph	M-p	định dạng lại đoạn văn bản
Sort	M-t	thực hiện sắp xếp

Thực đơn Options: các tuỳ chọn có thể thiết lập cho bộ soạn thảo.

General ...	thiết lập các tuỳ chọn cho bộ soạn thảo
Save mode ...	ghi lại mọi sự thiết lập được thay đổi

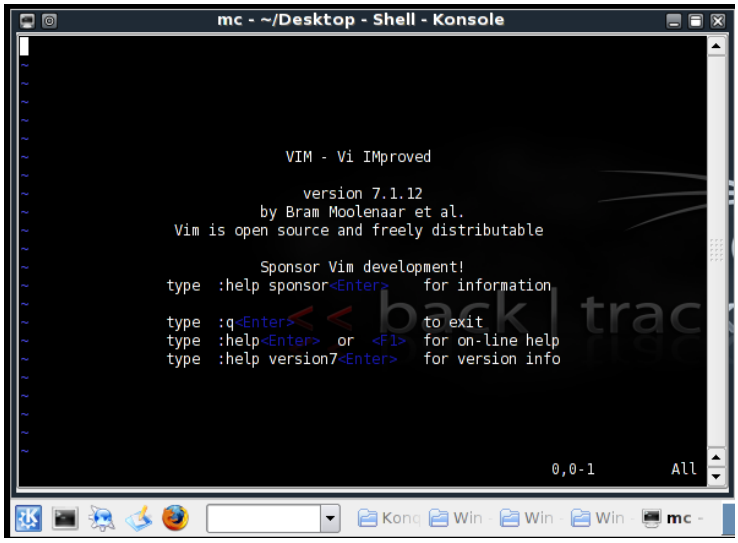
Các phím chức năng

F1	hiển thị trang trợ giúp
F2	ghi nội dung file
F3	thực hiện việc đánh dấu đoạn văn bản
F4	tìm và thay thế xâu văn bản
F5	thực hiện việc sao chép
F6	di chuyển file
F7	tìm kiếm xâu văn bản
F8	xoá đoạn văn bản được đánh dấu
F9	hiển thị thanh thực đơn ngang
F10	thoát khỏi bộ soạn thảo

5.10 Sử dụng trình soạn thảo VI

VI là chương trình soạn thảo văn bản theo trang màn hình:

- Màn hình được xem như một cửa sổ mở trên file.
- Có khả năng di chuyển cursor tới bất kỳ nơi nào trên màn hình.
- Cửa sổ có thể di chuyển tự do trên file.



Để hiển thị đúng, VI cần biết kiểu terminal đang dùng. Ta có thể định nghĩa được kiểu terminal bằng cách gán gía trị cho biến môi trường TERM: $TERM=tws2103; export TERM

Phần lớn các phím được dùng độc lập hoặc kết hợp với phím SHIFT và CTRL để tạo các lệnh của VI. Khi một lệnh bị gõ sai, vi báo hiệu bằng nháy màn hình, kêu beep hoặc thông báo lỗi.

Chương trình VI được xây dựng từ chương trình soạn thảo dòng ex. Các lệnh của ex có thể được gọi khi có dấu ":" ở dòng cuối màn hình.

Ta có thể gọi vi với tên file văn bản: $ vi tên_file

Cửa sổ soạn thảo sẽ được mở tại đầu file. ả ếu file chưa tồn tại, nó sẽ được tạo bởi lệnh ghi. Dòng cuối cùng trên màn hình được dùng cho những việc sau:

- vào các lệnh.
- thống kê.
- báo lỗi.

Đối với những người mới dùng vi, có thể dùng version khác của vi: **$vedit tên_file** version này của vi sẽ hiện thông báo Iả PUT MODE khi ta đang trong chế độ nhập văn bản. Khi

ta chỉ muốn xem nội dung của một file, dùng: **$view tên_file** version này của vi mở file chỉ để đọc, cho phép ta xem được nội dung mà tránh được nguy cơ file bị thay đổi.

Chuyển chế độ làm việc: Từ chế độ soạn thảo sang chế độ lệnh dùng phím ESC

Muốn ra khỏi vi và ghi file có thể dùng một trong các cách sau:

```
ZZ              hoặc
:w   sau đó:q   hoặc
:wq             hoặc
:x
```

Ra khỏi VI và không ghi file:

```
:q (nếu không có sửa đổi) hoặc
:q!
```

Khi đang trong VI, muốn làm việc với SHELL, ta có thể làm như sau:

- chạy một lệnh của SHELL

```
:!lệnh
```

- hoặc gọi SHELL, sau đó chạy các lệnh ta muốn, khi kết thúc ấn CTRL-D để trở lại VI:

```
:!sh
$lệnh
$CTRL-D
```

Chèn văn bản

Chèn ký tư trên môt dòng

a <text> <ESC>	Chèn ký tự vào sau cursor.
i <text> <ESC>	Chèn ký tự vào trước cursor.
A <text> <ESC>	Chèn ký tự vào cuối dòng.
I <text> <ESC>	Chèn ký tự vào đầu dòng.

Chèn dòng

o <text> <ESC>	Chèn một dòng vào trước dòng chứa cursor.
O <text> <ESC>	Chèn một dòng vào sau dòng chứa cursor.

Ghi chú: nhấn ESC để kết thúc chế độ xem, muốn chèn các ký tự không in được ta phải gõ: CTRL – V trước chúng.

Di chuyển cursor trong file

Theo ký tư

Sang trái: dùng phím mũi tên trái hoặc h hoặc backspace.

Xuống dòng: dùng phím mũi tên xuống hoặc j hoặc linefeed

Sang phải: dùng phím mũi tên phải hoặc i hoặc escape.

Lên dòng: dùng phím mũi tên lên hoặc k.

Theo dòng

^ về đầu dòng

$ cuối dòng

Enter đầu dòng tiếp

Đầu dòng trên

0(null) về đầu dòng vật lý (dòng bắt đầu bằng dấu cách hoặc tab)

Theo màn hình

H về đầu màn hình (Home)

M về giữa màn hình (Middle)

L về cuối màn hình (Last)

Theo từ (word)

w W về đầu từ tiếp

b B đầu từ hiện tại

e E cuối từ hiện tại

Theo câu (sentence)

(về đầu câu

) về cuối câu

dấu kết thúc một câu là các dấu ., ! hoặc ?

Theo đoạn văn (paragraph)

{ về đầu đoạn văn

} cuối đoạn văn

đoạn văn kết thúc bằng một dòng trống.

Theo cửa sổ (window)

z dòng hiện tại ở giữa cửa sổ.

z<Enter> dòng hiện tại ở đầu cửa sổ.

z- dòng hiện tại ở cuối cửa sổ.

^D xuống nửa cửa sổ

^U lên nửa cửa sổ

^F xuống một cửa sổ (-2 dòng)

^B lên một cửa sổ (2 dòng)

Ghi chú: ^ là ký hiệu của phím CTRL

Theo số thứ tự dòng Để hiển thị số thứ tự của các dòng soạn thảo: **:set nu**

Xoá bỏ hiển thị trên

:set nonu

:n <Enter> hoặc nG chuyển cursor đến dòng thứ n

:$ hoặc G đến dòng cuối văn bản

:se list hiển thị các ký tự ẩn (hidden)

Tìm dãy ký tự

/ ký hiệu chiều tìm xuôi.

? ký hiệu chiều tìm ngược.

/string chuyển cursor tới dòng chứa dãy ký tự theo chiều xuôi.

?string chuyển cursor tới dòng chứa dãy ký tự theo chiều ngược.

// lặp lại tìm xuôi.

?? lặp lại tìm ngược.

Xóa văn bản

Xóa ký tự

x xóa ký tự tại vị trí cursor

3x xóa 3 ký tự

X xóa ký tự trước vị trí cursor

Xóa dòng văn bản

dd hoặc d<CR> xóa dòng chứa cursor

3dd xóa 3 dòng bắt đầu từ dòng chứa cursor

d$ hoặc D xóa đến cuối dòng

dw xoá từ chứa cursor

3dw hoặc d3w xoá 3 từ

d/string xóa khi hết dãy string

Thay thế văn bản

Thay thế ký tự

rc thay thế ký tự hiện tại bằng ký tự c (???)

R<text><ESC> thay thế số ký tự bằng dãy “text”

Thay thế dòng

S<text><ESC> xóa dòng hiện tại và thay nó bằng “text”

Thay thế từ

cw<text><ESC> thay một từ bằng “text”. Từ được thay thế tính từ cursor đến ký tự $.

c2w<text><ESC> thay 2 từ.

C hoặc c$ thay thế cuối dòng

c/string thay thế đến hết ”string”

Xóa lệnh

u xóa tác dụng của lệnh cuối cùng

U xoá tất cả thay đổi đã làm trên dòng hiện tại.

Xem trạng thái văn bản đang soạn thảo

^G Hiển thị tên, trạng thái, số dòng, vị trí ,cursor và phần trăm văn bản tính từ vị trí cursor đến cuối văn bản.

Sao chép, di chuyển văn bản

Di chuyển văn bản

Mỗi lần thực hiện một lệnh xóa (x hoặc d), vi đều ghi lại phần văn bản bị xóa vào vùng đệm riêng cho đến lần xóa sau. Lệnh p và P cho phép lấy lại văn bản từ vùng đệm đó. Trước khi thực hiện lệnh này, cursor phải được đặt vào vị trí cùng kiểu với phần văn bản có trong vùng đệm:

- ký tự.
- từ.
- dòng.
- cuối dòng (end of line).

p sao phần văn bản xoá lần cuối cùng vào sau đối tượng trong cùng kiểu.

P sao phần văn bản xoá lần cuối vào trước đối tượng cùng kiểu.

Sao chép văn bản

Lệnh y (yank) cho phép sao phần văn bản ta muốn vào vùng đệm. Muốn sao phần văn bản từ vùng đệm ra, ta phải chuyển cursor vào nơi cần sao, sau đó dùng p hoặc P.

Y3w sao 3 từ vào vùng đệm.

Y hoặc yy sao dòng hiện tại vào vùng đệm.

5yy sao 5 dòng vào vùng đệm.

Một cách khác để sao chép dòng

:5,8t25 sao các dòng từ 5 đến 8 tới sau dòng 25

5.11 Sử dụng tài liệu giúp đỡ man

Trong DOS để biết cú pháp hay ý nghĩa của một lệnh chúng ta hay dùng giúp đỡ của lệnh bằng cách đánh tham số /? vào phía sau lệnh, còn Window có bộ Help cho phép ta tìm kiếm các thông tin liên quan đến một vấn đề nào đó.

Linux thì cung cấp cho ta một hệ thống thư viện giúp đỡ cho phép ta tìm các thông tin theo từ khóa ta nhập vào. Dù không có giao diện bằng Window, nhưng các tài liệu giúp đỡ này rất có ích đối với người sử dụng đặc biệt khi sử dụng các lệnh. Chúng ta sẽ biết các lệnh trong Linux sử dụng rất nhiều tùy chọn mà chúng ta không thể nhớ hết được. *Man* sẽ giúp chúng ta.

Chúng ta sử dụng *man* theo cú pháp: **$man từ-khóa [Enter]**

từ-khóa là từ mà chúng ta cần tìm kiếm thông tin về nó.

Ví dụ: Tìm kiếm các thông tin về lệnh *ls*

```
$man ls
LS(1)                          FSF                 LS(1)
NAME
ls - list directory contents
SYNOPSIS
ls [OPTION]... [FILE]...
DESCRIPTION
List information about the FILEs (the current directory by
default).  Sort entries alphabetically if none of –cftuSUX nor --
sort.
-a, --all
    do not hide entries starting with .
-A, --almost-all
    do not list implied . and ..
-b, --escape
    print octal escapes for nongraphic characters
--block-size=SIZE :
```

Ta dùng phép điều khiển lên, xuống để xem trang *man*. ả ếu muốn xem từng trang dùng phím *space***.**

Thoát khỏi *man* sử dụng lệnh: **: q**

Man phân dữ liệu mình lưu trữ thành những đoạn (session) khác nhau với các chủ đề khác nhau là:

Session	*Tên chủ đề*	*Ý nghĩa*
1	user command	các lệnh thông thường của hệ điều hành
2	system call	các hàm thư viện kernel của hệ thống
3	subroutines	các hàm thư viện lập trình
4	devices	các hàm truy xuất file và xử lý thiết bị

5	File format	các hàm định dạng file
6	games	các hàm liên quan đến trò chơi
7	Miscell	các hàm khác
8	sys. admin	các hàm quản trị hệ thống

Xác định cụ thể thông tin của một chủ đề nào chúng ta dùng: **$man session từ-khóa**

Ví dụ : **# man 3 printf**

Xem các thông tin về hàm *prinf* dùng trong lập trình. ẩ ếu chúng ta không xác định session thì session mặc nhiên là 1 .

CHƯƠNG 4: LẬP TRÌNH TRONG LINUX

1. LẬP TRÌNH SHELL

1.1 Khái niệm shell

Shell là chương trình luôn được thực thi khi chúng ta đăng nhập hệ thống. ả ó là chương trình cho phép chúng ta tương tác với hệ thống. Hiện tại có nhiều shell có sẵn trong hệ thống.

Shell là chương trình nằm giữa người sử dụng và kernel, thông thường nó là một bộ biên dịch dòng lệnh từ người sử dụng ở các thiết bị cuối (cũng có thể từ file) và thực hiện chúng. Không những thế, trong Uả IX shell còn là một ngôn ngữ lập trình thực sự với đầy đủ các cú pháp cần thiết như câu lệnh điều kiện, vòng lặp, các chương trình con, thủ tục...

Shell cung cấp cho người dùng một tập lệnh để người dùng thao tác với hệ thống. Khi người dùng thực hiện lệnh shell, shell sẽ dịch chúng thành các lời gọi hệ thống và chuyển cho kernel hệ điều hành xử lý. Shell cũng là một phần trong các ứng dụng mà kernel quản lý. Kernel chịu trách nhiệm cấp phát tài nguyên duy trì các tiến trình shell. Linux là hệ thống đa người dùng, khi mỗi người dùng đăng nhập hệ thống, họ sẽ nhận được một bản copy của shell để thao tác với hệ thống.

Unix shell bao gồm bộ biên dịch lệnh và ngôn ngữ lập trình. Có ba loại shell :

- **Bourne shell** của Steven Bourne đơn giản và hiệu quả. ả ó là mặc định trong đa số các hệ Uả IX (hoặc có thể gọi bởi sh).
- **C shell** của Bill Joy ở trường đại học Berkeley giống như Bourne shell nhưng bổ sung thêm các đặc điểm như bí danh, history vvv. ả ó có thể gọi bởi csh.
- **Korn shell** của David F. Korn kết hợp Bourne shell và C shell nhưng bổ sung thêm các đặc điểm riêng. ả ó có thể gọi bởi ksh.

1.2 Một số đặc điểm của Shell

Xử lý tương tác (Interative processing): ả gười dùng tương tác với shell dưới dạng đối thoại trực quan.

Chạy nền: Các chương trình trên shell có thời gian thực thi lâu và chiếm ít tài nguyên có thể cho phép chạy nền bên dưới trong khi đó người dùng có thể thực hiện các công việc khác. Điều này tăng hiệu quả sử dụng hệ thống.

Chuyển hướng (Redirection): Có thể linh hoạt chuyển đổi các dữ liệu ra vào chuẩn và lỗi.

Ống dẫn (pipe): Cho phép thực hiện nhiều lệnh liên tiếp trong đó dữ liệu ra của lệnh này được sử dụng như dữ liệu vào của lệnh kia.

Tập tin lệnh (shell script): Tạo các tập tin chứa các lệnh làm việc theo trình tự. Cấp quyền và thực thi tập tin này.

Biến shell: shell hỗ trợ sử dụng các biến lưu trữ các thông tin để điều khiển hoạt động.
Sử dụng lại các lệnh đã thực hiện (history command): Đây là tính năng rất có ích cho người dùng. Để thực hiện lại các lệnh mình đã thực hiện trước đó, thay vì phải gõ lại, người dùng có thể lại.

Cấu trúc lệnh như ngôn ngữ lập trình: Shell cho phép sử dụng lệnh như ngôn ngữ lập trình, bởi nó có thể kết hợp xử lý các tác vụ phức tạp.

Tự động hoàn tất tên file, hoặc lệnh: Chúng ta có thể gõ phần đầu của lệnh hoặc tập tin sau đó dùng <Tab> để hoàn tất phần còn lại.

Bí danh cho lệnh (command alias): Ta có thể dùng một tên mới cho một lệnh. Sau đó sử dụng tên này thay thế lệnh : *$alias dir='ls –l'* lúc này ta sử dụng lệnh dir dùng như ls –l

Các Shell trong Linux

Tên Shell	*Lịch sử ra đời*
sh (Bourne)	Shell nguyên thủy trong Unix
Csh, tcsh và zsh	Shell sử dụng cấu trúc lệnh lệnh của ngôn ngữ C làm ngôn ngữ script. Shell này được tạo bởi Bill Joy, đây là Shell thông dụng thứ 2 sau bash
Bash	Bash(Bourne Again Shell) là Shell sử dụng chính trong Linux, ra đời từ dự án Gẩ U. Bash có ưu điểm là mã nguồn mở, có thể download từ địa chỉ http://www.gnu.org
Rc	Là Shell mở rộng của C Shell với nhiều tương thích với ngôn ngữ C, ra đời từ dự án Gẩ U

Shell Linux mặc định là bash, nằm tại /bin/bash

Tất cả hệ điều hành Linux đều có Shell Bash. Muốn biết mình đang dùng Shell nào sử dụng lệnh sau: Echo $Shell

Dấu nhắc shell (dấu nhắc đợi lệnh)

#	khi ta là root (superuser), ở bất kỳ shell nào.
%	dấu nhắc khi chạy C shell.
$	dấu nhắc khi chạy Bourne shell hoặc Korn shell.
>	dấu nhắc khi chạy tcsh shell.

Trước dấu nhắc shell ta có thể đặt một chuỗi ký tự thể hiện tên riêng, tên máy tính, tên thư mục hoặc đĩa chỉ mạng.

Các siêu ký tự (wildcards)

Là những ký tự có ý nghĩa đặc biệt đối với shell : ? , *, [ ], -, !

Dấu "?" : thay thế cho 1 ký tự bất kỳ.

```
$ls fi?e
file  fine          fire
```

Dấu "*" : thay thế cho 0 hoặc nhiều ký tự bất kỳ.

```
$ls abc*xyz
abcxyz     abcdefxyz    abcdefghigjk0123456789xyz
```

Thay đổi shell làm việc

Thay đổi vĩnh viễn: dùng lệnh passwd

```
$passwd –s
Changing login shell for mang on Linux
Old shell: /bin/sh
New shell: /bin/bash
```

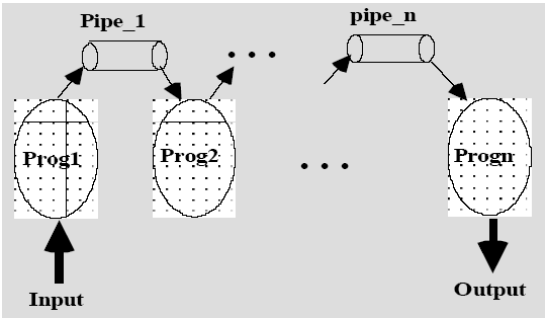
Thay đổi tạm thời (chuyển tạm thời qua shell khác) :

```
$bash
[nam@localhost nam]$ exit ( hoặc ^D, tức Ctrl-D )
```

1.3 Lập trình đường ống

Pipe còn gọi là đường ống, là cách truyền dữ liệu sử dụng kết hợp 2 chuyển tiếp. Pipe sử dụng kết xuất của một chương trình và làm nhập liệu cho một chương trình khác. Đặc điểm đường ống của Unix nối kết một lệnh với lệnh khác.

- Ống nối cho phép đầu ra của một lệnh là đầu vào của lệnh khác.
- Ống nối đơn thuần là một bộ đệm của kernel.
- Các tiến trình có thể chia sẻ dữ liệu thay cho việc sử dụng file tạm.



Đặc biệt hơn nó tạo xuất chuẩn của 1 lệnh thành nhập chuẩn của 1 lệnh khác. Ký hiệu | để thiết lập đường ống

Ví dụ: # wc baocao* | sort –n đầu ra của *wc* (trong trường hợp này là tổng số từ và ký tự của các tập tin có tên bắt đầu là baocao) và gửi nó đến lệnh sort để sắp thứ tự số. Kết quả cuối cùng là các tổng số từ sắp theo thứ tự tăng dần hiển thị trên màn hình.

Đường ống có thể kết hợp với đổi hướng: **wc baocao* | sort -n > rep-count**

kết quả sẽ đưa ra tập tin rep-count.

Với lệnh: **$ls –l | more** kết quả của lệnh ls không xuất ra màn hình mà chuyển cho lệnh more xử lý như dữ liệu đầu vào

Lệnh tee

Hoạt động chuyển tiếp và đường ống là đặc điểm của hệ điều hành Linux. Tuy nhiên ta cũng có thể sử dụng 1 lệnh của Uả IX để làm việc này. Đó là lệnh *tee*, nó sẽ giảm bớt các kết quả gián tiếp của chuỗi đường ống: **sort baocao | tee baocaostt | lp**

Đầu tiên lệnh *tee* gửi nhập chuẩn của nó đến xuất chuẩn của nó, trong trường hợp này gửi xuất của *sort* đến nhập của *lp*. Thứ hai *tee* lấy chỗ 1 bản sao của nhập chuẩn vào tên tập tin *baocaostt*.

1.4 Lập trình Shell Script

ả gôn ngữ Shell là dạng ngôn ngữ Script, không có độ uyển chuyển hay phức tạp như các ngôn ngữ lập trình chuyên nghiệp C, java,...Chương trình Shell được soạn thảo dưới dạng văn bản (text) và không được biên dịch thành file binary như các ngôn ngữ khác. Khi chạy chương trình Shell, Shell sẽ biên dịch và thực thi. Trong Linux chúng ta gặp rất nhiều các chương trình Shell xử lý những công việc rất hữu hiệu. Là nhà quản trị cần phải nắm vững cú pháp ngôn ngữ Shell để không chỉ viết những đoạn chương trình mà ít ra cũng hiểu được các script có sẵn điều khiển hệ thống của mình.

Các thành phần chính của Shell:

- Biến: kiểu chuỗi, tham số và biến môi trường.
- Điều kiện: kiểm tra luận lý.
- Các lệnh điều khiển: if, for, while, until, case.
- Hàm.
- Các lệnh nội trú của Shell.
- Các phép toán số học
- …

Chú thích trong Shell

Dòng chú thích sử dụng trong các mã nguồn chương trình dùng để giải thích ý nghĩa các lệnh hoặc chứa năng của một biến hay một đoạn chương trình. ẩ những dòng này không được biên dịch đối với các ngôn ngữ lập trình, và nó không được thực thi đối với chương trình shell.

Bắt đầu một dòng chú thích là dấu # .

```
# Dòng chú thích ghi ở đây
```

Ví dụ: Một đoạn chương trình sử dụng dòng chú thích.

```
# Kiểm tra có tồn tại tham số đầu tiên
if  test $1 –z ; then
      echo "Khong co tham so"
fi  # kết thúc if
```

Trường hợp đặc biệt chỉ thị #! không dùng để giải thích mà là đây chính là dòng lệnh gọi shell để thông dịch các lệnh trong tập tin này. Ta thường thấy dòng đầu tiên trong các chương trình shell là #! /bin/bash. Điều này có nghĩa là ta sẽ dùng shell bash để thông dịch lệnh. Shell chúng ta chạy có thể xem là shell phụ và chúng có thể thực thi các lệnh mà không làm biến đổi các biến môi trường của shell chính.

Cú pháp chung của chỉ thị này là: **#!shell-thực-thi**

ẩ ếu chúng ta không khai báo thì shell mặc nhiên trong Linux là bash. Các hệ Unix khác thì shell mặc nhiên là sh. Chỉ thị *#!* Còn dùng để chạy các chương trình khác trước khi thực thi các lệnh tiếp theo.

Sử dụng biến

Biến dùng trong chương trình shell không cần phải khai báo trước như các ngôn ngữ C, Pascal ... ẩ ó sẽ tự động khai báo khi người dùng lần đầu sử dụng. Dữ liệu biến lưu trữ được hiểu dưới dạng chuỗi dù nó có thể chứa số.

Trong trường hợp muốn sử dụng giá trị biến như là số thì phải có các phép biến đổi mà chúng ta học phía sau. Một vấn đề mà ta phải lưu ý là shell *phân biết chữ hoa và chữ thường.* Ví dụ hai biến *tong* và *Tong* là khác nhau.

Trong shell có thể kể tới 3 loại biến:

Biến môi trường: (biến shell đặc biệt, biến từ khóa, biến shell xác định trước hoặc biến shell chuẩn) được liệt kê như sau (các biến này thường gồm các chữ cái hoa):

HOME : đường dẫn thư mục riêng của người dùng,

MAIL: đường dẫn thư mục chứa hộp thư người dùng,

PATH: thư mục dùng để tìm các file thể hiện nội dung lệnh,

PS1: dấu mời ban đầu của shell (ngầm định là $),

PS2: dấu mời thứ 2 của shell (ngầm định là >),

PWD: Thư mục hiện tại người dùng đang làm,

SHELL: Đường dẫn của shell (/bin/sh hoặc /bin/ksh)

TERM: Số hiệu gán cho trạm cuối,

USER: Tên người dùng đã vào hệ thống,

Trong *.profile* ở thư mục riêng của mỗi người dùng thường có các câu lệnh dạng: **<biến môi trường> = <giá trị>**

Biến người dùng

Các biến này do người dùng đặt tên và có các cánh thức nhận giá trị các biến người dùng từ bàn phím (lệnh read). Biến được đặt tên gồm một xâu ký tự, quy tắc đặt tên như sau: ký tự đầu tiên phải là một chữ cỏi hoặc dấu gạch chân (_), sau tên là một hay nhiều ký tự khác. Để tạo ra một biến ta chỉ cần gán biến đó một giá trị nào đó. Phép gán là một dấu bằng (=). Ví dụ: myname=”duonglk”

Chú ý: không được có dấu cách (space) đằng trước hay đằng sau dấu bằng. Tên biến là phân biệt chữ hoa chữ thường. Để truy xuất đến một biến ta dùng cú pháp sau; $tên_biến. Chẳng hạn ta muốn in ra giá trị của biến myname ở trên ta chỉ cần ra lệnh: echo $myname.

```
$myname.
$myname.
```

Ta có thể khai báo một biến nhưng nó có giá trị ẩ ULL như trong những cách sau:

```
$ vech=
$ vech=""
```

ẩ ếu ta ra lệnh in giá trị của biến này thì ta sẽ thu được một giá trị ẩ ULL ra màn hình (một dòng trống).

Biến tự động (hay biến-chỉ đọc, tham số vị trí)

Là các biến do shell đã có sẵn; tên các biến này cho trước.

Có 10 biến tự động: $0, $1, $2, ..., $9.

Ký hiệu biến	*Ý nghĩa*
$1, $2, $3	Giá trị các biến tham số thứ nhất, thứ 2.. tương ứng với các tham số từ trái sang phải trong dòng tham số.
$0	Tên tập tin lệnh gọi (tên của shell script)
$*	Danh sách tham số đầy đủ
$#	Tổng số tham số.
$$	Số tiến trình (ID) mà chương trình đang hoạt động

Tham biến "$0" chứa tên của lệnh, các tham biến thực bắt đầu bằng "$1" (nếu tham số cú vị trí lớn hơn 9, ta phải sử dụng cú pháp ${} – ví dụ, ${10} để thu được các giá trị của chúng).

Shell bash có ba tham biến vị trí đặc biệt, "$#", "$@", và "$#". "$#" là số lượng tham biến vị trí (không tính "$0"). "$*" là một danh sách tất cả các tham biến vị trí loại trừ "$0", đã được định dạng như là một xâu đơn với mỗi tham biến được phân cách bởi kớ tự $IFS. "$@" trả về tất cả các tham biến vị trí được đưa ra dưới dạng ả xâu được bao trong dấu ngoặc kép.

Sự khác nhau giữa "$*" và "$@" là gì và tại sao lại có sự phân biệt? Sự khác nhau cho phép ta xử lý các đối số dòng lệnh bằng hai cách. Cách thứ nhất, "$*", do nó là một xâu đơn, nên có thể được biểu diễn linh hoạt hơn không cần yêu cầu nhiều mã shell. "$@" cho phép ta xử lý mỗi đối số riêng biệt bởi vì giá trị của chúng là ả đối số độc lập.

Một ví dụ khác về biến vị trí giúp ta phân biệt được sự khác nhau giữa biến $* và $@:

```
#!/bin/bash
#testparm.sh
function cntparm
{
     echo –e "inside cntparm $# parms: $*"
}
cntparm '$*'
cntparm '$@'
echo –e "outside cntparm $* parms\n"
echo –e "outside cntparm $# parms\n"
```

Khi chạy chương trình này ta sẽ thu được kết quả:

```
$./testparm.sh Kurt Roland Wall
inside cntparm 1 parms: Kurt Roland Wall
inside cntparm 3 parms: Kurt Roland Wall
outside cntparm: Kurt Roland Wall
outside cntparm: Kurt Roland Wall
```

Trong dòng thứ nhất và thứ 2 ta thấy kết quả có sự khác nhau, ở dòng thứ nhất biến "$*" trả về tham biến vị trí dưới dạng một xâu đơn, vì thế cntparm báo cáo một tham biến đơn. Dòng thứ hai gọi cntparm, trả về đối số dòng lệnh của là 3 xâu độc lập, vì thế cntparm báo cáo ba tham biến.

Nhập giá trị cho biến từ bàn phím

Cú pháp lệnh : read tên-biến gặp lệnh này chương trình sẽ đợi người dùng nhập giá trị vào, khi dữ liệu đã xong thì ấn Enter. Giá trị sẽ được gán vào biến tên-biến.

Ví dụ :

```
echo "Nhap vao ten cua ban"
read  ten
echo "Ten vua nhap la $ten"
```

Trong ví dụ trên khi xuất hiện dòng thông báo “ả hap vao ten cua ban “, người dùng nhập vào tên ví dụ như “ ả guyen Hung Dung” thì kết quả hiển thị là “Ten vua nhap la ả guyen Hung Dung “.

Dấu {} phải được sử dụng với tên biến theo sau bởi chữ hay số mà không phải là một phần của tên biến.

```
$ filename=chapt
$ echo ${filename}0
chapt0
```

Các ký tự đặc biệt trong bash

Ký tự	*Mô tả*
<	Định hướng đầu vào
>	Định hướng đầu ra
(	Bắt đầu subshell
)	Kết thúc subshell
|	Ký hiệu dẫn
\	Dùng để hiện ký tự đặc biệt
&	Thi hành lệnh chạy ở chế độ ngầm
{	Bắt đầu khối lệnh
}	Kết thúc khối lệnh
~	Thư mục home của người dùng hiện tại
`	Thay thế lệnh
;	Chia cắt lệnh
#	Lời chú giải
‘	Trích dẫn mạnh
“	Trích dẫn yếu
$	Biểu thức biến
*	Ký tự đại diện cho chuỗi
?	Ký tự đại diện cho một ký tự

Dấu chia cắt lệnh “ ; ” cho phép thực hiện những lệnh bash phức tạp đánh trên một dòng. ả hưng quan trọng hơn, nó là kết thúc lệnh theo lý thuyết POSIX.

Lệnh kiểm tra giá trị đúng sai của biểu thức

Lệnh test hoặc [ ] dùng để kiểm tra giá trị đúng sai của biểu thức.

Các toán tử string

Các toán tử string, cũng được gọi là các toán tử thay thế trong tài liệu về bash, kiểm tra giá trị của biến là chưa gán giá trị hoặc không xác định. Bảng dưới là danh sách các toán tử này cùng với miêu tả cụ thể cho chức năng của từng toán tử.

Toán tử	*Chức năng*
${var:- word}	ả ếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì trả về word.
${var:= word}	ả ếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì gán biến thành word, sau đó trả về giá trị của nó.
${var:+ word}	ả ếu biến tồn tại và xác định thì trả về word, còn không thì trả về null.
${var:?message}	ả ếu biến tồn tại và xác định thì trả về giá trị của nó, còn không thì hiển thị "bash: $var:$message" và thoát ra khỏi lệnh hay tập lệnh hiện thời.
${var: offset[:length]}	Trả về một xâu con của var bắt đầu tại offset của độ dài length. ả ếu length bị bỏ qua, toàn bộ xâu từ offset sẽ được trả về.

Để minh hoạ, hãy xem xét một biến shell có tên là *status* được khởi tạo với giá trị defined. Sử dụng 4 toán tử string đầu tiên cho kết quả status như sau:

```
$echo ${status:-undefined}
defined
$echo ${status:=undefined}
defined
$echo ${status:+undefined}
undefined
$echo ${status:?Dohhh\! undefined}
defined
```

Bây giờ sử dụng lệnh *unset* để xoá biến *status*, và thực hiện vẫn các lệnh đó, được *output* như sau:

```
$unset status
$echo ${status:-undefined}
undefined
$echo ${status:=undefined}
```

```
undefined
$echo ${status:+undefined}
undefined
$unset status
$echo ${status:?Dohhh\! undefined}
bash:status Dohhh! Undefined
```

Cần thiết *unset status* lần thứ hai vì ở lệnh thứ ba, *echo ${status:+undefined}*, khởi tạo lại *status* thành *undefined*.

Các toán tử *substring* đã có trong danh sách ở bảng trên đặc biệt có ích. Hãy xét biến foo có giá trị *Bilbo_the_Hobbit*. Biểu thức *${foo:7}* trả về *he_Hobbit*, trong khi *${foo:7:5}* lại trả về *he_Ho*.

Các toán tử Pattern-Matching

Các toán tử *pattern-matching* có ích nhất trong công việc với các bản ghi độ dài biến hay các xâu đã được định dạng tự do được định giới bởi các kí tự cố định. Biến môi trường *$PATH* là một ví dụ. Mặc dù nó có thể khá dài, các thư mục riêng biệt được phân định bởi dấu hai chấm. Bảng dưới là danh sách các toán tử *pattern-Matching* của bash và chức năng của chúng.

Toán tử	*Chức năng*
${var#pattern}	Xoá bỏ phần khớp (match) ngắn nhất của pattern trước var và trả về phần còn lại
${var##pattern}	Xoá bỏ phần khớp (match) dài nhất của pattern trước var và trả về phần còn lại
${var%pattern}	Xoá bỏ phần khớp ngắn nhất của pattern ở cuối var và trả về phần còn lại
${var%%pattern}	Xoá bỏ phần khớp dài nhất của pattern ở cuối var và trả về phần còn lại
${var/pattern/string}	Thay phần khớp dài nhất của pattern trong var bằng string. Chỉ thay phần khớp đầu tiên. Toán tử này chỉ có trong bash 2.0 hay lớn hơn.
${var//pattern/string}	Thay phần khớp dài nhất của pattern trong var bằng string. Thay tất cả các phần khớp. Toán tử này có trong bash 2.0 hoặc lớn hơn.

Thông thường quy tắc chuẩn của các toán tử *bash pattern-matching* là thao tác với file và tên đường dẫn. Ví dụ, giả sử ta có một tên biến shell là *mylife* có giá trị là

/usr/src/linux/Documentation/ide.txt (tài liệu về trình điều khiển đĩa IDE của nhân). Sử dụng mẫu *"/*"* và *"*/"* ta có thể tách được tên thư mục và tên file.

```
#!/bin/bash
########################################
myfile=/usr/src/linux/Documentation/ide.txt
echo '${myfile##*/}=' ${myfile##*/}
echo 'basename $myfile =' $(basename $myfile)
echo '${myfile%/*}=' ${myfile%/*}
echo 'dirname $myfile =' $(dirname $myfile)
```

Lệnh thứ 2 xoá xâu matching *"*/"* dài nhất trong tên file và trả về tên file. Lệnh thứ 4 làm khớp tất cả mọi thứ sau *"/"*, bắt đầu từ cuối biến, bỏ tên file và trả về đường dẫn của file.

Kết quả của tập lệnh này là:

```
$ ./pattern.sh
${myfile##*/}= ide.txt
basename $myfile = ide.txt
${myfile%/*}= /usr/src/linux/Documentation
dirname $myfile=/usr/src/linux/Documentation
```

Để minh hoạ về các toán tử pattern-matching và thay thế, lệnh thay thế mỗi dấu hai chấm trong biến môi trường $PATH bằng một dòng mới, kết quả hiển thị đường dẫn rất dễ đọc (ví dụ này sẽ sai nếu ta không có bash phiên bản 2.0 hoặc mới hơn):

```
$ echo –e ${PATH//:/\\n}
/usr/local/bin
/bin
/usr/bin
/usr/X11R6/bin
/home/kwall/bin
/home/wall/wp/wpbin
```

Các toán tử so sánh chuỗi:

Kiểm tra	*Điều kiện thực*
str1 = str2	str1 bằng str2
str1 != str2	str1 khác str2
-n str	str có độ dài lớn hơn 0 (khác null)
-z str	str có độ dài bằng 0 (null)

So sánh số hoc

Phép so sánh	*Kết quả*
bieuthuc1 –eq biethuc2	Đúng nếu bieuthuc1 bằng bieuthuc2
bieuthuc1 –ne biethuc2	Đúng nếu bieuthuc1 không bằng bieuthuc2
bieuthuc1 –gt biethuc2	Đúng nếu bieuthuc1 lớn hơn bieuthuc2
bieuthuc1 –ge biethuc2	Đúng nếu bieuthuc1 lớn hơn hoặc bằng bieuthuc2
bieuthuc1 –lt biethuc2	Đúng nếu bieuthuc1 nhỏ hơn bieuthuc2
bieuthuc1 –le biethuc2	Đúng nếu bieuthuc1 nhỏ hơn hoặc bằng bieuthuc2

Các toán tử kiểm tra file

Phép kiểm tra	*Kết quả*
-d file	file tồn tại và là một thư mục.
-e file	Đúng nếu file tồn tại.
-f file	Đúng nếu file là tập tin bình thường (không là một thư mục hay một file đặc biệt).
-g file	Đúng nếu file có xác lập set-group-id trên file
-s file	Đúng nếu file có kích thước khác rỗng (>0)
-u file	Đúng nếu file có xác lập set-user-id
-r file	Đúng nếu file cho phép đọc
-w file	Đúng nếu file có phép ghi
-x file	Đúng nếu file cho phép thực thi
-O file	Đúng nếu file hiện thời thuộc sở hữu của người dùng hiện thời.
-z file	Đúng nêu file có kích thước là 0.
-G file	file thuộc một trong các nhóm người dùng hiện tại là thành viên.
file1 - nt file2	file1 mới hơn file2
file1 - ot file2	file1 cũ hơn file2

Các toán tử logic

Phép kiểm tra	*Kết quả*
! expr	Đúng nếu expr không đúng

expr1 -a expr2 Đúng nếu expr1 và expr2 đúng

expr1 -o expr2 Đúng nếu expr1 đúng hoặc expr2 đúng.

Biểu thức tính toán expr hoặc let cho việc tính toán

Biểu thức *expr* sử dụng cho việc tính toán, các giá trị trong biểu thức được hiểu là số nguyên thay vì là chuỗi. ả ó cũng dùng để đổi chuỗi thành số.

Biểu thức *expr* được bao bọc bởi 2 dấu ` (Không phải dấu nháy đơn, là dấu ở phím bên trái phím số 1-!, hay là phím nằm dưới phím ESC). Trong biểu thức tính toán các toán tử và toán hạng cách nhau bằng khoảng trắng.

Các phép toán và phép so sánh expr cho phép

|	hoặc	=	bằng nhau
&	và	+	cộng
>	lớn hơn	-	trừ
<	nhỏ hơn	*	nhân
>=	lớn hơn hoặc bằng	/	chia
<=	nhỏ hơn hoặc bằng	%	chia lấy phần dư
!=	khác nhau		

Ví dụ:

```
$ let "a = 1 + 1"
$ echo $a
2
$ a=`expr $a + 1`
$ echo $a
3
```

Kết nối lệnh, khối lệnh và lấy giá trị của lệnh

Shell cho phép sử dụng phép hoặc (OR) và phép và (Aả D) để kết nối các lệnh.

Phép và (AND)

```
lệnh _1  &&  lệnh_2 && lệnh_3 …
```

Các lệnh thực hiện từ trái sang phải cho đến khi một lệnh có kết quả lỗi. Kết quả cuối cùng của dãy lệnh này là đúng (true) nếu tất cả các lệnh đều đúng, ngược lại là sai.

Phép hoặc (OR)

```
lệnh _1  || lệnh_2 || lệnh_3 …
```

Các lệnh thực hiện từ trái sang phải cho đến khi một lệnh có kết quả đúng. Kết quả cuối cùng của dãy lệnh này là đúng (true) nếu có ít nhất một lệnh là đúng, ngược lại là sai.

```
test -d demo && echo "demo is a directory"
test -d demo | | echo "demo is not a directory"
(test -d demo && ls -l demo) | | echo "demo not ok"
```

Ta có thể kết hợp lại cả 2 loại toán tử lại để có một biểu thức như sau:

```
command1 && comamnd2 || command3
```

ả ếu câu lệnh *command1* chạy thành công thì shell sẽ chạy lệnh *command2* và nếu *command1* không chạy thành công thì command3 được chạy.

Ví dụ **$ rm myf && echo "File is removed successfully" || echo "File is not removed"**

ả ếu file *myf* được xóa thành công (giá trị trả về của lệnh là 0) thì lệnh "*echo File is removed successfully*" sẽ được thực hiện, nếu không thì lệnh "*echo File is not removed*" được chạy.

Khối lệnh

Khi chúng ta cần thực thi nhiều lệnh liên tiếp nhau, có thể dùng khối lệnh. Khối lệnh nằm giữa 2 dấu { }.

Lấy giá trị của một lệnh. Khi viết chương trình nhiều khi lấy kết quả của lệnh này làm đối số hay giá trị xử lý của lệnh kia. Ta có thể làm được điều này bằng cách sử dụng cú pháp *$(command)*. Khi dùng *$(command)*, kết quả của việc thực hiện lệnh *command* được trả về.

1.5 Điều khiển luồng

Các cấu trúc điều khiển luồng của bash, nó bao gồm:

- if – Thi hành một hoặc nhiều câu lệnh nếu có điều kiện là true hoặc false.
- for – Thi hành một hoặc nhiều câu lệnh trong một số cố định lần.
- while – Thi hành một hoặc nhiều câu lệnh trong khi một điều kiện nào đó là true hoặc false.
- until – Thi hành một hoặc nhiều câu lệnh cho đến khi một điều kiện nào đó trở thành true hoặc false.
- case – Thi hành một hoặc nhiều câu lệnh phụ thuộc vào giá trị của biến.
- select – Thi hành một hoặc nhiều câu lệnh dựa trên một khoảng tuỳ chọn của người dùng.

Formatted: Bullets and Numbering

Biểu thức điều kiện if

Cú pháp: **if then else fi**

```
if condition
then
```

```
     statements
[elif  condition
     statements]
[else
     statements]
fi
```

Bash cung cấp sự thực hiện có điều kiện lệnh nào đó sử dụng câu lệnh if, câu lệnh if của bash đầy đủ chức năng như của C. Cú pháp của nó được khái quát như sau:

Đầu tiên, ta cần phải chắc chắn rằng mình hiểu *if* kiểm tra trạng thái thoát của câu lệnh trong *condition*. ẩ ếu nó là 0 (true), sau đó statements sẽ được thi hành, nhưng nếu nó khác 0, thì mệnh đề *else* sẽ được thi hành và điều khiển nhảy tới dòng đầu tiên của mã *fi*. Các mệnh đề *elif* (tuỳ chọn) (có thể nhiều tuỳ ý) sẽ chỉ thi hành khi điều kiện *if* là *false*.

Tương tự, mệnh đề *else* (tuỳ chọn) sẽ chỉ thi hành khi tất cả else không thỏa mãn. ẩ hìn chung, các chương trình Linux trả về 0 nếu thành công hay hoàn toàn bình thường, và khác 0 nếu ngược lại, vì thế không có hạn chế nào cả.

Chú ý: Không phải tất cả chương trình đều tuân theo cùng một chuẩn cho giá trị trả về, vì thế cần kiểm tra tài liệu về các chương trình ta kiểm tra mã thoát với điều kiện *if*.

Ví dụ chương trình *diff*, trả về 0 nếu không có gì khác nhau, 1 nếu có sự khác biệt và 2 nếu có vấn đề nào đó. ẩ ếu một câu điều kiện hoạt động không như mong đợi thì hãy kiểm tra tài liệu về mã thoát .

Không quan tâm đến cách mà chương trình xác định mã thoát của chúng, bash lấy 0 có nghĩa là true hoặc bình thường còn khác 0 là false. ẩ ếu ta cần cụ thể để kiểm tra một mã thoát của lệnh, sử dụng toán tử ***$?*** ngay sau khi chạy lệnh. ***$?*** trả về mã thoát của lệnh chạy ngay lúc đó.

Phức tạp hơn, bash cho phép ta phối hợp các mã thoát trong phần điều kiện sử dụng các toán tử && và || được gọi là toán tử logic Aẩ D và OR. Cú pháp đầy đủ cho toán tử Aẩ D như sau:

```
command1 && command2
```

Câu lệnh *command2* chỉ được chạy khi và chỉ khi *command1* trả về trạng thái là số 0 (true).

Cú pháp cho toán tử OR thì như sau:

```
command1 || command2
```

Câu lệnh *command2* chỉ được chạy khi và chỉ khi *command1* trả lại một giá trị khác 0 (false).

Ta có thể kết hợp lại cả 2 loại toán tử lại để có một biểu thức như sau:

```
command1 && comamnd2 || command3
```

ẩ ếu câu lệnh *command1* chạy thành công thì shell sẽ chạy lệnh *command2* và nếu *command1* không chạy thành công thì *command3* được chạy.

*Ví dụ:***$ rm myf && echo "File is removed successfully" || echo "File is not removed"**

ẩ ếu file *myf* được xóa thành công (giá trị trả về của lệnh là 0) thì lệnh "*echo File is removed successfully*" sẽ được thực hiện, nếu không thì lệnh "*echo File is not removed*" được chạy.

Giả sử trước khi ta vào trong một khối mã, ta phải thay đổi một thư mục và copy một file. Có một cách để thực hiện điều này là sử dụng các toán tử *if* lồng nhau, như là đoạn mã sau:

```
if  cd  /home/kwall/data
then
     if   cp  datafile  datafile.bak
     then
          # more code here
     fi
fi
```

Tuy nhiên, bash cho phép ta viết đoạn mã này ngắn gọn hơn nhiều như sau:

```
if cd /home/kwall/data && cp datafile datafile.bak
then
     # more code here
fi
```

Cả hai đoạn mã đều thực hiện cùng một chức năng, nhưng đoạn thứ hai ngắn hơn nhiều, gọn nhẹ và đơn giản. Mặc dù *if* chỉ kiểm tra các mã thoát, ta có thể sử dụng cấu trúc *[...]* lệnh *test* để kiểm tra các điều kiện phức tạp hơn. *[condition]* trả về giá trị biểu thị condition là true hay false. *test* cũng có tác dụng tương tự.

Một ví dụ khác về cách sử dụng cấu trúc ***if:***

```
#!/bin/sh
# Script to test if..elif...else
#
if [ $1 -gt 0 ]; then
     echo "$1 is positive"
elif [ $1 -lt 0 ]
```

```
then
     echo "$1 is negative"
elif [ $1 -eq 0 ]
then
     echo "$1 is zero"
else
     echo "Opps! $1 is not number, give number"
fi
```

Ta có thể kiểm tra các thuộc tính file, so sánh các xâu và các biểu thức số học.

Chú ý: Các khoảng trống trước dấu mở ngoặc và sau dấu đóng ngoặc trong *[condition]* là cần phải có. Đây là điều kiện cần thiết trong cú pháp shell của bash.

Ví dụ 1:

```
if test -f file1
then echo "file exists"
else echo "file does not exist"
fi
```

Ví dụ 2: ả hập vào điểm của môn học, cho biết kết quả.

```
echo chuong trinh ket qua mon hoc
echo Nhap vao diem
read diem
if  [ $diem  -ge  5 ] ; then
     echo "Dat"
else
     echo "Hong"
fi
```

Ví dụ 3:

```
if test -f file1; then
     echo "file exists"
elif test -d file1; then
     echo "file is a directory"
fi
```

trong trường hợp này fi dùng chung.

Ví dụ 4: ả hập vào điểm cho biết xếp loại.

```
echo  Xep loai
echo  Nhap vao diem
read  diem
if  test  $diem  -ge  8  ; then
     echo " Loai Gioi"
```

```
elif test  $diem –ge  7 ; then
     echo "Loai Kha"
 elif test  $diem –ge 5 ; then
     echo  "Loai TB"
 else
     echo  "Loai Yeu"
 fi
```

Ví dụ chương trình shell cho các toán tử *test* file trên các thư mục trong biến $PATH. Mã cho chương trình descpath.sh như sau:

```
#!/bin/bash
################################
IFS=:
for dir in $PATH;
do
      echo $dir
      if [ -w $dir ]; then
            echo -e "\tYou have write permission in $dir"
      else
            echo –e “\tYou don’t have write permission in $dir”
      fi
      if [ -0 $dir ]; then
            echo -e "\tYou own $dir"
      else
            echo –e “\tYou don’t own $dir”
      fi
      if [ -G $dir ]; then
            echo -e "\tYou are a member of $dir's group"
      else
            echo -e "\tYou aren't a member of $dir's group"
      fi
done # Ket thuc vong lap for
```

Biểu thức lệnh rẽ nhánh case

Cấu trúc điều khiển luồng tiếp theo là *case*, hoạt động cũng tương tự như lệnh switch của C. ả ó cho phép ta thực hiện các khối lệnh phụ thuộc vào giá trị của biến.

Cú pháp đầy đủ của *case* như sau:

```
case expr in
      pattern1 )       statements ;;
      pattern2 )       statements ;;
```

```
        …
        [*)             statements ;;] # giá trị mặc định
esac
```

expr được đem đi so sánh với từng pattern, nếu nó bằng nhau thì các lệnh tương ứng sẽ được thi hành. Dấu ;; là tương đương với lệnh break của C, tạo ra điều khiển nhảy tới dòng đầu tiên của mã *esac*. Không như từ khoá *switch* của C, lệnh case của bash cho phép ta kiểm tra giá trị của *expr* dựa vào pattern, nó có thể chứa các kí tự đại diện.

Cách làm việc của cấu trúc *case* như sau: nó sẽ khớp (match) biểu thức expr với các mẫu pattern1, pattern2,…nếu có một mẫu nào đó khớp thì khối lệnh tương ứng với mẫu đó sẽ được thực thi, sau đó nó thoát ra khỏi lệnh case. ả ếu tất cả các mẫu đều không khớp và ta có sử dụng mẫu * (trong nhánh *)), ta thấy đây là mẫu có thể khớp với bất kỳ giá trị nào (ký tự đại diện là *), nên các lệnh trong nhánh này sẽ được thực hiện.

Cấu trúc điều khiển *select* (không có trong các phiên bản bash nhỏ hơn 1.14) chỉ riêng có trong *Korn (K – Shell)* và các *shell bash (B – Shell)*. Thêm vào đó, nó không có sự tương tự như trong các ngôn ngữ lập trình quy ước. *select* cho phép ta dễ dàng trong việc xây dựng các menu đơn giản và đáp ứng các chọn lựa của người dùng.

Cú pháp của nó như sau:

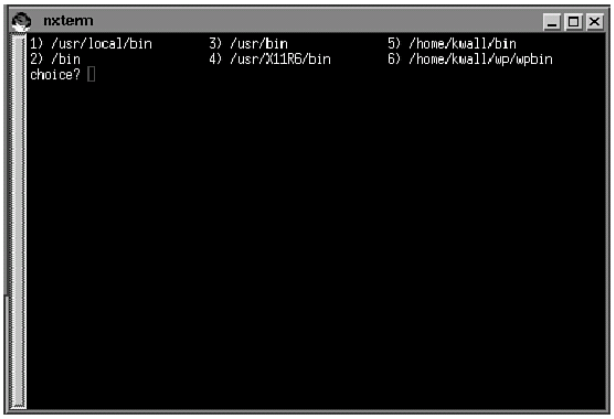
```
select  value [in list]
do
      statements that manipulate $value
done
```

Dưới đây là một ví dụ về cách sử dụng lệnh *select*: Chương trình tạo các menu bằng select.

```
#!/bin/bash
# menu.sh – Createing simple menus with select
#####################################
IFS=:
PS3="choice? "
# clear the screen
clear
select dir in $PATH
do
      if [ $dir ]; then
            cnt=$(ls –Al $dir | wc -l)
            echo "$cnt files in $dir"
      else
```

```
            echo "Dohhh! No such choice!"
        fi # kết thúc if
        echo -e "\nPress ENTER to continue, CTRL -C to quit"
        read
        clear
done
```

Lệnh đầu tiên đặt kí tự *IFS* là : (ký tự phân cách), vì thế select có thể phân tích hoàn chỉnh biến môi trường *$PATH*. Sau đó nó thay đổi lời nhắc *default* khi *select* bằng biến PS3. Sau khi xoá sạch màn hình, nó bước vào một vòng lặp, đưa ra một danh sách các thư mục nằm trong *$PATH* và nhắc người dùng chọn lựa như là minh hoạ trong hình dưới.



ả ếu người dùng chọn hợp lệ, lệnh *ls* được thực hiện kết quả được gửi cho lệnh đếm từ *wc* để đếm số file trong thư mục và hiển thị kết quả có bao nhiêu file trong thư mục đó. Do *ls* có thể sử dụng mà không cần đối số, script đầu tiên cần chắc chắn là $dir khác null (nếu nó là null, ls sẽ hoạt động trên thư mục hiện hành nếu người dùng chọn 1 menu không hợp lệ). ả ếu người dùng chọn không hợp lệ, một thông báo lỗi sẽ được hiển thị.

Câu lệnh *read* (được giới thiệu sau) cho phép người dùng đánh vào lựa chọn của mình và nhấn Enter để lặp lại vòng lặp hay nhấn Ctrl + C để thoát.

Chú ý: ả hư đã giới thiệu, các vòng lặp script không kết thúc nếu ta không nhấn Ctrl+C. Tuy nhiên ta có thể sử dụng lệnh *break* để thoát ra.

Dùng *case* khi chúng ta sử dụng giá trị của một biểu thức để rẽ các nhánh khác nhau.

Ví dụ: Trong ví dụ này sẽ tạo menu lựa chọn và cho phép người dùng chọn chức năng thực hiện. ả ếu biến chọn là 1 thì liệt kê thư mục hiện hành, 2 thì cho biết đường dẫn thư mục hiện hành, các số khác là không hợp lệ.

```
clear
echo
```

```
echo " Menu "
echo " 1. Liệt kê thư mục hiện hành"
echo " 2. Cho biết đường dẫn thư mục hiện hành"
read chon
case $chon in
     1    ) ls -l  ;;
     2    ) pwd ;;
     *    ) echo "Không hợp lệ";;
esac
```

Vòng lặp For

ả hư đã thấy ở chương trình trên, *for* cho phép ta chạy một đoạn mã một số lần nhất định. Tuy nhiên cấu trúc *for* của bash chỉ cho phép ta lặp đi lặp lại trong danh sách các giá trị nhất định bởi vì nó không tự động tăng hay giảm con đếm vòng lặp như là C, Pascal, hay Basic.

Tuy nhiên, vòng lặp *for* là công cụ lặp thường xuyên được sử dụng bởi vì nó điều khiển gọn gàng trên các danh sách, như là các tham số dòng lệnh và các danh sách các file trong thư mục.

Cú pháp đầy đủ của *for* là:

```
for value in list
do
     statements using $value
done
```

list là một danh sách các giá trị, ví dụ như là tên file. Giá trị là một thành viên danh sách đơn và *statements* là các lệnh sử dụng value. Một cú pháp khác của lệnh *for* có dạng như sau:

```
for (( expr1; expr2; expr3 ))
do
       .....
        ...
      repeat all statements between do and
      done until expr2 is TRUE
done
```

Linux không có tiện ích để đổi tên hay copy các nhóm của file. Trong MS-DOS nếu ta có 17 file có phần mở rộng a*.doc, ta có thể sử dụng lệnh COPY để copy *.doc thành file *.txt.

Lệnh DOS như sau: **C:\ cp doc*.doc doc*.txt**

sử dụng vòng lặp for của bash để bù đắp những thiếu sót này. Đoạn mã dưới đây có thể được chuyển thành chương trình shell thực hiện đúng như những gì ta muốn:

```
for docfile in doc/*.doc
do
```

```
        cp $docfile ${docfile%.doc}.txt
done
```

Sử dụng một trong các toán tử pattern-matching của bash, đoạn mã này làm việc copy các file có phần mở rộng là *.doc bằng cách thay thế .doc ở cuối của tên file bằng .txt.

Một ví dụ khác về vòng for đơn giản như sau:

```
#!/bin/bash
for i in 1 2 3 4 5
do
      echo "Welcome $i times"
done
```

Ta cũng có một cấu trúc về *for* như sau, chương trình này cũng có cùng chức năng như chương trình trên nhưng ta chú ý đến sự khác biệt về cú pháp của lệnh *for*.

```
#!/bin/bash
for ((´i = 0 ;´i <= 5;´i++´))
do
´     echo "Welcome $i times"
done
^D
$ sh for2
Welcome 0 times
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
```

Tiếp theo là một ví dụ về vòng *for* lồng nhau:

```
#!/bin/bash
for (( i = 1; i <= 5; i++ ))     ### Outer for loop ###
do
      for (( j = 1 ; j <= 5; j++ )) ### Inner for loop ###
      do
     echo -n "$i "
done
```

Ví dụ khác về cách sử dụng cấu trúc if và for như sau:

```
#!/bin/sh
#Script to test for loop#
#
if [ $# -eq 0 ]
```

```
then
      echo "Error - Number missing form command line argument"

      echo "Syntax : $0 number"
      echo "Use to print multiplication table for given number"
      exit                                                          1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
      echo "$n * $i = `expr $i \* $n`"
done
```

Khi ta chạy chương trình với tham số:

```
$ sh mtable  7
```

Ta thu được kết quả như sau:

```
7 * 1 = 7
7 * 2 = 14
...
..
7 * 10 = 70
```

Cho phép thực hiện một chuỗi lệnh như nhau với mỗi một giá trị trong danh sách đã cho. Số các vòng lặp bằng số các giá trị trong danh sách.

***Ví dụ*:** Shell_script copy sao chép các file trong danh sách đối vào danh mục /users/user8 và đổi nhóm thành nhóm student, đổi người sở hữu thành user8.

```
$cat copy
for i
do
      if [-f $i]; then
            cp $i /users/user8
            chgrp student /users/user8/$i
            chown user8 /users/user8/$i
      fi
done
```

Vòng lặp While và until

Vòng lặp *for* giới hạn số lần mà một đoạn mã được thi hành, các cấu trúc *while* và *until* của *bash* cho phép một đoạn mã được thi hành liên tục cho đến khi một điều kiện nào đó xảy ra.

Chỉ với chú ý là đoạn mã này cần viết sao cho điều kiện cuối phải xảy ra nếu không sẽ tạo ra một vòng lặp vô tận.

Cú pháp của nó như sau:

```
while condition
do
      statements
done
```

Cú pháp này có nghĩa là khi nào condition còn true, thì thực hiện statements cho đến khi *condition* trở thành *false* (cho đến khi một chương trình hay một lệnh trả về khác 0).

Hai lệnh thường dùng trong vòng lặp *while*:

```
true hoặc :      cho giá trị true(0)
sleep[n]         đợi n giây
```

shell_script disp_time hiển thị số liệu ngày tháng theo khoảng thời gian 30 giây.

```
$cat disp_time
while true hoặc sử dụng while :
do
      date
      sleep 30
done
```

Cú pháp *until* có nghĩa là trái ngược với *while*: cho đến khi *condition* trở thành *true* thì thi hành *statements* (có nghĩa là cho đến khi một lệnh hay chương trình trả về mã thoát khác 0).

```
until condition
do
      statements
done
```

Cấu trúc while của bash khắc phục thiếu sót không thể tự động tăng, giảm con đếm cua vòng lặp for. Ví dụ, ta muốn copy 150 bản của một file, thì vòng lặp *while* là một lựa chọn để giải quyết bài toán này. Dưới đây là chương trình:

```
#!/bin/sh
declare  -i  idx
idx=1
while [ $idx != 150]
do
      cp somefile  somefile.$idx
      idx=$idx+1
done
```

Chương trình này giới thiệu cách sử dụng tính toán số nguyên của bash. Câu lệnh *declare* khởi tạo một biến, *idx*, định nghĩa là một số nguyên. Mỗi lần lặp *idx* tăng lên, nó sẽ được kiểm tra để thoát khỏi vòng lặp. Vòng lặp *until* tuy cũng có khả năng giống *while* nhưng không được dùng nhiều vì rất khó viết và chạy chậm.

Một ví dụ nữa về cách sử dụng vòng lặp while được minh họa trong chương trình in bản nhân của một số:

```
#!/bin/sh
#Script to test while statement
if [ $# -eq 0 ]
then
      echo "Error - Number missing form command line argument"
      echo "Syntax : $0 number"
      echo " Use to print multiplication table for given number"
      exit 1
fi
n=$1
i=1
while [ $i -le 10 ]
do
      echo "$n * $i = `expr $i \* $n`"
      i=`expr $i + 1`
done
```

Ví dụ: Chương trình sẽ lặp cho đến khi n<=10

```
echo Nhap vao so n
read n
until [ $n –lt 10 ]
do
      echo " n lớn hơn 10"
      n=‘expr $n –1’
done
```

Lệnh Break, continue, exit

Lệnh break

Cho phép ta thoát ra khỏi vòng lặp mà không cần kiểm tra điều kiện lặp.

Lệnh exit

Làm chương trình thoát ra và trở về dấu nhắc lệnh $.

Ví dụ 1: ả hận số n từ đối số dòng lệnh, tính tổng S =1+2+ ..+n

```
echo "chuong trinh tinh tong"
if [-z $1 ]; then
      echo "tong  <n>"
      exit 0
fi
s=0
i=1
while true
do
      s=`expr $i + $s`
      i=`expr $i + 1`
      if  [i –gt n ]; then
           break;
      fi
done
echo $s
```

Ví dụ 2: **S**hell_script stock ghi các dòng ký tự vào từ bàn phím lên file lines cho tới khi ta gõ từ "Eả D"

```
$cat stock
while true
do
      echo "Enter your line:"
      read answer
      if test "$answer" = "END" ;then
           break
      else
           echo $answer >> lines
      fi
done
```

Chú ý: break[n] cho phép ra khỏi n mức của các vòng lặp lồng.

Lệnh continue: cho phép bỏ qua các lệnh còn lại, quay về đầu vòng lặp.

Ví dụ 3: shell_script supprim xoá tất cả các file có trong danh sách đối, trừ file save và source:

```
$cat supprim
set -x
for i
do
      if test "$i" = "save" -o "$i" = "source"
      then continue
      fi
```

```
        echo $i
        rm $i
done
```

Các lệnh khác

Lệnh Shift

Chuyển giá trị hiện thời được lưu trong dãy các tham số sang trái một vị trí, nếu thêm tham số đi cùng lệnh *shift* thì sẽ dịch chuyển sang trái ngần ấy vị trí .

Ví dụ:

```
$1 = -r $2 = file1 $3 = file2
$shift
```

Kết quả là:

```
$1 = file1 $2 = file2
```

ẩ ếu muốn dịch sang trái hai vị trí thì sử dụng lệnh.

```
shift 2
```

1.6 Hàm

Cũng như các ngôn ngữ lập trình khác, Shell cho phép sử dụng hàm. Hàm là một đoạn chương trình con nằm trong *script* chính. ẩ ó có thể được gọi lại nhiều lần trong script chính.

Hàm chức năng của bash là một cách mở rộng các tiện ích sẵn có trong shell, nó có các điểm lợi sau:

- Thi hành nhanh hơn do các hàm shell luôn thường trực trong bộ nhớ.
- Cho phép việc lập trình trở nên dễ dàng hơn vì ta có thể tổ chức chương trình thành các module.

Định nghĩa hàm:

```
tên-hàm() {
      các-lệnh-của-hàm.
}
```

ẩ ếu so sánh với C hay Pascal, hàm của bash không được chặt chẽ, nó không kiểm tra lỗi và không có phương thức trả về đối số bằng giá trị. Tuy nhiên giống như C và Pascal, các biến địa phương có thể khai báo cục bộ đối với hàm, do đó tránh được sự xung đột với biến toàn cục.

Để thực hiện điều này ta dùng từ khoá local như trong đoạn mã sau:

```
function   foo
{
      local  myvar
      local  yourvar=1
}
```

Trong ví dụ về các biến vị trí ở trên ta cũng thấy được cách sử dụng hàm trong bash. Các hàm shell giúp mã của ta dễ hiểu và dễ bảo dưỡng. Sử dụng các hàm và các chú thích ta sẽ đỡ rất nhiều công sức khi ta phải trở lại nâng cấp đoạn mã mà ta đã viết từ thời gian rất lâu trước đó.

Ví dụ:

```
chao()
{
   echo "hello"
}
```

Gọi hàm và truyền tham số cho hàm. Để gọi hàm thực hiện ta sử dụng tên hàm hoặc có thêm tham số đi kèm (chú ý tên hàm phải khác với tên của các lệnh unix đã tồn tại). Shell thực hiện các lệnh trong { } khi hàm được gọi.

```
tên-hàm
tên-hàm thamso-1 thamso-2 …
```

Sử dụng tham số trong hàm cũng như trong *script* chính. Hàm có thể truy xuất tập hợp biến của shell hiện hành.

```
lietke()
{
       /bin/ls -C
}
```

1.7 Mảng

Khái niệm mảng trong ngôn ngữ lập trình Shell Script cũng giống như trong các ngôn ngữ lập trình khác Pascal, C, C++, Java, php, asp…

Mảng được khởi tạo theo cấu trúc sau: **declare –a array_name**. Trong lập trình Shell ta cần chú ý là không có khái niệm cố định kích thước của mảng, để truy xuất các thành phần của mạng ta có thể sử dụng những cách truy xuất thường dùng: x[0], x[1], x[2]

```
declare -a nums=(45 33 100 65)
declare -ar names (array is readonly)
names=( Tom Dick Harry)
states=( ME [3]=CA CT )
x[0]=55
n[4]=100
```

tùy chọn –r chỉ ra rằng đây là mảng chỉ đọc.

Để lấy giá trị của một thành phần của mảng ta sử dụng cú pháp sau:

${arrayname[index]}

Ví dụ:

```
$declare -a lop
$lop=(abc defg ijklm pqwst)
$echo ${#lop}         # 3
$echo ${#lop[0]}      # 3
$echo ${#lop[1]}      # 4
$echo ${#lop[*]}      # 4
$echo ${#lop[@]}      # 4
$echo ${lop[*]} # abc defg ijklm pqwst
$echo ${lop[@]} # abc defg ijklm pqwst
```

Copy một mảng:

```
array2=( "${array1[@]}" )
array2="${array1[@]}"
```

Thêm một thành phần cho mảng:

```
array=( "${array[@]}" "new element" )
array[${#array[*]}]="new element"
```

Một số ví dụ về sử dụng mảng trong lập trình shell

The Bubble Sort [1]

```
#!/bin/bash
#bubble.sh: Bubble sort, of sorts.
#Recall the algorithm for a bubble sort. In this particular version...
# With each successive pass through the array to be sorted,
#+ compare two adjacent elements, and swap them if out of order.
# At the end of the first pass, the "heaviest" element has sunk to
bottom.
# At the end of the second pass, the next "heaviest" one has sunk next
to bottom.
# And so forth.
# This means that each successive pass needs to traverse less of the
array.
# You will therefore notice a speeding up in the printing of the later
passes.
exchange()
{
      # Swaps two members of the array.
      local temp=${Countries[$1]} # Temporary storage
      #+ for element getting swapped out.
      Countries[$1]=${Countries[$2]}
      Countries[$2]=$temp
      return
```

```
}
declare -a Countries # Declare array,
#+ optional here since it's initialized below.
# Is it permissable to split an array variable over multiple lines
#+ using an escape (\)?
# Yes.
Countries=(Netherlands Ukraine Zaire Turkey Russia Yemen Syria Brazil
Argentina Nicaragua Japan Mexico Venezuela Greece England Israel Peru
Canada Oman Denmark Wales France Kenya Xanadu Qatar Liechtenstein
Hungary)
# "Xanadu" is the mythical place where, according to Coleridge,
#+ Kubla Khan did a pleasure dome decree.
clear # Clear the screen to start with.
echo "0: ${Countries[*]}" # List entire array at pass 0.
number_of_elements=${#Countries[@]}
let "comparisons = $number_of_elements - 1"
count=1 # Pass number.
while [ "$comparisons" -gt 0 ] # Beginning of outer loop
do
      index=0 # Reset index to start of array after each pass.
      while [ "$index" -lt "$comparisons" ] # Beginning of inner loop
      do
      if [ ${Countries[$index]} \> ${Countries[`expr $index + 1`]} ]
      # If out of order...
      # Recalling that \> is ASCII comparison operator
      #+ within single brackets.
      # if [[ ${Countries[$index]} > ${Countries[`expr $index + 1`]} ]]
      #+ also works.
      then
            exchange $index `expr $index + 1` # Swap.
      fi # end if
      let "index += 1"
      done # End of inner loop
      let "comparisons -= 1"
      #Since "heaviest" element bubbles to bottom,
      #+ we need do one less comparison each pass.
      echo
      echo "$count: ${Countries[@]}"
      # Print resultant array at end of each pass.
      echo
```

```
      let "count += 1" # Increment pass count.
      done # End of outer loop
      # All done.
exit 0
^D
```

Push- down Stack [1]

```
#!/bin/bash
#stack.sh: push-down stack simulation
#Similar to the CPU stack, a push-down stack stores data items
#sequentially, but releases them in reverse order,LIFO
BP=100          # Base Pointer of stack array.
                # Begin at element 100.
SP=$BP          # Stack Pointer.
                # Initialize it to "base"(bottom) of stack
Data=                 # Contents of stack location.
                # Must use local variable,
                #because of limitation on function return range
declare -a stack
push() {              # Push item on stack.
      if [ -z "$1" ]    # Nothing to push?
      then
            return
      fi
      let "SP -= 1"         # Bump stack pointer.
      stack[$SP]=$1
      return
}
pop() {               # Pop item off stack.
      Data=           # Empty out data item.
      if [ "$SP" -eq "$BP" ] # Stack empty?
      then
            return
      fi    # This also keeps SP from getting past 100,
      #+ i.e., prevents a runaway stack.
      Data=${stack[$SP]}
      let "SP += 1"    # Bump stack pointer.
      return
}
status_report(){ # Find out what's happening
      echo "-------------------------"
```

```
      echo "REPORT"
      echo "Stack Pointer = $SP"
      echo "Just popped \""$Data"\"off the stack"
      echo "----------------------------"
      echo
}
# the main(). Now, for exect this function
echo   # See if you can pop anything off empty stack
pop
status_report
echo
push garbage
pop
status_report         # Garbage in, garbage out
value1=23;       push $value1
value2=skidoo;   push $value2
value3=FINAL;    push $value3
pop                         # FINAL
status_report
pop                         # skidoo
status_report
pop                         # 23
status_report               # Last-in, first-out!
# Notice how the stack pointer decrements with each push,
#+ and increments with each pop.
echo
# ==========================================================
# Exercises:
# ---------
# 1) Modify the "push()" function to permit pushing
# + multiple element on the stack with a single function call.
# 2) Modify the "pop()" function to permit popping
# + multiple element from the stack with a single function call.
# 3) Using this script as a jumping-off point,
# + write a stack-based 4-function calculator.
exit 0
^D
```

Mô phỏng mảng hai chiều trong Shell Script

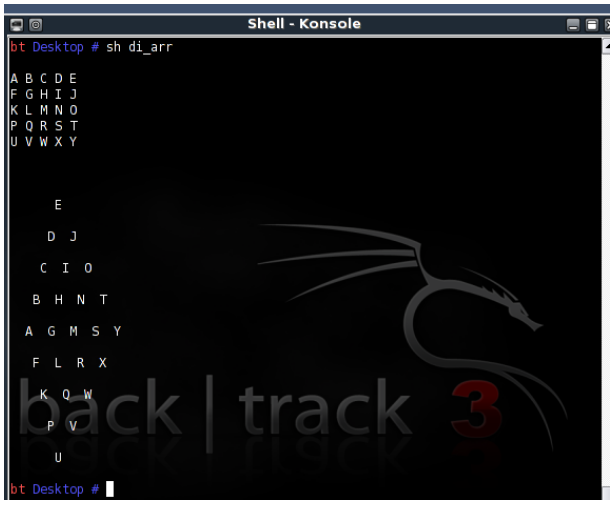
Mã nguồn chương trình mô phỏng mảng 2 chiều (di_arr)

```
#!/bin/bash
# Simulating a two-dimensional array.
# A two-dimensional array stores rows sequentially.
Rows=5
Columns=5
declare -a alpha # char alpha [Rows] [Columns];
# Unnecessary declaration.
load_alpha()
{
      local rc=0
      local index
      for i in A B C D E F G H I J K L M N O P Q R S T U V W X Y
      do
            local row=`expr $rc / $Columns`
            local column=`expr $rc % $Rows`
            let "index = $row * $Rows + $column"
            alpha[$index]=$i # alpha[$row][$column]
            let "rc += 1"
      done
      # Simpler would be
      #declare -a alpha=( A B C D E F G H I J K L M N O P Q R S T U V W X Y)
      # but this somehow lacks the "flavor" of a two-dimensional array.
}
print_alpha()
{
      local row=0
      local index
      echo
      while [ "$row" -lt "$Rows" ] # Print out in "row major" order -
      do    # columns vary
            # while row (outer loop) remains the same.
            local column=0
            while [ "$column" -lt "$Columns" ]
            do
                  let "index = $row * $Rows + $column"
                  echo -n "${alpha[index]} " # alpha[$row][$column]
                  let "column += 1"
            done
            let "row += 1"
            echo
```

```
      done
      # The simpler equivalent is
      # echo ${alpha[*]} | xargs -n $Columns
      echo
}
filter () # Filter out negative array indices.
{
      echo -n " " # Provides the tilt.
      if [[ "$1" -ge 0 && "$1" -lt "$Rows" && "$2" -ge 0 && "$2" -lt
"$Columns" ]]
then
let "index = $1 * $Rows + $2"
# Now, print it rotated.
echo -n " ${alpha[index]}" # alpha[$row][$column]
fi
}
rotate () # Rotate the array 45 degrees
{ # ("balance" it on its lower lefthand corner).
local row
local column
for (( row = Rows; row > -Rows; row-- )) # Step through the array
backwards.
do
for (( column = 0; column < Columns; column++ ))
do
if [ "$row" -ge 0 ]
then
let "t1 = $column - $row"
let "t2 = $column"
else
let "t1 = $column"
let "t2 = $column + $row"
fi
filter $t1 $t2 # Filter out negative array indices.
done
echo; echo
done
# Array rotation inspired by examples (pp. 143-146) in
# "Advanced C Programming on the IBM PC", by Herbert Mayer
# (see bibliography).
```

```
}
#--------------------------------------------------------#
load_alpha # Load the array.
print_alpha # Print it out.
rotate # Rotate it 45 degrees counterclockwise.
#--------------------------------------------------------#
# This is a rather contrived, not to mention kludgy simulation.
```

Chạy chương trình



1.8 Một số các lệnh thường dùng trong lập trình Shell

Các toán tử định hướng vào ra

Ta đã được biết về các toán tử định hướng vào ra, > và <. Toán tử định hướng ra cho phép ta gửi kết quả ra của một lệnh vào một file.

Ví dụ như lệnh sau: ***$ cat $HOME/.bash_profile > out***

ả ó sẽ tạo một file tên là *out* trong thư mục hiện tại chứa các nội dung của file *bash_profile,* bằng cách định hướng đầu ra của *cat* tới file đó.

Tương tự, ta có thể cung cấp đầu vào là một lênh từ một file hoặc là lệnh sử dụng toán tử đầu vào, <.

Tacó thể viết lại lệnh cat để sử dụng toán tử định hướng đầu vào như sau:

```
$ cat < $HOME/.bash_profile > out
```

Kết quả của lệnh này vẫn như thế nhưng nó cho ta hiểu thêm về cách sử dụng định hướng đầu vào đầu ra.

Toán tử định hướng đầu ra, >, sẽ ghi đè lên bất cứ file nào đang tồn tại. Đôi khi điều này là không mong muốn, vì thế bash cung cấp toán tử nối thêm dữ liệu, >>, cho phép nối thêm dữ liệu vào cuôi file. Hay xem lệnh thêm bí danh cdlpu vào cuối của file .bashrc của tôi:

```
$echo "alias cdlpu='cd $HOME/kwall/projects/lpu' " >> $HOME/.bashrc
```

Một cách sử dụng định hướng đầu vào là đầu vào chuẩn (bàn phím). Cú pháp của lệnh này như sau:

```
Command << label
Input …
Label
```

Cú pháp này nói lên rằng *command* đọc các input cho đến khi nó gặp label. Dưới đây là ví dụ về cách sử dụng cấu trúc này:

```
#!/bin/bash
###################################
 USER=anonymous
PASS=kwall@xmission.com
ftp    -i    -n   <<   END
open   ftp.caldera.com
user $USER $PASS
cd   /pub
ls
close
END
```

Hiện dòng văn bản

Lệnh *echo* hiện ra dòng văn bản được ghi ngay trong dòng lệnh có cú pháp:

echo [tùy chọn] [xâu ký tự]…

với các tùy chọn như sau:

- _____n : hiện xâu ký tự và dấu nhắc trên cùng một dòng.
- _____e : bật khả năng thông dịch được các ký tự điều khiển.
- _____E : tắt khả năng thông dịch được các ký tự điều khiển.
- _____- help : hiện hỗ trợ và thoát. Một số bản Linux không hỗ trợ tham số này.

Formatted: Bullets and Numbering

Ví dụ, dùng lệnh *echo* với tham số *-e*

```
# echo -e 'thử dùng lệnh echo \n'
```

sẽ thấy hiện ra chính dòng văn bản ở lệnh:

```
thử dùng lệnh echo
```

Ở đây ký tự điều khiển '\n' là ký tự xuống dòng.

Lệnh set

Để gán kết quả đưa ra từ lệnh shell ra các biến tự động, ta dùng lệnh set.

Dạng lệnh set: **set `<lệnh>`**

Sau lệnh này, kết quả thực hiện lệnh không hiệ ra lê màn hình mà gán kết quả đó tương ứng cho các biến tự động. Một cách tự động các từ trong kết quả thực hiện lệnh sẽ gán tương ứng cho các biến tự động (từ $1 trở đi).

Xem xét một ví dụ sau đây (chương trình thu2.arg) có nội dung:

```
#!/bin/sh
# Hien thoi diem chay chuong trinh nay
set `date`
echo "Thoi gian: $4 $5"
echo "Thu: $1"
echo "Ngay $3 thang $2 nam $6"
```

Sau khi đổi mode của File chương trình này và chạy, chúng ta nhận được:

```
Thoi gian: 7:20:15 EST
Thu: Tue
Ngay 20 thang Oct nam 1998
Như vậy,
$# = 6
$* = Tue Oct 20 7:20:15 EST 1998
$1 = Tue   $2=Oct     $3 = 20    $4 = 7:20:15
$5 = EST   $6 = 1998
```

1.8 Đệ quy

Tất cả các shell_script đều có tính đệ quy (recursivity).

Ví dụ: shell_script dir_tree hiển thị cây thư mục bắt đầu từ thư mục là đối của nó.

```
$cat dir_tree
if test -d $1
then echo $1 is a directory
for j in $1/*
do $0 $j              #$0 tên shell_script chính là dir_tree
done
fi
$dir_tree /usr
/usr is a directory
/usr/adm is a directory
/usr/adm/acct is a directory
/usr/adm/acct/fiscal is a directory
/usr/adm/acct/nite is a directory
```

```
/usr/adm/sa is a directory
/usr/bin is a directory
```

1.9 Lập trình hội thoại

Cú pháp chung của các hội thoại hay được sử dụng:

```
dialog --title {title} --backtitle {backtitle} {Box options}
    where Box options can be any one of following
    --yesno      {text}  {height} {width}
    --msgbox     {text}  {height} {width}
    --infobox    {text}  {height} {width}
    --inputbox   {text}  {height} {width} [{init}]
    --textbox    {file}  {height} {width}
    --menu  {text}  {height} {width} {menu} {height} {tag1} {item1}...
```

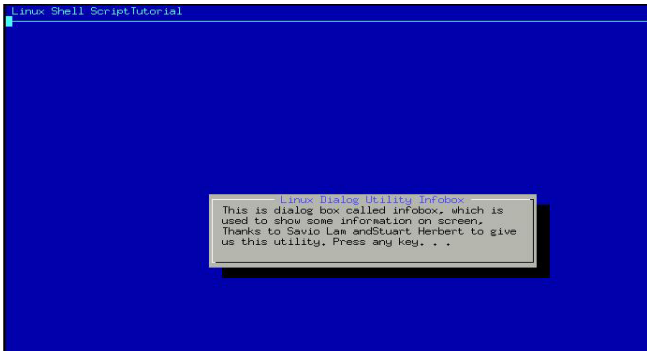
Infobox

Mã nguồn chương trình (dia1)

```
$ cat > dia1
dialog --title "Linux Dialog Utility Infobox" --backtitle "Linux Shell Script
Tutorial" --infobox "This is dialog box called infobox, which is used \
to show some information on screen, Thanks to Savio Lam and\
Stuart Herbert to give us this utility. Press any key. . . " 7 50 ; read
```

Chạy chương trình

```
$ sh dia1
```



Ở đây số 7 và 50 là chiều cao và chiều rộng của hộp thoại.

Message box (msgbox)

Mã nguồn chương trình (dia2)

```
$cat > dia2
dialog --title "Linux Dialog Utility Msgbox" --backtitle "Linux Shell\
Script Tutorial" --msgbox "This is dialog box called msgbox, which is\
```

```
used to show some information on screen which has also Ok button,\
Thanks to Savio Lam and Stuart Herbert to give us this utility. Press
any key. . . " 9 50
```

Chạy chương trình

```
$ sh dia2
```

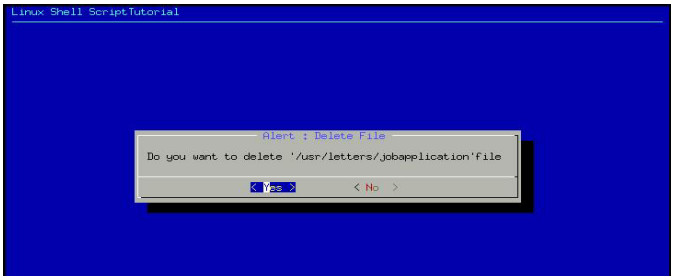
Yesno box

Mã nguồn chương trình (dia3)

```
$ cat > dia3
dialog --title "Alert : Delete File" --backtitle "Linux Shell Script \
Tutorial" --yesno "\nDo you want to delete\
'/usr/letters/jobapplication'file" 7 60
sel=$?
case $sel in
     0) echo "User select to delete file";;
     1) echo "User select not to delete file";;
     255) echo "Canceled by user by pressing [ESC] key";;
esac
```

Chạy chương trình

```
$ sh dia3
```



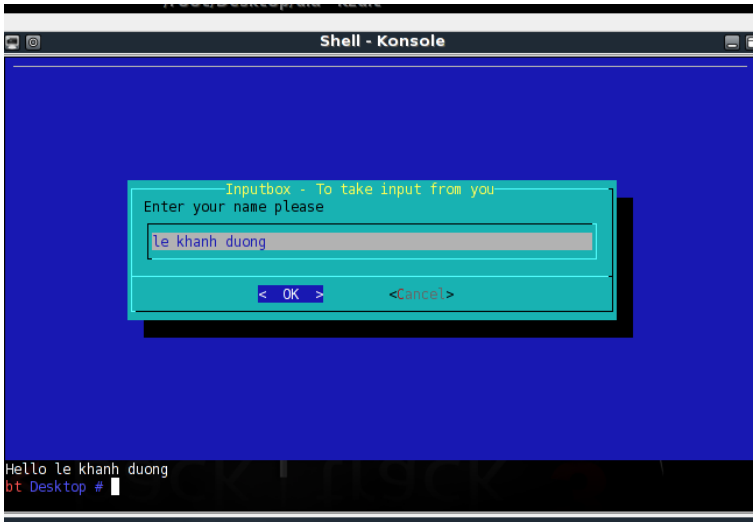
Input Box (inputbox)

Mã nguồn chương trình (dia4)

```
dialog --title "Inputbox - To take input from you" --backtitle "Linux
Shell Script Tutorial" --inputbox "Enter your name please" 8 60
2>/tmp/input.$$
sel=$?
na=`cat /tmp/input.$$`
case $sel in
  0) echo "Hello $na" ;;
  1) echo "Cancel is Press" ;;
  255) echo "[ESCAPE] key pressed" ;;
esac
rm -f /tmp/input.$$
```

Chạy chương trình

```
$ sh dia4
```



1.10 Một số ví dụ về Shell

Chương trình tính tổng 2 số

Mã nguồn chương trình (tong1.sh)

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
# Q1.Script to sum to nos
if [ $# -ne 2 ]
then
    echo "Usage - $0   x    y"
```

```
    echo "          Where x and y are two nos for which I will
print sum"
    exit 1
fi
    echo "Sum of $1 and $2 is `expr $1 + $2`"
```

Chương trình in ra kế quả như 5,4,3,2,1

Mã nguồn chương trình

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
# Q3
# Algo:
# 1) START: set value of i to 5 (since we want to start from 5,
#    if you want to start from other value put that value)
# 2) Start While Loop
# 3) Chechk, Is value of i is zero, If yes goto step 5 else
#    continue with next step
# 4) print i, decement i by 1 (i.e. i=i-1 to goto zero) and
#     goto step 3
# 5) END
i=5
while test $i != 0
do
      echo "$i"
      i=`expr $i - 1`
done
```

Chương trình tính tổng: 1-> n

Mã nguồn chương trình (tong.sh)

```
#!/bin/sh
echo “Chuong trinh tinh tong 1- $1”
index=0
tong=0
while [ $index -lt $1 ]
do
      index=$(($index + 1))
      tong=$(($tong + $index))
done
echo "Tong 1-$1= $tong"
```

```
exit 0
```

Chạy chương trình

```
# sh tong 100
```

Chương trình tính giai thừa

Mã nguồn chương trình (giaithua.sh)

```
#!/bin/sh
echo "Chuong trinh tinh $1!"
index=0
gt=1
while [ $index -lt $1 ]
do
      index=$(($index + 1))
      gt=$(($gt * $index))
done
echo "$1!= $gt"
exit 0
```

Chạy chương trình

```
# sh giaithua 5
```

Chương trình đếm số dòng của một file

Mã nguồn chương trình (demdong.sh)

```
#!/bin/sh
echo "Chuong trinh dem so dong cua tap tin $1"
{
n=0
while read line
do
      n=$(($n + 1))
done
echo "So dong cua tap tin $1 la : $n"
} < $1
exit 0
```

Chạy chương trình

```
# sh demdong vidu.txt
```

Chương trình đếm số từ trong một file

Mã nguồn chương trình (demtu.sh)

```
#!/bin/sh
echo "Chuong trinh dem so tu cua tap tin $1"
{
```

```
n=0
while read line
do
      for wd in $line
      do
            n=$(($n + 1))
      done
done
echo "Tong so tu cua tap tin $1 la : $n"
}<$1
exit 0
```

Chạy chương trình

```
# sh demtu vidu.txt
```

Chương trình tìm dòng dài nhất trong tập tin

Mã nguồn chương trình (dongmax.sh)

```
#!/bin/sh
echo "Chuong trinh tim dong dai nhat trong tap tin $1"
{
n=0
max=0
dong=""
while read line
do
      n=`expr length "$line"`
      if [ $n -gt $max ]
      then
            dong="$line"
            max=$n
      fi
done
echo "Dong trong tap tin $1 co do dai max = $max la : $dong"
}<$1
exit 0
```

Chạy chương trình

```
# sh dongmax vidu.txt
```

Chương trình tìm một xâu trong một tập tin

Mã nguồn chương trình (timxau.sh)

```
#!/bin/sh
```

```
echo "Chuong trinh tim xau $1 trong tap tin $2"
{
wordlen=`expr length "$1"`       # Do dai tu can tim
while read textline
do
      textlen=`expr length "$textline"` # Do dai cua dong vua doc
      end=$(($textlen - wordlen + 1))
      index=1
      while [ $index -le $end ]
      do
            temp=`expr substr "$textline" $index $wordlen
            if [ "$temp" = $1 ]
            then
                  echo "Tim thay $1 tai dong $textline"
                  break
            fi
            index=$(($index + 1))
      done
done
}<$2
exit 0
```

Chạy chương trình

```
# sh timxau abc vidu.txt
```

Chương trình kiểm tra số nguyên tố

Mã nguồn chương trình (nguyento.sh)

```
$ cat >  nt.sh
n=$1
for (( i=2; i< n; i++ ))
do
      temp=`expr $n % $i`
      if   test $temp –eq 0; then
            echo " không nguyên tố"
            exit
      fi
done
echo " la nguyên tố"
```

Chạy chương trình

```
# sh nguyento 5
```

Chương trình tính ước chung lớn nhất của hai số

Mã nguồn chương trình (ucln.sh)

```
if  test   $n  -ne 2
then
      echo" truyền tham sô"
fi
      g=0
      x=$1
      y=$2
if test $x –lt 0
then
      x=`expr 0 -$x `
fi
if [ $y –lt 0 ]
then
      $y=`expr 0 -$y`
while [ $x –gt 0 ]
do
      g=$x
      x=`expr $y %$x`
      y=$g
done
echo " ucln là :$g"
```

Chạy chương trình

```
# sh ucln 4 8
```

Chương trình tìm số lớn nhất trong 3 tham số truyền vào

Mã nguồn chương trình

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
# Q2. Script to find out bigest number
# Algo:
#  1) START: Take three nos as n1,n2,n3.
#  2) Is n1 is greater than n2 and n3, if yes
#  print n1 is bigest no goto step 5, otherwise goto next step
#  3) Is n2 is greater than n1 and n3, if yes
#  print n2 is bigest no goto step 5, otherwise goto next step
#  4) Is n3 is greater than n1 and n2, if yes
#   print n3 is bigest no goto step 5, otherwise goto next step
```

```
#  5) END
    if [ $# -ne 3 ]
    then
           echo "$0: number1 number2 number3 are not given" >&2
           exit 1
    fi
    n1=$1
    n2=$2
    n3=$3
    if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
    then
           echo "$n1 is Bigest number"
    elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
    then
           echo "$n2 is Bigest number"
    elif [ $n3 -gt $n1 ] && [ $n3 -gt $n2 ]
    then
           echo "$n3 is Bigest number"
    elif [ $1 -eq $2 ] && [ $1 -eq $3 ] && [ $2 -eq $3 ]
    then
           echo "All the three numbers are equal"
    else
           echo "I can not figure out which number is biger"
    fi
```

Chương trình in ra số theo dạng ngược lại, nếu 123 thì in ra 321

Mã nguồn chương trình (daonguoc.sh)

```
# Algo:
# 1) Input number n
# 2) Set rev=0, sd=0
# 3) Find single digit in sd as n % 10 it will give (left most digit)
# 4) Construct revrse no as rev * 10 + sd
# 5) Decrment n by 1
# 6) Is n is greater than zero, if yes goto step 3, otherwise next
step
# 7) Print rev
if [ $# -ne 1 ]
then
    echo "Usage: $0 number"
    echo "       I will find reverse of given number"
```

```
    echo "       For eg. $0 123, I will print 321"
    exit 1
fi
n=$1
rev=0
sd=0
while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    rev=`expr $rev \* 10  + $sd`
    n=`expr $n / 10`
done
    echo  "Reverse number is $rev"
```

Chạy chương trình

```
# sh daonguoc.sh 1234
```

Chương trình in ra tổng của các chữ số, ví dụ 123 kết quả là 1+2+3 = 6

Mã nguồn chương trình (tongso.sh)

```
Algo:
#1) Input number n
#2) Set sum=0, sd=0
#3) Find single digit in sd as n % 10 it will give (left most digit)
#4) Construct sum no as sum=sum+sd
#5) Decrment n by 1
#6) Is n is greater than zero, if yes goto step 3, otherwise next step
#7) Print sum
#
if [ $# -ne 1 ]
then
    echo "Usage: $0   number"
    echo " I will find sum of all digit for given number"
    echo " For eg. $0 123, I will print 6 as sum of all digit (1+2+3)"
    exit 1
fi
n=$1
sum=0
sd=0
while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
```

```
    sum=`expr $sum + $sd`
    n=`expr $n / 10`
done
echo  "Sum of digit for numner is $sum"
```

Chương trình tính tổng của hai số thực a=5.66, b=8.67, c=a+b

Mã nguồn chương trình tong.sh

```
a=5.66
b=8.67
c=`echo $a + $b | bc` # đầu ra của phép tính tổng sẽ được bc tính
echo "$a + $b = $c"
```

Chạy chương trình

```
# sh tong.sh
```

Chương trình sử dụng getopts

Mã nguồn chương trình (ham.sh)

```
# -c clear
# -d dir
# -m mc
# -e vi { editor }
# Function to clear the screen
cls()
{
    clear
    echo "Clear screen, press a key . . ."
    read
    return
}
# Function to show files in current directory
show_ls()
{
    ls
    echo "list files, press a key . . ."
    read
    return
}
# Function to start mc
start_mc()
{
    if which mc > /dev/null ; then
```

```
            mc
            echo "Midnight commander, Press a key . . ."
            read
    else
            echo "Error: Midnight commander not installed, Press a key
                  . . ."
            read
    fi
    return
}
# Function to start editor
start_ed()
{
    ced=$1
    if which $ced > /dev/null ; then
            $ced
            echo "$ced, Press a key . . ."
            read
    else
            echo "Error: $ced is not installed or no such editor exist,
                  Press a key . . ."
            read
    fi
    return
}
# Function to print help
print_help_uu()
{
      echo "Usage: $0 -c -d -m -v {editor name}";
      echo "Where -c clear the screen";
      echo "      -d show dir";
      echo "      -m start midnight commander shell";
      echo "      -e {editor}, start {editor} of your choice";
      return
}
# Main procedure start here
# Check for sufficent args
if [ $# -eq 0 ] ; then
    print_help_uu
    exit 1
```

```
fi
# Now parse command line arguments
while getopts cdme: opt
do
    case "$opt" in
           c) cls;;
           d) show_ls;;
           m) start_mc;;
           e) thised="$OPTARG"; start_ed $thised ;;
           \?) print_help_uu; exit 1;;
    esac
done
```

Chạy chương trình

```
# sh ham.sh
```

Chương trình in ra date, time, username, và thư mục hiện thời

Mã nguồn chương trình

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
echo "Hello, $LOGNAME"
echo "Current date is `date`"
echo "User is `who i am`"
echo "Current direcotry `pwd`"
```

Viết Shell script in ra "Hello World" (in Bold, Blink effect, and in different colors like red, brown)

Mã nguồn chương trình

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
# echo command with escape sequance to give differnt effects
# Syntax: echo -e "escape-code your message, var1, var2 etc"
# For eg. echo -e "\033[1m  Hello World"
#                   |         |
#                   |         |
#               Escape code   Message
#
```

```
clear
echo -e "\033[1m Hello World"    # bold effect
echo -e "\033[5m Blink"          # blink effect
echo -e "\033[0m Hello World"     # back to noraml
echo -e "\033[31m Hello World"    # Red color
echo -e "\033[32m Hello World"    # Green color
echo -e "\033[33m Hello World"    # See remaing on screen
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
echo -e "\033[36m Hello World"
echo -e -n "\033[0m "              # back to noraml
echo -e "\033[41m Hello World"
echo -e "\033[42m Hello World"
echo -e "\033[43m Hello World"
echo -e "\033[44m Hello World"
echo -e "\033[45m Hello World"
echo -e "\033[46m Hello World"
echo -e "\033[0m Hello World"      # back to noraml
```

Viế ra Shell Script thể hiện đồng hồ số

Mã nguồn chương trình (ms)

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
echo
echo "Digital Clock for Linux"
echo "To stop this clock use command kill pid, see above for pid"
echo "Press a key to continue. . ."
while :
do
    ti=`date +"%r"`
    echo -e -n "\033[7s"    #save current screen postion & attributes
    #
    # Show the clock
    #
    tput cup 0 69          # row 0 and column 69 is used to show clock
    echo -n $ti            # put clock on screen
    echo -e -n "\033[8u"   #restore current screen postion & attributs
```

```
    #
    #Delay fro 1 second
    #
    sleep 1
done
^D
```

Chạy chương trình

```
#sh ms &
```

Shell script to convert upercase filename to lowercase in current

Mã nguồn chương trình (up2low)

```
#!/bin/bash
# up2low : script to convert upercase filename to lowercase in current
# working dir
# Author : Vivek G. Gite <vivek@nixcraft.com>
#Copy this file to your bin directory i.e. $HOME/bin as cp rename.awk
# $HOME/bin
AWK_SCRIPT="rename.awk"
# change your location here
awkspath=$HOME/bin/$AWK_SCRIPT
ls -1 > /tmp/file1.$$
tr "[A-Z]" "[a-z]" < /tmp/file1.$$ > /tmp/file2.$$
paste /tmp/file1.$$ /tmp/file2.$$ > /tmp/tmpdb.$$
rm -f /tmp/file1.$$
rm -f /tmp/file2.$$
# Make sure awk script exist
if [ -f $awkspath ]; then
   awk -f $awkspath /tmp/tmpdb.$$
else
   echo -e "\n$0: Fatal error - $awkspath not found"
   echo -e "\nMake sure \$awkspath is set correctly in $0 script\n"
fi
rm -f /tmp/tmpdb.$$
```

Chương trình xem các thông tin hệ thống

Mã nguồn chương trình

```
#!/bin/bash
# Linux Shell Scripting Tutorial 1.05r3, Summer-2002
# Written by Vivek G. Gite <vivek@nixcraft.com>
# Latest version can be found at http://www.nixcraft.com/
```

```
nouser=`who | wc -l`
echo -e "User name: $USER (Login name: $LOGNAME)" >> /tmp/info.tmp.01.$$$
echo -e "Current Shell: $SHELL"  >> /tmp/info.tmp.01.$$$
echo -e "Home Directory: $HOME" >> /tmp/info.tmp.01.$$$
echo -e "Your O/s Type: $OSTYPE" >> /tmp/info.tmp.01.$$$
echo -e "PATH: $PATH" >> /tmp/info.tmp.01.$$$
echo -e "Current directory: `pwd`" >> /tmp/info.tmp.01.$$$
echo -e "Currently Logged: $nouser user(s)" >> /tmp/info.tmp.01.$$$
if [ -f /etc/redhat-release ]
then
    echo -e "OS: `cat /etc/redhat-release`" >> /tmp/info.tmp.01.$$$
fi
if [ -f /etc/shells ]
then
    echo -e "Available Shells: " >> /tmp/info.tmp.01.$$$
    echo -e "`cat /etc/shells`"  >> /tmp/info.tmp.01.$$$
fi
if [ -f /etc/sysconfig/mouse ]
then
    echo -e "--------------------------------" >> /tmp/info.tmp.01.$$$
    echo -e "Computer Mouse Information: " >> /tmp/info.tmp.01.$$$
    echo -e "--------------------------------" >> /tmp/info.tmp.01.$$$
    echo -e "`cat /etc/sysconfig/mouse`" >> /tmp/info.tmp.01.$$$
fi
echo -e "--------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer CPU Information:" >> /tmp/info.tmp.01.$$$
echo -e "------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/cpuinfo >> /tmp/info.tmp.01.$$$
echo -e "------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer Memory Information:" >> /tmp/info.tmp.01.$$$
echo -e "------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/meminfo >> /tmp/info.tmp.01.$$$
if [ -d /proc/ide/hda ]
then
    echo -e "------------------------------" >> /tmp/info.tmp.01.$$$
    echo -e "Hard disk information:" >> /tmp/info.tmp.01.$$$
    echo -e "------------------------------" >> /tmp/info.tmp.01.$$$
    echo -e "Model: `cat /proc/ide/hda/model` " >> /tmp/info.tmp.01.$$$
    echo -e "Driver: `cat /proc/ide/hda/driver` " >> /tmp/info.tmp.01.$$$
    echo -e "Cache size: `cat /proc/ide/hda/cache` " >> /tmp/info.tmp.01.$$$
fi
```

```
echo -e "--------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "File System (Mount):" >> /tmp/info.tmp.01.$$$
echo -e "--------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/mounts >> /tmp/info.tmp.01.$$$
if which dialog > /dev/null
then
      dialog   --backtitle "Linux Software Diagnostics (LSD) Shell Script
      Ver.1.0"   --title   "Press   Up/Down   Keys   to   move"   --textbox
      /tmp/info.tmp.01.$$$ 21 70
else
    cat /tmp/info.tmp.01.$$$ |more
fi
rm -f /tmp/info.tmp.01.$$$
```

2. Lập trình C trên Linux

Linux cung cấp nhiều công cụ hỗ trợ phát triển các ứng dụng dựa trên ngôn ngữ C và C++, nội dung chính của chương này là mô tả các công cụ hỗ trợ biên dịch, gỡ lỗi các ứng dụng C trên nền tảng Linux.

- Trình biên dịch gcc.
- Sử dụng gdb gỡ lỗi.

2.1 Trình biên dịch gcc

Hệ điều hành Unix luôn kèm theo bộ dịch ngôn ngữ lập trình C với tên gọi là cc (C compiler). Trong Linux, bộ dịch có tên là *gcc* (Gả U C Compiler) với ngôn ngữ lập trình không khác nhiều với C chuẩn.

ả ội dung chi tiết về các ngôn ngữ lập trình c trên Linux thuộc phạm vi của các tài liệu khác. ***gcc*** cho người lập trình kiểm tra trình biên dịch. Quá trình biên dịch bao gồm bốn giai đoạn:

- Tiền xử lý.
- Biên dịch.
- Tạp hợp.
- Liên kết.

Ta có thể dừng quá trình sau một trong những giai đoạn để kiểm tra kết quả biên dịch tại giai đoạn ấy. *gcc* cũng có thể chấp nhận ngôn ngữ khác của C, như Aả SI C hay C truyền thống. ả hư đã nói ở trên, *gcc* thích hợp biên dịch C++ hay Objective-C. Ta có thể kiểm soát lượng cũng như kiểu thông tin cần debug, tất nhiên là có thể nhúng trong quá trình nhị phân hóa kết quả và giống như hầu hết các trình biên dịch, *gcc* cũng thực hiện tối ưu hóa mã.

Trước khi bắt đầu đi sâu vào nghiên cứu *gcc*, ta xem một ví dụ sau:

```
#include<stdio.h>
int main (void)
{
       fprintf( stdout, "Hello, Linux programming world!\n");
       return  0;
 }
```

Để biên dịch và chạy chương trình này hãy gõ:

```
1     $  gcc  hello.c  -o  hello
2     $  ./hello
3     Hello, Linux programming world!
```

Dòng lệnh đầu tiên chỉ cho *gcc* phải biên dịch và liên kết file nguồn *hello.c*, tạo ra tập tin thực thi, bằng cách chỉ định sử dụng đối số *-o hello*. Dòng lệnh thứ hai thực hiện chương trình, và kết quả cho ra trên dòng thứ 3.

Có nhiều chỗ mà ta không nhìn thấy được, *gcc* trước khi chạy *hello.c* thông qua bộ tiền xử lý của *cpp*, để mở rộng bất kỳ một macro nào và chèn thêm vào nội dung của những file *#include.* Tiếp đến, nó biên dịch mã nguồn tiền xử lý sang mã *obj* . Cuối cùng, trình liên kết, tạo ra mã nhị phân cho chương trình *hello*.

Ta có thể tạo lại từng bước này bằng tay, chia thành từng bước qua tiến trình biên dịch. Để chỉ cho gcc biết phải dừng việc biên dịch sau khi tiền xử lý, ta sử dụng tuỳ chọn –E của gcc:

```
$  gcc  -E  hello.c  -o  hello.cpp
```

Xem xét *hello.cpp* và ta có thể thấy nội dung của *stdio.h* được chèn vào file, cùng với những mã thông báo tiền xử lý khác. Bước tiếp theo là biên dịch *hello.cpp* sang mã *obj*. Sử dụng tuỳ chọn *–c* của *gcc* để hoàn thành:

```
$  gcc  -x  cpp-output  -c  hello.cpp  -o  hello.o
```

Trong trường hợp này, ta không cần chỉ định tên của file output bởi vì trình biên dịch tạo một tên file obj bằng cách thay thế .c bởi .o. Tuỳ chọn –x chỉ cho gcc biết bắt đầu biên dịch ở bước được chỉ báo trong trường hợp này với mã nguồn tiền xử lý.

Làm thế nào gcc biết chia loại đặc biệt của file? ầ ó dựa vào đuôi mở rộng của file ở trên để xác định rõ phải xử lý file như thế nào cho dúng. Hầu hết những đuôi mở rộng thông thường và chú thích của chúng được liệt kê trong bảng dưới.

Phần mở rộng	***Kiểu***
.c	Mã nguồn ngôn ngữ C
.c, .cpp	Mã nguồn ngôn ngữ C++
.i	Mã nguồn C tiền xử lý

.ii	Mã nguồn C++ tiền xử lý
.S, .s	Mã nguồn Hợp ngữ
.o	Mã đối tượng biên dịch (obj)
.a, .so	Mã thư viện biên dịch

Các phần mở rộng của tên file đối với *gcc.*

Liên kết file đối tượng, và cuối cùng tạo ra mã nhị phân:

```
$ gcc hello.o -o hello
```

Trong trường hợp , ta chỉ muốn tạo ra các file obj, và như vậy thì bước liên kết là không cần thiết.

Hầu hết các chương trình C chứa nhiều file nguồn thì mỗi file nguồn đó đều phải được biên dịch sang mã obj trước khi tới bước liên kết cuối cùng. Giả sử có một ví dụ, ta đang làm việc trên killerapp.c là chương trình sử dụng phần mã của helper.c, như vậy để biên dịch killerapp.c ta phải dùng dòng lệnh sau:

```
$ gcc killerapp.c helper.c -o killerapp
```

gcc qua lần lượt các bước tiền xử lý - biên dịch – liên kết, lúc này tạo ra các file obj cho mỗi file nguồn trước khi tạo ra mã nhị phân cho killerapp.

Một số tuỳ chọn dòng lệnh của gcc:

- o FILE Chỉ định tên file output; không cần thiết khi biên dịch sang mã obj. ả ếu FILE không được chỉ rõ thì tên mặc định sẽ là a.out.
- c Biên dịch không liên kết.
- DF00=BAR Định nghĩa macro tiền xử lý đặt tên F00 với một giá trị của BAR trên dòng lệnh.
- IDIRả AME Trước khi chưa quyết định được DIRả AME hãy tìm kiếm những file include trong danh sách các thư mục(tìm trong danh sách các đường dẫn thư mục)
- LDIRả AME Trước khi chưa quyết định được DIRả AME hãy tìm kiếm những file thư viện trong danh sách các thư mục. Với mặc định gcc liên kết dựa trên những thư viện dùng chung
- static Liên kết dựa trên những thư viện tĩnh
- lF00 Liên kết dựa trên libF00
- g Bao gồm chuẩn gỡ rối thông tin mã nhị phân
- ggdb Bao gồm tất cả thông tin mã nhị phân mà chỉ có chương trình gỡ rối Gả U- gdb mới có thể hiểu được

– O Tối ưu hoá mã biên dịch

– Oả Chỉ định một mức tối ưu hoá mã ả , 0<=ả <=3.

– Aả SI Hỗ trợ chuẩn Aả SI/ISO của C, loại bỏ những mở rộng của Gả U mà xung đột với chuẩn (tuỳ chọn này không bảo đảm mã theo Aả SI).

– pedantic Cho ra tất cả những cảnh báo quy định bởi chuẩn

– pedantic-erors Thông báo ra tất cả các lỗi quy định bởi chuẩn Aả SI/ISO của C.

– traditional Hỗ trợ cho cú pháp ngôn ngữ C của Kernighan và Ritchie (giống như cú pháp định nghĩa hàm kiểu cũ).

– w Chặn tất cả thông điệp cảnh báo.

– Wall Thông báo ra tất cả những cảnh báo hữu ích thông thường mà gcc có thể cung cấp.

– werror Chuyển đổi tất cả những cảnh báo sang lỗi mà sẽ làm ngưng tiến trình biên dịch.

– MM Cho ra một danh sách sự phụ thuộc tương thích được tạo.

– v Hiện ra tất cả các lệnh đã sử dụng trong mỗi bước của tiến trình biên dịch.

Chú ý: ả ếu không có tùy chọn –o thì kết quả sẽ tạo ra một file thực thi có tên là a.out.

```
$gcc thu.c
```

Kết quả sẽ tạo ra file a.out, để hiển thị kết quả sử dụng lệnh sau :

```
$./ a.out
```

ả ếu có thêm tùy chọn –o kết quả sẽ tạo ra file thực thi với tên do người dùng tạo ra.

```
$ gcc –o thu thu.c
$ sh thu
```

2.2 Công cụ GNU make

Trong trường hợp ta viết một chương trình rất lớn được cấu thành bởi từ nhiều file, việc biên dịch sẽ rất phức tạp vì phải viết các dòng lệnh gcc rất là dài. Để khắc phục tình trạng này, công cụ Gả U make đã được đưa ra.

Gả U make được giải quyết bằng cách chứa tất cả các dòng lệnh phức tạp đó trong một file gọi là makefile. ả ó cũng làm tối ưu hóa quá trình dịch bằng cách phát hiện ra những file nào có thay đổi thì nó mới dịch lại, còn file nào không bị thay đổi thì nó sẽ không làm gì cả, vì vậy thời gian dịch sẽ được rút ngắn.

Một makefile là một cơ sở dữ liệu văn bản chứa cách luật, các luật này sẽ báo cho chương trình make biết phải làm gì và làm như thế nào. Một luật bao gồm các thành phần như sau:

- Đích (target) – cái mà make phải làm.
- Một danh sách các thành phần phụ thuộc (dependencies) cần để tạo ra đích.

Formatted: Bullets and Numbering

- Một danh sách các câu lệnh để thực thi trên các thành phần phụ thuộc.

Khi được gọi, Gầ U make sẽ tìm các file có tên là Gầ Umakefile, makefile hay Makefile. Các luật sẽ có cú pháp như sau:

```
target: dependency1, dependency2, ….
      command
      command
      ……
```

Target thường là một file như file khả thi hay file object ta muốn tạo ra. *Dependency* là một danh sách các file cần thiết như là đầu vào để tạo ra target. *Command* là các bước cần thiết (chẳng hạn như gọi chương trình dịch) để tạo ra *target*.

Dưới đây là một ví dụ về một makefile về tạo ra một chương trình khả thi có tên là editor (số hiệu dòng chỉ đưa vào để tiện theo dõi, còn nội dung của makefile không chứa số hiệu dòng). Chương trình này được tạo ra bởi một số các file nguồn: editor.c, editor.h, keyboard.h, screen.h, screen.c, keyboard.c.

```
1.  editor : editor.o screen.o keyboard.o
2.  gcc -o  editor.o screen.o keyboard.o
3.  editor.o : editor.c editor.h keyboard.h screen.h
4.  gcc -c  editor.c
5.  screen.o : screen.c screen.h
6.  gcc -c  screen.c
7.  keyboard.o : keyboard.c keyboard.h
8.  gcc -c  keyboard.c
9.  clean:
10. rm *.o
```

Để biên dịch chương trình này ta chỉ cần ra lệnh *make* trong thư mục chứa file này.

Trong makefile này chứa tất cả 5 luật, luật đầu tiên có đích là *editor* được gọi là đích ngầm định. Đây chính là file mà make sẽ phải tạo ra, editor có 3 dependencies editor.o, screen.o, keyboard.o. Tất cả các file này phải tồn tại thì mới tạo ra được đích trên. Dòng thứ 2 là lệnh mà make sẽ gọi thực hiện để tạo ra đích trên. Các dòng tiếp theo là các đích và các lệnh tương ứng để tạo ra các file đối tượng (object).

2.3 Sử dụng nhãn file (mô tả file – file descriptor)

Trong Linux, để làm việc với file ta sử dụng nhãn file (file descriptor). Một trong những thuận lợi trong Linux và các hệ thống Uầ IX khác là giao diện file làm như nhau đối với nhiều loại thiết bị. Đĩa từ, các thiết bị vào/ra, cổng song song, giả máy trạm (pseudoterminal), cổng máy in, bảng mạch âm thanh, và chuột được quản lý như các thiết bị đặc biệt giống như các tệp

thông thường để lập trình ứng dụng. Các socket TCP/IP và miền, khi kết nối được thiết lập, sử dụng mô tả file như thể chúng là các file chuẩn. Các ống (pipe) cũng tương tự các file chuẩn.

Một nhãn file file đơn giản chỉ là một số nguyên được sử dụng như chỉ mục (index) vào một bảng các file mở liên kết với từng tiến trình. Các giá trị 0, 1 và 2 liên quan đến các dòng (streams) vào ra chuẩn: *stdin, stderr* và *stdout*; ba dòng đó thường kết nối với máy của người sử dụng và có thể được chuyển tiếp (redirect).

Một số lời gọi hệ thống sử dụng mô tả file. Hầu hết các lời gọi đó trả về giá trị -1 khi có lỗi xảy ra và biến *errno* ghi mã lỗi. Mã lỗi được ghi trong trang chính tuỳ theo từng lời gọi hệ thống. Hàm *perror()* được sử dụng để hiển thị nội dung thông báo lỗi dựa trên mã lỗi.

Hàm open()

Lời gọi open() sử dụng để mở một file. Khuôn mẫu của hàm và giải thích tham số và cờ của nó được cho dưới đây:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

Đối số *pathname* là một xâu chỉ ra đường dẫn đến file sẽ được mở. Thông số thứ ba xác định chế độ của file Unix (các bit được phép) được sử dụng khi tạo một file và nên được sử dụng khi tạo một file. Tham số flags nhận một trong các giá trị O_RDOẩ LY, O_WROẩ LY hoặc O_RDWR

Cờ	*Chú giải*
O_RDOẩ LY	Mở file để đọc
O_WROẩ LY	Mở file để ghi
O_RDWR	Mở file để đọc và ghi
O_CREAT	Tạo file nếu chưa tồn tại file đó
O_EXCL	Thất bại nếu file đã có
O_ẩ OCTTY	Không điều khiển tty nếu tty đã mở và tiến trình không điều khiển tty
O_TRUẩ C	Cắt file nếu nó tồn tại
O_APPEẩ D	ẩ ối thêm và con trỏ đặt ở cuối file
O_ẩ Oẩ BLOCK	ẩ ếu một quá trình không thể hoàn thành mà không có trễ, trả về trạng thái trước đó
O_ẩ ODELAY	Tương tự O_ẩ Oẩ BLOCK
O_SYẩ C	Thao tác sẽ không trả về cho đến khi dữ liệu được ghi vào đĩa hoặc thiết bị

khác

open() trả về một mô tả file nếu không có lỗi xảy ra. Khi có lỗi , nó trả về giá trị -1 và đặt giá trị cho biến errno. Hàm create() cũng tương tự như open() với các cờ O_CREATE | O_WROẩ LY | O_TRUẩ C

Hàm close()

Chúng ta nên đóng nhãn file khi đã thao tác xong với nó. Chỉ có một đối số đó là số mô tả file mà lời gọi open() trả về. Dạng của lời gọi close() là:

```
#include <unistd.h>
int close(int fd);
```

Tất cả các khoá (lock) do tiến trình xử lý trên file được giải phóng, cho dù chúng được đặt mô tả file khác. ẩ ếu quá trình đóng file làm cho bộ đếm liên kết bằng 0 thì file sẽ bị xoá. ẩ ếu đây là mô tả file cuối cùng liên kết đến một file được mở thì bản ghi ở bảng file mở được giải phóng. ẩ ếu không phải là một file bình thường thì các hiệu ứng không mong muốn có thể xảy ra.

Hàm read()

Lời gọi hệ thống *read()* sử dụng để đọc dữ liệu từ file tương ứng với một mô tả file.

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);
```

Đối số đầu tiên là mô tả file mà được trả về từ lời gọi *open()* trước đó. Đối số thứ hai là một con trỏ tới bộ đệm để sao chép dữ liệu và đối số thứ ba là số byte sẽ được đọc. *read()* trả về số byte được đọc hoặc -1 nếu có lỗi xảy ra.

Hàm write()

Lời gọi hệ thống *write()* sử dụng để ghi dữ liệu vào file tương ứng với một mô tả file.

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t count);
```

Đối số đầu tiên là số mô tả file được trả về từ lời gọi *open()* trước đó. Đối số thứ hai là con trỏ tới bộ đệm (để sao chép dữ liệu, có dung lượng đủ lớn để chứa dữ liệu) và đối số thứ ba xác định số byte sẽ được ghi. write() trả về số byte đọc hoặc -1 nếu có lỗi xảy ra.

Hàm ftruncate()

Lời gọi hệ thống ftruncate() cắt file tham chiếu bởi mô tả file fd với độ dài được xác định bởi tham số length.

```
#include <unistd.h>
int ftruncate(int fd, size_t length);
```

Trả về giá trị 0 nếu thành công và -1 nếu có lỗi xảy ra.

Hàm lseek()

Hàm lseek() đặt vị trí đọc và ghi hiện tại trong file được tham chiếu bởi mô tả file files tới vị trí offset:

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fildes, off_t offset, int whence);
```

Phụ thuộc vào giá trị của whence, giá trị của offset là vị trí bắt đầu (SEEK_SET), vị trí hiện tại (SEEK_CUR), hoặc cuối file (SEEK_Eẩ D). Giá trị trả về là kết quả của offset: bắt đầu file, hoặc một giá trị của off_t , giá trị -1 nếu có lỗi.

Hàm fstat()

Hàm *fstat()* đưa ra thông tin về file thông qua việc nhãn các file, nơi kết quả của struct stat được chỉ ra ở con trỏ chỉ đến *buf()*. Kết quả trả về giá trị 0 nếu thành công và nhận giá trị -1 nếu sai (kiểm tra lỗi).

```
#include <sys/stat.h>
#include <unistd.h>
int fstat(int filedes, struct stat *buf);
```

Sau đây là định nghĩa của struct stat

```
struct stat
{
     dev_t           st_dev;    / * thiết bị */
     int_t           st_ino;    /* inode */
     mode_t          st_mode;   /* chế độ bảo vệ */
     nlink_t         st_nlink;  /* số lượng các liên kết cứng  */
     uid_t           st_uid;    /* số hiệu của người chủ */
     gid_t           st_gid;    /* số hiệu nhóm của người chủ*/
     dev_t           st_rdev;   /* kiểu thiết bị */
     off_t           st_size;   /* kích thước bytes */
     unsigned long   st_blksize;     /* kích thước khối*/
     unsigned long   st_blocks; /* Số lượng các khối đã sử dụng*/
     time_t          st_atime;  /* thời gian truy cập cuối cùng*/
     time_t          st_mtime;  /* thời gian cập nhật cuối cùng */
     time_t          st_ctime;  /* thời gian thay đổi cuối cùng */
};
```

Hàm fchown()

Lời gọi hệ thống fchown() cho phép ta thay đổi người chủ và nhóm người chủ kết hợp với việc mở file.

```
#include <sys/types.h>
#include <unistd.h>
int fchown(int  fd, uid_t  owner, gid_t  group);
```

Tham số đầu tiên là nhãn file, tham số thứ hai là số định danh của người chủ, và tham số thứ ba là số định danh của nhóm người chủ. ẩ gười dùng hoặc nhóm người dùng sẽ được phép sử dụng khi giá trị -1 thay đổi. Giá trị trả về là 0 nếu thành công và –1 nếu gặp lỗi (kiểm tra biến errno).

Thông thường người dùng có thể thay đổi nhóm các file thuộc về họ. Chỉ *root* mới có quyền thay đổi người chủ sở hữu của nhiều nhóm.

Hàm fchdir( )

Lời gọi hàm *fchdir()* thay đổi thư mục bằng cách mở file được nhãn bởi biến *fd.* Giá trị trả về là 0 nếu thành công và –1 nếu có lỗi (kiểm tra biến errno).

```
#include <unistd.h>
int fchdir(int fd);
```

Một ví dụ về cách sử dụng các hàm thao tác với file

```
/* filedes_io.c */
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <fcntl.h>
#include <unistd.h>
#include <assert.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>  /*for print */
char sample1[] = "This is sample data 1\n";
char sample2[] = "This is sample data 2\n";
char data[16];
main()
{
      int fd;
      int rc;
      struct  stat statbuf;
      printf("Creating file\n");
      fd = open("junk.out", O_WRONLY | O_CREAT| O_TRUNC, 0666);
      assert(fd>=0);
      rc = write(fd, sample1, strlen(sample1) );
      assert(fd>=0);
```

```
rc = write(fd, sample1, strlen(sample1));
assert(rc == strlen(sample1));
close(fd);
printf("Appending to file\n");
fd = open("junk.out", O_WRONLY| O_APPEND);
assert(fd>=0);
printf("locking file\n");
rc = flock(fd, LOCK_EX);
assert(rc == 0);
printf("sleeping for 10 seconds\n");
sleep(10);
printf("writing data\n");
rc = write(fd, sample2, strlen(sample2));
assert(rc == strlen(sample2));
printf("unlocking file\n");
rc = flock(fd, LOCK_UN);
assert(rc == 0);
close(fd);
printf("Reading file \n");
fd = open("junk.out", O_RDONLY);
assert(fd >=0);
while (1)
{
      rc = read(fd, data, sizeof (data));
      if(rc > 0) {
            data[rc]=0;
            printf(" Data read(rc = %d): <%s>\n", rc, data);
      }
      else if(rc == 0) {
            printf(" End of file read \n");
            break;
      }
      else
      {
            perror("read error");
            break;
      }
}
close(fd);
printf(" Fiddling with inode\n");
```

```
        fd = open (" junk.out", O_RDONLY);
        assert (fd >= 0);
        printf (" changing file mode\n");
        rc = fchmod ( fd, 0600);
        assert(rc == 0);
        if(getuid () == 0 )
        {
              printf("changing file owner \n ");
              rc = fchown(fd, 99, 99);
              assert(rc == 0);
        } else
        {
              printf("not changing file owner\n");
        }
        fstat(fd, &statbuf);
        printf(" file mode = o% (octal) \n", statbuf.st_mode);
        printf("Owner uid = %d \n", statbuf.st_uid);
        printf(" Owner gid = %d \n", statbuf.st_uid);
        close(fd);
}
```

2.4 Thư viện liên kết

Phần này sẽ giới thiệu cách tạo ra và sử dụng thư viện (các module chương trình đã được viết và được tái sử dụng nhiều lần). Thư viện gốc của C/C++ trên Linux chính là *glibc,* thư viện này cung cấp cho người dùng rất nhiều lời gọi hệ thống. Các thư viện trên Linux thường được tổ chức dưới dạng tĩnh (static library), thư viện chia sẻ (shared library) và động (dynamic library - giống như DLL trên MS Windows).

Thư viện tĩnh được liên kết cố định vào trong chương trình trong quá trình liên kết. Thư viện dùng chung được nạp vào bộ nhớ trong khi chương trình bắt đầu thực hiện và cho phép các ứng dụng cùng chia sẻ loại thư viện này. Thư viện liên kết động được nạp vào bộ nhớ chỉ khi nào chương trình gọi tới.

Thư viện liên kết tĩnh

Thư viện tĩnh và các thư viện dùng chung (shared library) là các file chứa các file được gọi là các module đã được biên dịch và có thể sử dụng lại được. Chúng được lưu trữ dưới một định dạng đặc biệt cùng với một bảng (hoặc một bản đồ) phục vụ cho quá trình liên kết và biên dịch. Các thư viện liên kết tĩnh có phần mở rộng là .a.

Để sử dụng các module trong thư viện ta cần thêm phần *#include* file tiêu đề (header) vào trong chương trình nguồn và khi liên kết (sau quá trình biên dịch) thì liên kết với thư viện đó. Dưới đây là một ví dụ về cách tạo và sử dụng một thư viên liên kết tĩnh. Có 2 phần trong ví dụ này, phần thứ nhất là mã nguồn cho thư viện và phần thứ 2 cho chương trình sử dụng thư viện.

```
/* Mã nguồn file liberr.h */
#ifndef   _LIBERR_H
#define  _LIBERR_H
#include <stdarg.h>
/* in ra một thông báo lỗi tới việc gọi stderr và return hàm gọi */
void err_quit(const char *fmt, … );
/*  in ra một thông điệp lỗi cho logfile và trả về hàm gọi */
void log_ret(char *logfile, const char *fmt, …);
 /* in ra một thông điệp lỗi cho logfile và thoát */
void log_quit( char *logfile, const char  *fmt , …);
/* in ra một  thông  báo lỗi và trả lại hàm gọi */
void err_prn(const char *fmt, va_list ap, char *logfile);
#endif //_LIBERR_H

/* Mã nguồn file liberr.c*/
#include <errno.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include  "liberr.h"
#define MAXLINELEN 500
void err_ret(const char *fmt, ...)
{
           va_list ap;
           va_start(ap, fmt);
           err_prn(fmt, ap, NULL);
           va_end(ap);
           return;
}
void err_quit(const char *fmt, ...)
{
           va_list ap;
           va_start(ap, fmt);
           err_prn(fmt, ap, NULL);
```

```
            va_end(ap);
            exit(1);
}
void log_ret(char *logfile, const char *fmt, ...)
{
            va_list ap;
            va_start(ap, fmt);
            err_prn(fmt,ap, logfile);
            va_end(ap);
            return;
}
void log_quit(char *logfile, const char *fmt,... )
{
            va_list ap;
            va_start(ap, fmt);
            err_prn(fmt, ap,logfile);
            va_end(ap);
            exit(1);
}
extern void err_prn( const char *fmt, va_list ap, char *logfile)
{
            int save_err;
            char buf[MAXLINELEN];
            FILE *plf;
            save_err = errno;
            vsprintf(buf,fmt, ap);
            sprintf( buf+strlen(buf),  ": %s", strerror(save_err));
            strcat(buf, "\n");
            fflush(stdout);
            if(logfile !=NULL){
            if((plf=fopen(logfile, "a") ) != NULL){
                  fputs(buf, plf);
                  fclose(plf);
            }else
                  fputs("failed to open log file \n", stderr);
            }else fputs(buf, stderr);
            fflush(NULL);
            return;
}
```

Để tạo một thư viện tĩnh, bước đầu tiên là dịch đoạn mã của form đối tượng:

```
$gcc –H –c liberr.c –o liberr.o
```

tiếp theo:

```
$ar rcs liberr.a  liberr.o
```

```
/* Mã nguồn file testerr.c*/
#include <stdio.h>
#include <stdlib.h>
#include "liberr.h"
#define ERR_QUIT_SKIP 1
#define LOG_QUIT_SKIP 1
int main(void)
{
   FILE *pf;
   fputs("Testing err_ret()...\n", stdout);
   if((pf = fopen("foo", "r")) == NULL)
   err_ret("%s %s", "err_ret()", "failed to open foo");
   fputs("Testing log_ret()...\n", stdout);
   if((pf = fopen("foo", "r")) == NULL);
   log_ret("errtest.log", "%s %s", "log_ret()",
   "failed to open foo");
   #ifndef ERR_QUIT_SKIP
   fputs("Testing err_quit()...\n", stdout);
   if((pf = fopen("foo", "r")) == NULL)
   err_ret("%s %s", "err_quit()", "failed to open foo");
   #endif /* ERR_QUIT_SKIP */
   #ifndef LOG_QUIT_SKIP
   fputs("Testing log_quit()...\n", stdout);
   if((pf = fopen("foo", "r")) == NULL)
   log_ret("errtest.log",  "%s  %s",  "log_quit()",  "failed  to  open
     foo");
   #endif /* LOG_QUIT_SKIP */
   return EXIT_SUCCESS;
}
```

Biên dịch chương trình kiểm tra, ta sử dụng dòng lệnh:

```
$ gcc -g errtest.c -o errtest -L. -lerr
```

Tham số -L. chỉ ra đường dẫn tới thư mục chứa file thư viện là thư mục hiện thời, tham số –lerr chỉ rõ thư viện thích hợp mà chúng ta muốn liên kết. Sau khi dịch ta có thể kiểm tra bằng cách chạy chương trình.

Thư viện dùng chung

Thư viện dùng chung có nhiều thuận lợi hơn thư viện tĩnh.Thứ nhất, thư viện dùng chung tốn ít tài nguyên hệ thống, chúng sử dụng ít không gian đĩa vì mã nguồn thư viện dùng chung không biên dịch sang mã nhị phân nhưng được liên kết và được dùng tự động mỗi lần dùng.

Chúng sử dụng ít bộ nhớ hệ thống vì nhân chia sẻ bộ nhớ cho thư viện dùng chung này và tất cả các chương trình đều sử dụng chung miền bộ nhớ này. Thứ 2, thư viện dùng chung nhanh hơn vi chúng chỉ cần nạp vào một bộ nhớ. Lí do cuối cùng là mã nguồn trong thư viện dùng chung dễ bảo trì. Khi các lỗi được sửa hay thêm vào các đặc tính, người dùng cần sử dụng thư viện nâng cấp. Đối với thư viện tĩnh, mỗi chương trình khi sử dụng thư viện phải biên dịch lại.

Trình liên kết (linker)/module tải (loader) *ld.so* liên kết tên biểu tượng tới thư viện dùng chung mỗi lần chạy. Thư viện dùng chung có tên đặc biệt (gọi là soname), bao gồm tên thư viện và phiên bản chính. Ví dụ: tên đầy đủ của thư viện C trong hệ thống là libc.so.5.4.46, tên thư viện là libc.so, tên phiên bản chính là 5, tên phiên bản phụ là 4, 46 là mức vá (patch level). ầ hư vậy, soname thư viện C là libc.5. Thư viện libc6 có soname là libc.so.6, sự thay đổi phiên bản chính là sự thay đổi đáng kể thư viện. Phiên bản phụ và patch level thay đổi khi lỗi được sửa nhưng soname không thay đổi và bản mới có sự thay khác biệt đáng kể so với bản cũ.

Các chương trình ứng dụng liên kết dựa vào soname. Tiện ích *idconfig* tạo một biểu tượng liên kết từ thư viện chuẩn libc.so.5.4.46 tới soname libc.5 và lưu trữ thông tin này trong /etc/ld.so.cache. Trong lúc chạy, ld.so đọc phần lưu trữ, tìm soname thích hợp và nạp thư viện hiện tai vào bộ nhớ, kết nối hàm ứng dụng gọi tới đối tượng thích hợp trong thư viện.

Các phiên bản thư viện khác nhau nếu:

- Các giao diện hàm đầu ra thay đổi.
- Các giao diện hàm mới được thêm.
- Chức năng hoạt động thay đổi so với đặc tả ban đầu
- Cấu trúc dữ liệu đầu ra thay đổi
- Cấu trúc dữ liệu đầu ra được thêm

Để duy trì tính tương thích của thư viện, cần đảm bảo các yêu cầu:

- Không thêm vào những tên hàm đã có hoặc thay đổi hoạt động của nó
- Chỉ thêm vào cuối cấu trúc dữ liệu đã có hoặc làm cho chúng có tính tuỳ chọn hay được khởi tạo trong thư viện
- Không mở rộng cấu trúc dữ liệu sử dụng trong các mảng

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Xây dựng thư viện dùng chung hơi khác so với thư viện tĩnh, quá trình xây dựng thư viện dùng chung được minh hoạ dưới đây:

- Khi biên dịch file đối tượng, sử dụng tùy chọn -fpic của gcc nó sẽ tạo ra mã độc lập vị trí (position independence code) từ đó có thể liên kết hay sử dụng ở bất cứ chỗ nào.
- Không loại bỏ file đối tượng và không sử dụng các tùy chọn *–fomit –frame -pointer* của *gcc*, vì nếu không sẽ ảnh hưởng đến quá trình gỡ rối (debug).
- Sử dụng tuỳ chọn *-shared* and *–soname* của *gcc*
- Sử dụng tuỳ chọn *–Wl* của *gcc* để truyền tham số tới trình liên kết *ld*.
- Thực hiện quá trình liên kết dựa vào thư viện C, sử dụng tuỳ chọn *–l* của *gcc*

Trở lại thư viện xử lý lỗi, để tạo thư viện dùng chung trước hết xây dựng file đối tượng:

```
$ gcc -fPiC -g -c liberr.c -o liberr.o
```

Tiếp theo liên kết thư viện:

```
$ gcc -g -shared -Wl,-soname,liberr.so -o liberr.so.1.0.0 liberr.o -lc
```

Vì không thể cài đặt thư viện này như thư viện hệ thống trong /usr hay /usr/lib chúng ta cần tạo 2 kiên kết, một cho soname:

Và cho trình liên kết khi kết nối dựa vào liberr, sử dụng *–lerr*:

```
$ ln -s liberr.so.1.0.0 liberr.so
```

Bây giờ, để dử dụng thư viện dùng chung mới chúng ta quay lại chương trình kiểm tra, chúng ta cần hướng trình liên kết tới thư viện nào để sử dụng và tìm nó ở đâu, vì vậy chúng ta sẽ sử dụng tuỳ chọn –l và –L:

```
$ gcc -g errtest.c -o errtest -L. -lerr
```

Cuối cùng để chạy chương trình, chúng ta cần chỉ cho ld.so nơi để tìm thư viện dùng chung :

```
$ LD_LIBRARY_PATH=$(pwd) ./errtest
```

Sử dụng đối tượng dùng chung theo cách động

Một cách để sử dụng thư viện dùng chung là nạp chúng tự động mỗi khi chạy không giống như những thư viện liên kết và nạp một cách tự động. Ta có thể sử dụng giao diện dl (dynamic loading) vì nó tạo sự linh hoạt cho lập trình viên hay người dùng.

Giả sử ta đang tạo một ứng dụng sử lý đồ hoạ. Trong ứng dụng, ta biểu diễn dữ liệu ở một dạng không theo chuẩn nhưng lại thuận tiện cho ta xử lý, và ta cần có nhu cầu chuyển dữ liệu đó ra các định dạng thông dụng đã có (số lượng các định dạng này có thể có hàng trăm loại) hoặc đọc dữ liệu từ các định dạng mới này vào để xử lý. Để giải quyết vấn đề này ta có thể sử dụng giải pháp là thư viện. ấ hưng khi có thêm một định dạng mới thì ta lại phải biên dịch lại chương trình. Đây lại là một điều không thích hợp lắm. Khả năng sử dụng thư viện động sẽ giúp

Formatted: Bullets and Numbering

ta giải quyết vấn đề vừa gặp phải. Giao diện dl cho phép tạo ra giao diện (các hàm) đọc và viết chung không phụ thuộc vào định dạng của file ảnh. Để thêm hoặc sửa các định dạng của file ảnh ta chỉ cần viết thêm một module để đảm nhận chức năng đó và báo cho chương trình ứng dụng biết là có thêm một module mới bằng cách chỉ cần thay đổi một file cấu hình trong một thư mục xác định nào đó.

Giao diện dl (cũng đơn thuần được xây dựng như một thư viện - thư viện libdl) chứa các hàm để tải (load), tìm kiếm và giải phóng (unload) các đối tượng chia sẻ. Để sử dụng các hàm này ta thêm file <dlfcn.h> vào phần #include vào trong mã nguồn, và khi dịch thì liên kết nó với thư viện libdl bằng cách sử dụng tham số và tên –ldl trong dòng lệnh dịch.

dl cung cấp 4 hàm xử lí các công việc cần thiết để tải, sử dụng và giải phóng đối tượng dùng chung.

Truy cập đối tượng chia sẻ

Để truy cập một đối tượng chia sẻ, dùng hàm dlopen() có đặc tả như sau:

```
void *dlopen(const char *filename, int flag);
```

dlopen() truy cập đối tượng chia sẻ bằng filename và bằng cờ. Filename có thể là đường dẫn đầy đủ, tên file rút gọn hay ả ULL. ả ếu là ả ULL dlopen() mở chương trình đang chạy, đó là chương trình của bạn, nếu filename là đường dẫn dlopen() mở file đó, nếu là tên rút gọn dlopen() sẽ tìm trong vị trí sau để tìm file:

```
$LD_ELF_LiBRARY_PATH,
$LD_LIBRARY_PATH, /etc/ld.so.cache, /usr/lib, và /lib.
```

Cờ có thể là RTLD_LAZY, có nghĩa là các kí hiệu (symbol) hay tên hàm từ đối tượng truy cập sẽ được tìm mỗi khi chúng được gọi, hoặc cờ có thể là RTLD_ả OW, có nghĩa tất cả kí hiệu từ đối tượng truy cập sẽ được tìm trước khi hàm dlopen() trả về. dlopen() trả điều khiển tới đối tượng truy nhập nếu nó tìm thấy từ filename hay trả về giá trị ả ULL nếu không tìm thấy.

Sử dụng đối tượng chia sẻ

Trước khi có thể sử dụng mã nguồn trong thư viện ta phải biết đang tìm cái gì và tìm ở đâu. Hàm dlsym() sẽ giúp điều đó:

```
void *dlsym(void *handle, char *symbol);
```

dlsym() tìm kí hiệu hay tên hàm trong truy cập và trả lại con trỏ kiểu void tới đối tượng hay ả ULL nếu không thành công.

Kiểm tra lỗi

Hàm dlerror() sẽ giúp ta kiểm tra lỗi khi sử dụng đối tượng truy cập động:

const char *dlerror(void);

ả ếu một trong các hàm lỗi, dlerror() trả về thông báo chi tiết lỗi và gán giá trị ả ULL cho phần bị lỗi.

Giải phóng đối tượng chia sẻ

Để bảo vệ tài nguyên hệ thống đặc biệt bộ nhớ, khi ta sử dụng xong module trong một đối tượng chia sẻ, thì giải phóng chúng. Hàm dlclose() sẽ đóng đối tượng chia sẻ:

int dlclose(void *handle);

Sử dụng giao diện dl

Để minh hoạ cách sử dụng dl,chúng ta quay lại thư viện xử lí lỗi, sử dụng một chương trình khác như sau:

```
/*
* Mã nguồn chương trình dltest.c
* Dynamically load liberr.so and call err_ret()
*/
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
 int main(void)
 {
     void *handle;
     void (*errfcn)();
     const char *errmsg;
     FILE *pf;
     handle = dlopen("liberr.so", RTLD_NOW);
     if(handle == NULL) {
        fprintf(stderr, "Failed to load liberr.so: %s\n", dlerror());
        exit(EXIT_FAILURE);
     }
     dlerror();
     errfcn = dlsym(handle, "err_ret");
     if((errmsg = dlerror()) != NULL) {
           fprintf(stderr, "Didn't find err_ret(): %s\n", errmsg);
           exit(EXIT_FAILURE);
     }
     if((pf = fopen("foobar", "r")) == NULL)
           errfcn("couldn't open foobar");
     dlclose(handle);
     return EXIT_SUCCESS;
 }
```

Biên dịch ví dụ trên bằng lệnh

```
$ gcc -g -Wall dltest.c -o dltest -ldl
```

ả hư tacó thể thấy, chúng ta không liên kết dựa vào liberr hay liberr.h trong mã nguồn. Tất cả truy cập tới liberr.so thông qua dl. Chạy chương trình bằng cách sau:

```
$ LD_LIBRARY_PATH=$(pwd) ./dltest
```

ả ếu thành công thì ta nhận được kết quả như sau:

```
couldn't open foobar: No such file or directory
```

2.5 Các công cụ cho thư viện

Công cụ nm

Lệnh *nm* liệt kê toàn bộ các tên hàm (symbol) được mã hoá trong file đối tượng (object) và nhị phân (binary). Lệnh *nm* sử dụng cú pháp sau: **nm [options] file**

Lệnh *nm* liệt kê những tên hàm chứa trong file. Bảng dưới liệt kê các tuỳ chọn của lệnh *nm*:

Tuỳ chọn	*Miêu tả*
-C| -demangle	Chuyển tên ký tự vào tên mức người dùng để cho dễ đọc.
-s|-print-armap	Khi sử dụng các file lưu trữ (phần mở rộng là ".a"), in ra các chỉ số của module chứa hàm đó.
-u| -undefined-only	Chỉ đưa ra các hàm không được định nghĩa trong file này, tức là các hàm được định nghĩa ở một file khác.
- l | -line-numbers	Sử dụng thông tin gỡ rối để in ra số dòng nơi hàm được định nghĩa.

Ví dụ: xem các hàm được mã hóa trong file nhị phân a.out (file nhị phân sau khi dịch filedes_io.c)

```
   bt Desktop # nm a.out                  08048be8 r __PRETTY_FUNCTION__.2764
   08049da4 d _DYNAMIC                             U __assert_fail@@GLIBC_2.0
   08049e70 d _GLOBAL_OFFSET_TABLE_       08049ef8 A __bss_start
   08048be4 R _IO_stdin_used              08049ebc D __data_start
            w _Jv_RegisterClasses         08048b90 t __do_global_ctors_aux
   08049d94 d __CTOR_END__                080485f0 t __do_global_dtors_aux
   08049d90 d __CTOR_LIST__               08049ec0 D __dso_handle
   08049d9c d __DTOR_END__                08048b50 T __fstat
   08049d98 d __DTOR_LIST__                          U __fxstat@@GLIBC_2.0
   08048d8c r __FRAME_END__                          w __gmon_start__
08049da0 d __JCR_END__                      08048b47                      T
08049da0 d __JCR_LIST__                     __i686.get_pc_thunk.bx
```

```
08049d90 d __init_array_end                    U fchown@@GLIBC_2.0
08049d90 d __init_array_start                  U flock@@GLIBC_2.0
08048ad0 T __libc_csu_fini             08048620 t frame_dummy
08048ae0 T __libc_csu_init             08048b50 W fstat
     U __libc_start_main@@GLIBC_2.0            U getuid@@GLIBC_2.0
08049ef8 A _edata                      08048644 T main
08049f0c A _end                                U open@@GLIBC_2.0
08048bc4 T _fini                       08049ec4 d p.5743
08048be0 R _fp_hw                              U perror@@GLIBC_2.0
08048470 T _init                               U printf@@GLIBC_2.0
080485a0 T _start                              U puts@@GLIBC_2.0
080485c4 t call_gmon_start                     U read@@GLIBC_2.0
      U close@@GLIBC_2.0               08049ec8 D sample1
08049ef8 b completed.5745              08049edf D sample2
08049efc B data                                U sleep@@GLIBC_2.0
08049ebc W data_start                          U write@@GLIBC_2.0
      U fchmod@@GLIBC_2.0
```

Công cụ ar

Lệnh *ar* sử dụng cú pháp sau: **ar {dmpqrtx} [thành viên] file**

Lệnh *ar* tạo, chỉnh sửa và trích các file lưu trữ. ẩ ó thường được sử dụng để tạo các thư viện tĩnh- những file mà chứa một hoặc nhiều file đối tượng chứa các chương trình con thường được sử dụng (subrountine) ở định dạng tiền biên dịch (precompiled format), lệnh *ar* cũng tạo và duy trì một bảng mà tham chiếu qua tên ký tự tới các thành viên mà trong đó chúng được định nghĩa.

```
bt Desktop # ar a.out
ar: illegal option -- .
Usage: ar [emulation options] [-]{dmpqrstx}[abcfilNoPsSuvV] [member-
name] [count] archive-file file...
       ar -M [<mri-script]
 commands:
  d            - delete file(s) from the archive
  m[ab]        - move file(s) in the archive
  p            - print file(s) found in the archive
  q[f]         - quick append file(s) to the archive
  r[ab][f][u]  -  replace  existing  or  insert  new  file(s)  into  the
archive
  t            - display contents of archive
  x[o]  - extract file(s) from the archive command specific modifiers:
```

```
  [a] - put file(s) after [member-name]
  [b] - put file(s) before [member-name] (same as [i])
  [N] - use instance [count] of name
  [f] - truncate inserted file names
  [P] - use full path names when matching
  [o] - preserve original dates
  [u] -only replace files that are newer than current archive contents
eneric modifiers:
  [c] - do not warn if the library had to be created
  [s] - create an archive index (cf. ranlib)
  [S] - do not build a symbol table
  [v] - be verbose
  [V] - display the version number
  @<file> - read options from <file> emulation options:
  No emulation specific options
ar: supported targets: elf32-i386 a.out-i386-linux efi-app-ia32 elf64-
x86-64 elf64-little elf64-big elf32-little elf32-big srec symbolsrec
tekhex binary ihex trad-core
```

2.6 Biến môi trường và file cấu hình

Chương trình tải (loader) và trình liên kết (linker) *ld.so* sử dụng 2 biến môi trường. Biến thứ nhất là $LD_LIBRARY, chứa danh sách các thư mục chứa các file thư viện được phân cách bởi dấu hai chấm để tìm ra các thư viện cần thiết khi chạy. ả ó giống như biến môi trường $PATH. Biến môi trường thứ hai là $LD_PRELOAD, một danh sách các thư viện được người dùng thêm vào được phân cách nhau bởi khoảng trống (space).

ld.so cũng cho phép sử dụng 2 file cấu hình mà có cùng mục đích với biến môi trường được đề cập ở trên. File /etc/ld.so.conf chứa một danh sách các thư mục mà chương trình tải và trình liên kết (loader/linker) nên tìm kiếm các thư viện chia sẻ bên cạnh /usr/lib và /lib. /etc/ld.so.preload chứa danh sách các file thư viện được phân cách bằng một khoảng trống các thư viện này là thư viện người dùng tạo ra.

2.7 Sử dụng gdb để gỡ lỗi

Sử dụng gdb để gỡ lỗi khi biên dịch file nguồn c, cú pháp: **# gdb fileName**

Bảng các lệnh cơ bản của gdb:

Lệnh	*Mô tả*
file	ả ạp file thực thi sẽ được gỡ lỗi

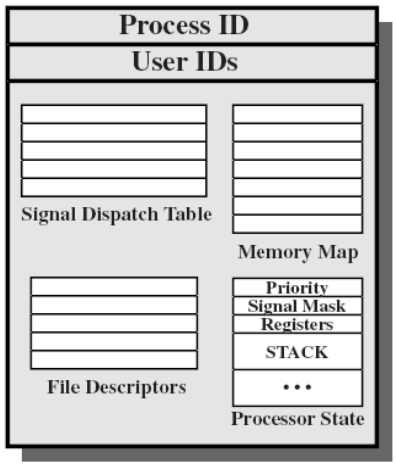
kill	Kết thúc chương trình đang được gỡ lỗi.
list	Liệt kê các đoạn của mã nguồn được sử dụng để tạo file thực thi.
next	Tiến lên một dòng trong mã nguồn tại hàm hiện thời, không dùng để chuyển đến những hàm khác.
step	Tiến lên một dòng trong mã nguồn tại hàm hiện thời, có thể dùng để chuyển đến những hàm khác.
run	Thực thi chương trình đã được gỡ lỗi.
quit	Kết thúc gdb.
watch	Cho phép ta xem giá trị biến số của chương trình mỗi khi giá trị thay đổi.
break	Thiết lập một điểm ngắt trong mã; cho phép chương trình ngừng làm việc gì đó mỗi khi gặp điểm ngắt này.
make	Cho phép thực thi lại chương trình mà không cần thoát khỏi gdb hoặc sử dụng cửa sổ khác.
shell	Cho phép thực thi lệnh Shell mà không cần ngừng gdb.

CHƯƠNG 5: QUẢN LÝ TÀI NGUYÊN VÀ TRUYỀN THÔNG TRONG LINUX

Trong chương này phạm vi tìm hiểu "tài nguyên" gồm 3 nội dung là: Tiến trình, đĩa cứng, người dùng hay nhóm người dùng.

1. Quản lý tiến trình

Trong Linux, bất cứ chương trình nào đang chạy đều được coi là một tiến trình. Có thể có nhiều tiến trình cùng chạy một lúc. Ví dụ dòng lệnh **ls -l | sort | more** sẽ khởi tạo ba tiến trình: **ls**, **sort** và **more**.



Hình 5.1Cấu trúc của một tiến trình trong Unix

Tiến trình có thể trải qua nhiều trạng thái khác nhau và tại một thời điểm một tiến trình rơi vào một trong các trạng thái đó. Bảng dưới đây giới thiệu các trạng thái cơ bản của tiến trình trong Linux.

Ký hiệu	*Ý nghĩa*
D	(uninterruptible sleep) ở trạng thái này tiến trình bị treo và không thể chạy lại nó bằng một tín hiệu.
R	(runnable) trạng thái sẵn sàng thực hiện, tức là tiến trình có thể thực hiện được nhưng chờ đến lượt thực hiện vì một tiến trình khác đang có CPU.
S	(sleeping) trạng thái tạm dừng, tức là tiến trình tạm dừng không hoạt động (20 giây hoặc ít hơn)
T	(traced or stopped) trạng thái dừng, tiến trình có thể bị treo bởi một tiến trình ngoài
Z	(zombie process) tiến trình đã kết thúc thực hiện, nhưng nó vẫn được tham chiếu

W trong hệ thống

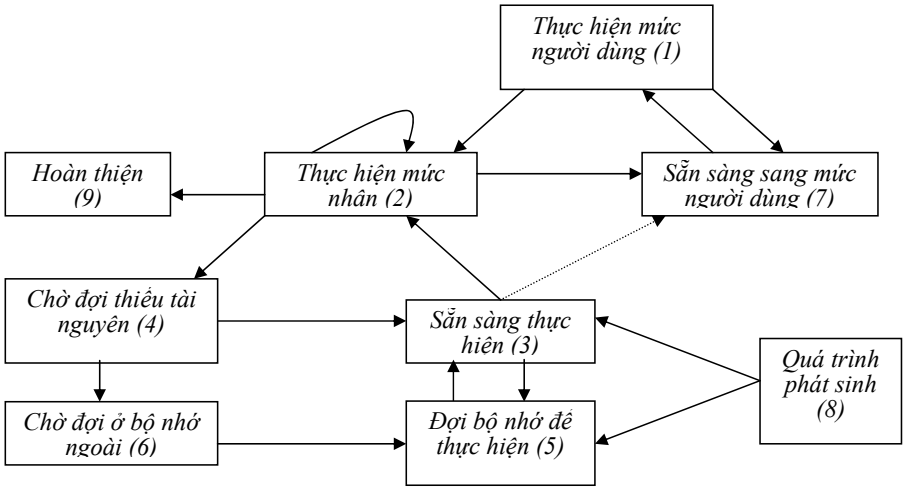
< không có các trang thường trú

ả tiến trình có mức ưu tiên cao hơn

L tiến trình có mức ưu tiên thấp hơn

có các trang khóa bên trong bộ nhớ

Sơ đồ biểu diễn các trạng thái và việc chuyển trạng thái trong Uả IX được trình bày trong hình dưới đây (Số hiệu trạng thái quá trình xem trong hình vẽ).



Hình 5.2 Sơ đồ trạng thái các tiến trình

Khi quá trình được phát sinh nó ở trạng thái (8), tùy thuộc vào tình trạng bộ nhớ quá trình được phân phối bộ nhớ trong (3) hay bộ nhớ ngoài (5). Trạng thái (3) thể hiện quá trình đã sẵn sáng thực hiện, các thành phần của nó đã ở bộ nhớ trong chờ đợi CPU để thực hiện. Việc thực hiện tiếp theo tùy thuộc vào trạng thái trước đó của nó. ả ếu lần đầu phát sinh, nó cần đi tới thực hiện mức nhân để hoàn thiện công việc lời gọi fork sẽ từ trạng thái (3) sang trạng thái (1), trong trường hợp khác, từ trạng thái (3) nó đi tới trạng thái chờ dợi CPU ở mức người dùng (7). Trong trạng thái thực hiện ở mức người dùng (1), quá trình đi tới trạng thái (2) khi gặp lời gọi hệ thống hoặc hiện tượng ngắt xảy ra. Từ trạng thái (1) tới trạng thái (7) khi hết lượng tử thời gian.

Trạng thái (4) là trạng thái chờ đợi trong bộ nhớ còn trạng thái (6) thể hiện việc chờ đợi trong bộ nhớ ngoài.

Cung chuyển từ trạng thái (2) vào ngay trạng thái (2) xảy ra khi ở quá trình ở trạng thái thực hiện mức nhân, nhân hệ thống gọi các hàm xử lý ngắt tương ứng.

2. Các lệnh cơ bản trong quản lý tiến trình

2.1 Sử dụng lệnh ps trong quản lý tiến trình

Linux cung cấp cho người dùng hai cách thức nhận biết có những chương trình nào đang chạy trong hệ thống. Cách dễ hơn, đó là lệnh jobs sẽ cho biết các quá trình nào đã dừng hoặc là được chạy trong chế độ nền.

Cách phức tạp hơn là sử dụng lệnh ps. Lệnh này cho biết thông tin đầy đủ nhất về các quá trình đang chạy trên hệ thống.

Ví dụ:

```
# ps
PID   TTY   TIME   CMD
7813  pts/0      00:00:00 bash
7908  pts/0      00:00:00 ps
#
```

(PID - chỉ số của tiến trình, TTY - tên thiết bị đầu cuối trên đó tiến trình được thực hiện, TIME - thời gian để chạy tiến trình, CMD - lệnh khởi tạo tiến trình).

Cú pháp lệnh ps: **ps [tùy-chọn]**

Lệnh ps có một lượng quá phong phú các tùy chọn được chia ra làm nhiều loại. Dưới đây là một số các tùy chọn hay dùng.

Các tùy chọn đơn giản:

- A, -e : chọn để hiển thị tất cả các tiến trình.
- T : chọn để hiển thị các tiến trình trên trạm cuối đang chạy.
- a : chọn để hiển thị tất cả các tiến trình trên một trạm cuối, bao gồm cả các tiến trình của những người dùng khác.
- r : chỉ hiển thị tiến trình đang được chạy.

Chọn theo danh sách:

- C : chọn hiển thị các tiến trình theo tên lệnh.
- G : hiển thị các tiến trình theo chỉ số nhóm người dùng.
- U : hiển thị các tiến trình theo tên hoặc chỉ số của người dùng thực sự (người dùng khởi động tiến trình).
- p : hiển thị các tiến trình theo chỉ số của tiến trình.
- s : hiển thị các tiến trình thuộc về một phiên làm việc.

Formatted: Bullets and Numbering

- _____t : hiển thị các tiến trình thuộc một trạm cuối.
- _____u : hiển thị các tiến trình theo tên và chỉ số của người dùng hiệu quả.

Thiết đặt định dạng được đưa ra của các tiến trình:

- _____f : hiển thị thông tin về tiến trình với các trường sau UID - chỉ số người dùng, PID - chỉ số tiến trình, PPID - chỉ số tiến trình khởi tạo ra tiến trình, C - , STIME - thời gian khởi tạo tiến trình, TTY - tên thiết bị đầu cuối trên đó tiến trình được chạy, TIME - thời gian để thực hiện tiến trình, CMD - lệnh khởi tạo tiến trình
- _____l : hiển thị đầy đủ các thông tin về tiến trình với các trường F, S, UID, PID, PPID, C, PRI, ả I, ADDR, SZ, WCHAả , TTY, TIME, CMD
- _____o xâu-chọn : hiển thị các thông tin về tiến trình theo dạng do người dùng tự chọn thông qua xâu-chọn các kí hiệu điều khiển hiển thị có các dạng như sau:

```
%C, %cpu   % CPU được sử dụng cho tiến trình
%mem       % bộ nhớ được sử dụng để chạy tiến trình
%G    tên nhóm người dùng
%P    chỉ số của tiến trình cha khởi động ra tiến trình con
%U    định danh người dùng
%c    lệnh tạo ra tiến trình
%p    chỉ số của tiến trình
%x    thời gian để chạy tiến trình
%y    thiết bị đầu cuối trên đó tiến trình được thực hiện
```

Ví dụ: muốn xem các thông tin như tên người dùng, tên nhóm, chỉ số tiến trình, chỉ số tiến trình khởi tạo ra tiến trình, tên thiết bị đầu cuối, thời gian chạy tiến trình, lệnh khởi tạo tiến trình, hãy gõ lệnh:

```
# ps -o '%U %G %p %P %y %x %c'
USER GROUP PID PPID TTY TIME COMMAND
root root 1929 1927 pts/1 00:00:00 bash
root root 2279 1929 pts/1 00:00:00 ps
```

2.2 Hủy một tiến trình sử dụng lệnh kill

Trong một số trường hợp, sử dụng lệnh kill để hủy bỏ một tiến trình. Điều quan trọng nhất khi sử dụng lệnh *kill* là phải xác định được chỉ số của tiến trình mà chúng ta muốn hủy.

Cú pháp lệnh: **kill [tùy-chọn] <chỉ-số-của-tiến-trình>**

kill -l [tín hiệu]

Lệnh *kill* sẽ gửi một tín hiệu đến tiến trình được chỉ ra. ả ếu không chỉ ra một tín hiệu nào thì ngầm định là tín hiệu TERM sẽ được gửi.

Một số tùy chọn:

– **s** xác định tín hiệu được gửi. Tín hiệu có thể là số hoặc tên của tín hiệu. Dưới đây là một số tín hiệu hay dùng:

Số	*Tên*	*Ý nghĩa*
1	SIGHUP	(hang up) đây là tín hiệu được gửi đến tất cả các tiến trình đang chạy trước khi logout khỏi hệ thống
2	SIGIả T	(interrupt) đây là tín hiệu được gửi khi nhấn CTRL+c
9	SIGKILL	(kill) tín hiệu này sẽ dừng tiến trình ngay lập tức
15	SIGTERM	tín hiệu này yêu cầu dừng tiến trình ngay lập tức, nhưng cho phép chương trình xóa các file tạm.

– p lệnh kill sẽ chỉ đưa ra chỉ số của tiến trình mà không gửi một tín hiệu nào.

– l hiển thị danh sách các tín hiệu mà lệnh kill có thể gửi đến các tiến trình (các tín hiệu này có trong file /usr/include/Linux/signal.h)

Ví dụ:

```
# ps
PID TTY TIME CMD
2240 pts/2 00:00:00 bash
2276 pts/2 00:00:00 man
2277 pts/2 00:00:00 more
2280 pts/2 00:00:00 sh
2281 pts/2 00:00:00 sh
2285 pts/2 00:00:00 less
2289 pts/2 00:00:00 man
2291 pts/2 00:00:00 sh
2292 pts/2 00:00:00 gunzip
2293 pts/2 00:00:00 less
# kill 2277
PID TTY TIME CMD
2240 pts/2 00:00:00 bash
2276 pts/2 00:00:00 man
2280 pts/2 00:00:00 sh
2281 pts/2 00:00:00 sh
2285 pts/2 00:00:00 less
2289 pts/2 00:00:00 man
2291 pts/2 00:00:00 sh
2292 pts/2 00:00:00 gunzip
2293 pts/2 00:00:00 less
```

2.3 Cho máy ngừng hoạt động một thời gian với lệnh sleep

ẩ ếu muốn cho máy nghỉ một thời gian mà không muốn tắt vì ngại khởi động lại thì cần dùng lệnh sleep.

Cú pháp: **sleep [tuỳ-chọn]... NUMBER[SUFFIX]**

– ẩ UMBER: số giây(s) ngừng hoạt động.

– SUFFIX : có thể là giây(s) hoặc phút(m) hoặc giờ hoặc ngày(d)

Các tùy chọn:

– **-**help hiện thị trợ giúp và thoát

– **-** version hiển thị thông tin về phiên bản và thoát

2.4 Xem cây tiến trình với lệnh pstree

Đã biết lệnh để xem các tiến trình đang chạy trên hệ thống, tuy nhiên trong Linux còn có một lệnh cho phép có thể nhìn thấy mức độ phân cấp của các tiến trình, đó là lệnh pstree.

Cú pháp lệnh: **pstree [tùy-chọn] [pid | người-dùng]**

Lệnh pstree sẽ hiển thị các tiến trình đang chạy dưới dạng cây tiến trình. Gốc của cây tiến trình thường là init. ẩ ếu đưa ra tên của một người dùng thì cây của các tiến trình do người dùng đó sở hữu sẽ được đưa ra.

pstree thường gộp các nhánh tiến trình trùng nhau vào trong dấu ngoặc vuông, ví dụ:

```
nit -+-getty
     |-getty
     |-getty
     |-getty
```

thành

```
init ---4*[getty]
```

– a chỉ ra tham số dòng lệnh. ẩ ếu dòng lệnh của một tiến trình được tráo đổi ra bên ngoài, nó được đưa vào trong dấu ngoặc đơn.

– c không thể thu gọn các cây con đồng nhất. Mặc định, các cây con sẽ được thu gọn khi có thể

– h hiển thị tiến trình hiện thời và "tổ tiên" của nó với màu sáng trắng

– H giống như tùy chọn -h, nhưng tiến trình con của tiến trình hiện thời không có màu sáng trắng

– l hiển thị dòng dài.

– n sắp xếp các tiến trình cùng một tổ tiên theo chỉ số tiến trình thay cho sắp xếp theo tên

Ví dụ:

```
# pstree
init-+-apmd
 |-atd
 |-automount
 |-crond
 |-enlightenment
 |-gdm-+-X
 | `-gdm---gnome-session
 |-gen_util_applet
 |-gmc
 |-gnome-name-serv
 |-gnome-smproxy
 |-gnomepager_appl
 |-gpm
 |-identd---identd---3*[identd]
 |-inetd
 |-kflushd
 |-klogd
 |-kpiod
 |-kswapd
 |-kupdate
 |-lockd---rpciod
 |-login---bash---mc-+-bash-+-cat
 | | |-passwd
 | | `-pstree
 | `-cons.saver
 |-lpd
 |-mdrecoveryd
 |-5*[mingetty]
 |-panel
 |-portmap
 |-rpc.statd
 |-sendmail
 |-syslogd
 `-xfs
```

2.5 Lệnh thiết đặt lại độ ưu tiên của tiến trình nice và lệnh renice

ẩ goài các lệnh xem và hủy bỏ tiến trình, trong Linux còn có hai lệnh liên quan đến độ ưu tiên của tiến trình, đó là lệnh *nice* và lệnh *renice*.

Để chạy một chương trình với độ ưu tiên định trước, hãy sử dụng lệnh *nice*.

Cú pháp lệnh: **nice [tùy-chọn] [lệnh [tham-số]...]**

Lệnh *nice* sẽ chạy một chương trình (lệnh) theo độ ưu tiên đã sắp xếp. ẩ ếu không có lệnh, mức độ ưu tiên hiện tại sẽ hiển thị. Độ ưu tiên được sắp xếp từ -20 (mức ưu tiên cao nhất) đến 19 (mức ưu tiên thấp nhất).

- _____ADJUST : tăng độ ưu tiên theo ADJUST đầu tiên
- _____- help : hiển thị trang trợ giúp và thoát

Formatted: Bullets and Numbering

Để thay đổi độ ưu tiên của một tiến trình đang chạy, hãy sử dụng lệnh **renice**.

Cú pháp lệnh: **renice <độ-ưu-tiên> [tùy-chọn]**

Lệnh *renice* sẽ thay đổi mức độ ưu tiên của một hoặc nhiều tiến trình đang chạy.

- _____g : thay đổi quyền ưu tiên theo nhóm người dùng
- _____p : thay đổi quyền ưu tiên theo chỉ số của tiến trình
- _____u : thay đổi quyền ưu tiên theo tên người dùng

Formatted: Bullets and Numbering

Ví dụ:

```
# renice +1 987 -u daemon root -p 32
```

lệnh trên sẽ thay đổi mức độ ưu tiên của tiến trình có chỉ số là 987 và 32, và tất cả các tiến trình do người dùng *daemon* và *root* sở hữu.

2.6 Lệnh fg và lệnh bg

Linux cho phép người dùng sử dụng tổ hợp phím CTRL+z để dừng một quá trình và khởi động lại quá trình đó bằng cách gõ lệnh fg. Lệnh fg (foreground) tham chiếu đến các chương trình mà màn hình cũng như bàn phím đang làm việc với chúng.

Ví dụ, người dùng đang xem trang man của lệnh sort, nhìn xuống cuối thấy có tùy chọn -b, muốn thử tùy chọn này đồng thời vẫn muốn xem trang man. Thay cho việc đánh q để thoát và sau đó chạy lại lệnh man, cho phép người dùng gõ CTRL+z để tạm dừng lệnh man và gõ lệnh thử tùy chọn -b. Sau khi thử xong, hãy gõ fg để tiếp tục xem trang man của lệnh sort. Kết quả của quá trình trên hiển thị như sau:

```
# man sort | more
SORT(1) FSF SORT(1)
NAME
sort - sort lines of text Files
SYNOPSIS
```

```
 ../src/sort [OPTION] ... [Files]...
DESCRIPTION
Write sorted concatenation of all FILE(s) to standard out-put.
 +POS1 [-POS2]
start a key at POS1,end it *before* POS2 obsoles-cent)field numbers
and character offsets are num-bered starting with zero(contrast with
the -k option)
-b ignore leading blanks in sort fields or keys
           --More--
(CTRL+z)
[1]+ Stopped    man sort | more
# ls -s | sort -b | head -4
      1 Archives/
      1 InfoWorld/
      1 Mail/
      1 News/
      1 OWL/
# fg
      man sort | more
      --More--
```

Trong phần trước, cách thức gõ phím CTRL+z để tạm dừng một quá trình đã được giới thiệu. Linux còn người dùng cách thức để chạy một chương trình dưới chế độ nền (background) - sử dụng lệnh bg - trong khi các chương trình khác đang chạy, và để chuyển một chương trình vào trong chế độ nền - dùng ký hiệu &.

ả ếu một tiến trình hoạt động mà không đưa ra thông tin nào trên màn hình và không cần nhận bất kỳ thông tin đầu vào nào, thì có thể sử dụng lệnh bg để đưa nó vào trong chế độ nền (ở chế độ này nó sẽ tiếp tục chạy cho đến khi kết thúc).

Khi chương trình cần đưa thông tin ra màn hình hoặc nhận thông tin từ bàn phím, hệ thống sẽ tự động dừng chương trình và thông báo cho người dùng. Cũng có thể sử dụng chỉ số điều khiển công việc (job control) để làm việc với chương trình nào muốn. Khi chạy một chương trình trong chế độ nền, chương trình đó được đánh số thứ tự (được bao bởi dấu ngoặc vuông []), theo sau là chỉ số của quá trình.

Sau đó có thể sử dụng lệnh fg + số thứ tự của chương trình để đưa chương trình trở lại chế độ nổi và tiếp tục chạy.

Để có một chương trình (hoặc một lệnh ống) tự động chạy trong chế độ nền, chỉ cần thêm ký hiệu '&' vào cuối lệnh.

Trong một số hệ thống, khi tiến trình nền kết thúc thì hệ thống sẽ gửi thông báo tới người dùng, nhưng trên hầu hết các hệ thống, khi quá trình trên nền hoàn thành thì hệ thống sẽ chờ cho đến khi người dùng gõ phím Enter thì mới hiển thị dấu nhắc lệnh mới kèm theo thông báo hoàn thành quá trình (thường thì một tiến trình hoàn thành sau khoảng 20 giây).

ả ếu cố để chuyển một chương trình vào chế độ nền mặc dù nó có các thông tin cần xuất hoặc nhập từ các thiết bị vào ra chuẩn thì hệ thống sẽ đưa ra thông báo lỗi dưới dạng sau:

```
Stopped (tty input/output) tên chương trình.
```

Ví dụ, lệnh sau đây thực hiện việc tìm kiếm file thu1 trong chế độ nền:

```
# find -name thu1 &
[5] 918
```

trong chế độ này, số thứ tự của chương trình là [5], chỉ số quá trình tương ứng với lệnh find là 918. Vì gõ Enter khi quá trình chưa thực hiện xong nên trên màn hình chỉ hiển thị số thứ tự của chương trình và chỉ số quá trình, nếu chờ khoảng 30 hoặc 40 giây sau rồi gõ Enter lần nữa, màn hình hiển thị thông báo hoàn thành chương trình như sau:

```
#
[5] Done   find -name thu1
#
```

Giả sử chương trình chưa hoàn thành và muốn chuyển nó lên chế độ nổi, hãy gõ lệnh sau:

```
# fg 5
find -name thu1
./thu1
```

chương trình đã hoàn thành và hiển thị thông báo rằng file thu1 nằm ở thư mục gốc.

Thông thường sẽ đưa ra một thông báo lỗi nếu người dùng cố chuyển một chương trình vào chế độ nền khi mà chương trình đó cần phải xuất hoặc nhập thông tin từ thiết bị vào ra chuẩn. Ví dụ, lệnh:

```
# vi &
[6] 920
#
```

nhấn Enter

```
#
[6] + Stopped (tty output) vi
#
```

Lệnh trên chạy chương trình vi trong chế độ nền, tuy nhiên lệnh gặp phải lỗi vì đây là chương trình đòi hỏi hiển thị các thông tin ra màn hình (output). Dòng thông báo lỗi Stopped (tty intput) vi cũng xảy ra khi chương trình vi cần nhận thông tin.

3. Quản lý trị hệ thống

3.1 Khởi động và đóng tắt hệ thống

Khi một máy PC bắt đầu khởi động, bộ vi xử lý sẽ tìm đến cuối vùng bộ nhớ hệ thống của BIOS và thực hiện các chỉ thị ở đó.

BIOS sẽ kiểm tra hệ thống, tìm và kiểm tra các thiết bị, và tìm kiếm đĩa chứa trình khởi động. Thông thường, BIOS sẽ kiểm tra ổ đĩa mềm, hoặc CDROM xem có thể khởi động từ chúng được không, rồi đến đĩa cứng. Thứ tự của việc kiểm tra các ổ đĩa phụ thuộc vào các cài đặt trong BIOS.

Khi kiểm tra ổ đĩa cứng, BIOS sẽ tìm đến MBR và nạp vào vùng nhớ hoạt động chuyển quyền điều khiển cho nó.

MBR chứa các chỉ dẫn cho biết cách nạp trình quản lý khởi động GRUB/LILO cho Linux hay ả TLDR cho Windows ả T/2000. MBR sau khi nạp trình quản lý khởi động, sẽ chuyển quyền điều khiển cho trình quản lý khởi động.

Trình quản lý khởi động sẽ cho hiện trên màn hình một danh sách các tùy chọn để người dùng xử lý xem nên khởi động hệ điều hành nào.

Các chỉ dẫn cho việc nạp hệ điều hành thích hợp được ghi rõ trong các tập tin cấu hình tương ứng với các trình quản lý khởi động.

- LILO lưu cấu hình trong tập tin /etc/lilo.conf
- GRUB lưu trong tập tin /boot/grub/grub.conf
- ả TLDR lưu trong c:\boot.ini

3.2 Tìm hiểu về trình nạp Linux

LILO là một boot manager nằm trọn gói chung với các bản phát hành Red Hat, và là boot manager mặc định cho Red Hat 7.1 trở về trước.

Thiết lập cấu hình LILO:

LILO đọc thông tin chứa trong tập tin cấu hình /etc/lilo.conf để biết xem hệ thống máy ta có những hệ điều hành nào, và các thông tin khởi động nằm ở đâu. LILO được lập cấu hình để

khởi động một đoạn thông tin trong tập tin /etc/lilo.conf cho từng hệ điều hành. Sau đây là ví dụ về tập tin /etc/lilo.conf

```
Boot=/dev/hda
Map=/boot/map
Install=/boot/boot.b
Prompt
Timeout=50
Message=/boot/message
Lba32
Default=linux
Image=/boot/vmlinuz-2.4.0-0.43.6
     Label=linux
     Initrd=/boot/initrd-2.4.0-0.43.6.img
     Read-only
     Root=/dev/hda5
Other=/dev/hda1
     Label=dos
```

Đoạn thứ nhất:

- Cho biết LILO cần xem xét vào MBR (boot=/dev/hda1)
- Kiểm tra tập tin map
- ẩ ó còn cho biết LILO có thể cài đặt một tập tin đặc biệt (/boot/boot.b) như là một sector khởi động mới
- Thời gian chờ trước khi nạp hệ điều hành mặc định (default=xxx) được khai báo thông qua dòng timeout=50 (5 giây) ? thời gian tính bằng 1/10 của giây.
- ẩ ạp thông tin trong quá trình khởi động từ tập tin /boot/message
- Dòng LBA32 cho biết cấu hình của đĩa cứng: cho biết đĩa cứng của ta hỗ trợ LBA32, thông thường dòng này có giá trị linear (ta không nên đổi lại dòng này nếu ta không hiểu rõ ổ đĩa cứng của ta, ta có thể tìm hiểu đĩa cứng của ta có hỗ trợ LBA32 hay không bằng cách xem trong BIOS)

Đoạn thứ hai:

- Cung cấp thông tin khởi động cho hệ điều hành linux
- Dòng image báo cho LILO biết vị trí của kernel Linux
- Dòng label hiện diện ở cả 2 đoạn cho biết tên của hệ điều hành nào sẽ xuất hiện tại trình đơn khởi động của LILO.

- Dòng root xác định vị trí root file system của Linux

Đoạn thứ ba:

- Dòng other cho biết partition của một hệ điều hành nữa đang ở hda1 của ổ đĩa cứng.

3.3 Tìm hiểu GRUB, trình nạp Linux.

Định nghĩa: GRUB cũng chỉ là một trình quản lý khởi động tương tự LILO.

Tập tin cấu hình GRUB: ẩ hư trên, ta thấy thông thường sẽ có 3 đoạn cơ bản.

Đoạn thứ nhất: mô tả các chỉ thị tổng quát như :

- Hệ điều hành mặc định (default)
- Thời gian chờ đợi người dùng nhập dữ liệu trước khi thực hiện lệnh mặc định (timeout=10), tính bằng giây.
- Ta cũng có thể chọn màu để hiển thị trình đơn (color green/black light-gray/blue)

Đoạn thứ hai: cho biết các thông số để khởi động hệ Linux:

- Tiêu đề trên trình đơn là Red Hat Linux (title)
- Hệ điều hành này sẽ khởi động từ partition đầu tiên của ổ đĩa thứ nhất ? root (hda0,0:ổ đĩa thứ nhất, partition thứ nhất). Và cần phải mount partition này trước.
- Tập tin vmlinuz đang được chứa trong thư mục root và filesystem root đang nằm trên partition thứ năm của đĩa cứng thứ nhất (/dev/hdc5)
- Dòng lệnh boot nhắc phải nạp ngay hệ điều hành đã được khai báo ở trên.

Đoạn thứ ba: cho biết các thông số về hệ điều hành thứ hai đang được cài đặt trong hệ thống.

- Tiêu đề là Windows
- Hệ điều hành đang chiếm partition thứ nhất của ổ đĩa thứ hai (hda1,0). Có điều với lệnh rootnoverify, GRUB không cần chú ý kiểm tra xem partition này có được mount hay không.
- Câu lệnh chainloader + 1 đã sử dụng +1 làm tên tập tin cần khởi động như một mắc xích trong tiến trình: +1 có nghĩa là sector thứ nhất của partition đang xét ta có thể dùng lệnh man grub.conf để tìm hiểu thêm về tập tin cấu hình này.

3.4 Quá trình khởi động

Sau khi ta bật máy, máy sẽ nạp boot loader (lilo or grub), boot loader nạp file boot image để khởi tạo hệ điều hành, sau đó hệ điều hành kiểm tra các thiết bị phần cứng, hệ điều hành bắt đầu kiểm tra partition, mount các file system cần thiết cho hệ thống, tiếp theo nó đọc tập tin /etc/inittab để chọn default runlevel, khởi tạo các deamon, cuối cùng yêu cầu người dùng logon

vào trước khi sử dụng hệ thống, sau khi log on bằng username và password, hệ thống sẽ chạy chương trình shell (hoặc chạy X Windows) để giao tiếp với người dùng.

4. Quản trị người dùng

Trong môi trường nhiều người cùng làm việc trên hệ thống, cùng sử dụng, chia sẻ các tài nguyên như bộ nhớ, đĩa cứng, máy in và các thiết bị khác. Chính sách quản lý người dùng tốt sẽ là chìa khóa cho hoạt động hiệu quả của hệ thống.

4.1 Superuser (root)

Các hệ thống máy chủ đều có account quản trị, ví như ổ T có account administrator, ổ ovell có admin. Đây là account có quyền cao nhất, dùng cho người quản trị quản lý, giám sát hệ thống. Trong quá trình cài đặt Linux chúng ta khởi tạo người sử dụng root cho hệ thống. Đây là superuser, tức là người sử dụng đặc biệt có quyền không giới hạn. Sử dụng quyền root chúng ta thấy rất thoải mái vì chúng ta có thể làm được thao tác mà không phải lo lắng gì đến xét quyền thâm nhập này hay khác.

Tuy nhiên, khi hệ thống bị sự cố do một lỗi lầm nào đó, chúng ta mới thấy sự nguy hiểm khi làm việc như root. Do vậy chúng ta chỉ sử dụng account này vào các mục đích cấu hình, bảo trì hệ thống chứ không nên sử dụng vào mục đích hằng ngày.

Ta cần tạo các tài khoản (account) cho người sử dụng thường sớm nhất có thể được (đầu tiên là cho bản thân ta). Với những server quan trọng và có nhiều dịch vụ khác nhau, thậm chí ta có thể tạo ra các superuser thích hợp cho từng dịch vụ để tránh dùng root cho các công tác này. Ví dụ như superuser cho công tác backup chỉ cần chức năng đọc (read-only) mà không cần chức năng ghi.

Account root

Tài khoản này có quyền hạn rất lớn nên nó là mục tiêu mà các kẻ xấu muốn chiếm đoạt. Chúng ta sử dụng nó phải cẩn thận, không sử dụng bừa bãi trên qua telnet hay kết nối từ xa mà không có công cụ kết nối an toàn.

Trong Linux, chúng ta có thể tạo các user có tên khác có quyền của root, bằng cách tạo user có UserID bằng 0.

Cần phân biệt ta đang login như root hay người sử dụng thường thông qua dấu nhắc của shell.

```
login: natan
Password:
Last login: Wed Mar 13 19:00:42 2002 from 172.29.8.3
[natan@NetGroup natan]$ su -
```

```
Password:
[root@NetGroup /root]#
```

Dòng thứ tư với dấu $ cho thấy ta đang kết nối như một người sử dụng thường (tnminh).

Dòng cuối cùng với dấu # cho thấy ta đang thực hiện các lệnh với root.

 # su user_name (trong ubuntu sử dụng lệnh #sudo su user_name)

Cho phép ta thay đổi login dưới một user khác (user_name) mà không phải logout rồi login lại.

4.2 Tài khoản người dùng

Mọi người muốn đăng nhập và sử dụng hệ thống Linux đều cần có 1 account. Việc tạo ra và quản lý account người dùng là vấn đề quan trọng mà người quản trị phải thực hiện. Trừ account root, các account khác do người quản trị tạo ra.

Mỗi tài khoản người dùng phải có một tên sử dụng (username), một mật khẩu (password) riêng để người quản trị dễ dàng quản lý hoạt động của người dùng cũng như tăng cường tính an toàn cho hệ thống. Tập tin /etc/passwd là tập tin chứa các thông tin về tài khoản người dùng của hệ thống.

Tập tin /etc/passwd

Tập tin /etc/passwd đóng vai trò sống còn đối với một hệ thống Unix/Linux. Mọi người đều có thể đọc được tập tin này nhưng chỉ có root mới có quyền thay đổi nó. Tập tin /etc/passwd được lưu dưới dạng text như đại đa số các tập tin cấu hình của Unix/Linux.

```
[natan@NetGroup natan]$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
ftp:x:14:50:FTP User:/var/ftp:
nobody:x:99:99:Nobody:/:
nscd:x:28:28:NSCD Daemon:/:/bin/false
mailnull:x:47:47::/var/spool/mqueue:/dev/null
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/bin/false
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
nthung:x:525:526:nguyen tien hung:/home/nthung:/bin/bash
natan:x:526:527::/home/natan:/bin/bash
```

Mỗi user được lưu trong một dòng gồm 7 cột:

- Cột 1 : tên người sử dụng.
- Cột 2 : mã liên quan đến passwd cho Unix chuẩn và ?x? đối với Linux. Linux lưu mã này trong một tập tin khác /etc/shadow mà chỉ có root mới có quyền đọc.
- Cột 3:4 : user ID:group ID.
- Cột 5 : tên đầy đủ của người sử dụng. Một số phần mềm phá password sử dụng dữ liệu của cột này để thử đoán password.
- Cột 6 : thư mục cá nhân.
- Cột 7 : chương trình sẽ chạy đầu tiên sau khi login (thường là shell) cho user.

Tập tin mở đầu bởi superuser root. Chú ý là tất cả những user có user ID = 0 đều là root.

Tiếp theo là các user hệ thống. Đây là các user không có thật và không thể login vào hệ thống. Cuối cùng là các user bình thường.

Tên người dùng và định danh người dùng (Username và User ID)

Tên người dùng là chuỗi ký tự xác định duy nhất một người dùng. ẩ gười dùng tên này khi đăng nhập cũng như truy xuất tài nguyên.

Trong linux tên phân biệt chữ hoa, thường. Thông thường tên người dùng thường sử dụng chữ thường.

Để quản lý người dùng linux sử dụng khái niệm định danh người dùng (user ID). Mỗi người dùng mang một con số định danh cho mình.

Linux sử dụng số định danh để kiểm soát hoạt động của người dùng. Theo qui định chung các người dùng có định danh là 0 là người dùng quản trị (root). Các số định danh từ 1- 99 sử dụng cho các tài khoản hệ thống, định danh của người dùng bình thường sử dụng giá trị bắt đầu từ 100.

Mật khẩu (Password)

Mỗi người dùng phải có một mật khẩu riêng để sử dụng tài khoản người dùng của mình. Mọi người đều có quyền đổi mật khẩu của chính mình. ẩ gười quản trị thì có thể đổi mật khẩu của những người khác.

Unix/Linux truyền thống lưu các thông tin liên quan tới mật khẩu để đăng nhập (login) ở trong /etc/passwd. Tuy nhiên, do đây là tập tin phải đọc được bởi tất cả mọi người do một số yêu cầu cho hoạt động bình thường của hệ thống (như chuyển User ID thành tên khi hiển thị trong lệnh ls chẳng hạn) và nhìn chung các user đặt mật khẩu ?yếu?, do đó, hầu hết các phiên bản Unix mới đều lưu mật khẩu trong một tập tin khác /etc/shadow và chỉ có root được quyền đọc tập tin này.

Chú ý: Theo cách xây dựng mã hóa mật khẩu, chỉ có 2 cách phá mật khẩu là vét cạn (brute force) và đoán. Phương pháp vét cạn, theo tính toán chặt chẽ, là không thể thực hiện nổi vì đòi hỏi thời gian tính toán quá lớn, còn đoán thì chỉ tìm ra những mật khẩu ngắn, hoặc ?yếu?, ví dụ như những từ tìm thấy trong từ điển như god, darling ?

Group ID

Định danh của nhóm mà user này là một thành viên của nhóm.

Comment

Dòng chú thích về user này.

Home Directory

Khi người dùng login vào hệ thống được đặt làm việc tại thư mục cá nhân của mình. Thường thì mỗi người có một thư mục cá nhân riêng, người dùng có toàn quyền trên nó, nó dùng chứa dữ liệu cá nhân và các thông tin hệ thống cho hoạt động của người dùng như biến môi trường, script khởi động, profile khi sử dụng X window ?

Thư mục mặc nhiên sử dụng cho các thư mục cá nhân của người dùng bình thường là /home; cho root là /root. Tuy nhiên chúng ta cũng có thể đặt vào vị trí khác.

4.3 Thêm người dùng với lệnh useradd

ẩ gười quản trị hệ thống sử dụng lệnh useradd (trong một số phiên bản là adduser) để tạo một người dùng mới hoặc cập nhật ngầm định các thông tin về người dùng.

Cú pháp lệnh: **useradd [tùy-chọn] <tên-người-dùng>**
useradd -D [tùy-chọn]

ẩ ếu không có tùy chọn -D, lệnh *useradd* sẽ tạo một tài khoản người dùng mới sử dụng các giá trị được chỉ ra trên dòng lệnh và các giá trị mặc định của hệ thống. Tài khoản người dùng mới sẽ được nhập vào trong các file hệ thống, thư mục cá nhân sẽ được tạo, hay các file khởi tạo được sao chép, điều này tùy thuộc vào các tùy chọn được đưa ra.

Các tùy chọn như sau:

- c, comment soạn thảo trường thông tin về người dùng.
- d, home_dir tạo thư mục đăng nhập cho người dùng.
- e, expire_date thiết đặt thời gian (YYYY-MM-DD) tài khoản người dùng sẽ bị hủy bỏ.
- f, inactive_days tùy chọn này xác định số ngày trước khi mật khẩu của người dùng hết hiệu lực khi tài khoản bị hủy bỏ. ẩ ếu =0 thì hủy bỏ tài khoản người dùng ngay sau khi mật khẩu hết hiệu lực, =-1 thì ngược lại (mặc định là -1).

– g, initial_group tùy chọn này xác định tên hoặc số khởi tạo đăng nhập nhóm người dùng. Tên nhóm phải tồn tại, và số của nhóm phải tham chiếu đến một nhóm đã tồn tại. Số nhóm ngầm định là 1.

– G, group danh sách các nhóm phụ mà người dùng cũng là thành viên thuộc các nhóm đó. Mỗi nhóm sẽ được ngăn cách với nhóm khác bởi dấu ',', mặc định người dùng sẽ thuộc vào nhóm khởi tạo.

– m với tùy chọn này, thư mục cá nhân của người dùng sẽ được tạo nếu nó chưa tồn tại.

– M không tạo thư mục người dùng.

– n ngầm định khi thêm người dùng, một nhóm cùng tên với người dùng sẽ được tạo. Tùy chọn này sẽ loại bỏ sự ngầm định trên.

– p, passwd tạo mật khẩu đăng nhập cho người dùng.

– s, shell thiết lập shell đăng nhập cho người dùng.

– u, uid thiết đặt chỉ số người dùng, giá trị này phải là duy nhất.

4.4 Thay đổi thông tin của user

Cú pháp lệnh: **usermod [tùy-chọn] <tên-đăng-nhập>**

Lệnh **usermod** sửa đổi các file tài khoản hệ thống theo các thuộc tính được xác định trên dòng lệnh. Các tùy chọn của lệnh:

– c, comment thay đổi thông tin cá nhân tài khoản người dùng.

– d, home_dir thay đổi thư mục cá nhân tài khoản người dùng.

– e, expire_date thay đổi thời điểm hết hạn tài khoản người dùng (YYYY-MM-DD).

– f, inactive_days thiết đặt số ngày hết hiệu lực của mật khẩu trước khi tài khoản người dùng hết hạn sử dụng.

– g, initial_group tùy chọn này thay đổi tên hoặc số khởi tạo đăng nhập nhóm người dùng. Tên nhóm phải tồn tại, và số của nhóm phải tham chiếu đến một nhóm đã tồn tại. Số nhóm ngầm định là 1.

– G, group thay đổi danh sách các nhóm phụ mà người dùng cũng là thành viên thuộc các nhóm đó. Mỗi nhóm sẽ được ngăn cách với nhóm khác bởi dấu ',' mặc định người dùng sẽ thuộc vào nhóm khởi tạo.

– l, login_namethay đổi tên đăng nhập của người dùng. Trong một số trường hợp, tên thư mục cá nhân của người dùng có thể sẽ thay đổi để tham chiếu đến tên đăng nhập mới.

– p, passwd thay đổi mật khẩu đăng nhập của tài khoản người dùng.

– s, shell thay đổi shell đăng nhập.

– u, uid thay đổi chỉ số người dùng.

Lệnh usermod không cho phép thay đổi tên của người dùng đang đăng nhập. Phải đảm bảo rằng người dùng đó không thực hiện bất kỳ tiến trình nào trong khi lệnh usermod đang thực hiện thay đổi các thuộc tính của người dùng đó.

Ví dụ : muốn thay đổi tên người dùng **new** thành tên mới là **newuser**, hãy gõ lệnh sau:

```
# usermod -l new newuser
```

4.5 Hủy user

Lệnh hay được dùng để xóa bỏ một tài khoản người dùng là lệnh **userdel.**

Cú pháp: **userdel [-r] <tên-người-dùng>**

Lệnh này sẽ thay đổi nội dung của các file tài khoản hệ thống bằng cách xóa bỏ các thông tin về người dùng được đưa ra trên dòng lệnh. ẩ gười dùng này phải thực sự tồn tại. Tuỳ chọn -r có ý nghĩa:

– r các file tồn tại trong thư mục cá nhân của người dùng cũng như các file nằm trong các thư mục khác có liên quan đến người dùng sẽ bị xóa bỏ cùng lúc với thư mục người dùng.

Lệnh userdel sẽ không cho phép xóa bỏ người dùng khi họ đang đăng nhập vào hệ thống. Phải hủy bỏ mọi tiến trình có liên quan đến người dùng trước khi xoá bỏ người dùng đó.

ẩ goài ra cũng có thể xóa bỏ tài khoản của một người dùng bằng cách hiệu chỉnh lại file /etc/passwd.

4.6 Tạo nhóm người dùng groupadd

Việc tạo nhóm xuất phát từ việc gom các người dùng có chung một số quyền hạn trên tài nguyên. Mỗi nhóm có một tên và một định danh nhóm, Một nhóm có thể chứa nhiều người dùng và người dùng có thể thuộc nhiều nhóm. Tuy nhiên tại một điểm một người chỉ thuộc một nhóm mà thôi.

Thông tin của nhóm lưu tại tập tin /etc/group. Mỗi dòng định nghĩa một nhóm, các trường trên dòng cách nhau bằng dấu **:**

ẩ ội dung của một dòng: tên-nhóm : password-của-nhóm: định-danh-nhóm:các-user-thuộc-nhóm

Chúng ta có thể thêm trực tiếp vào file /etc/group hoặc dùng lệnh groupadd:

```
#groupadd tên-nhóm
```

Thêm user vào group

Chúng ta có thể sửa từ tập tin /etc/group, các tên người dùng cách nhau bằng dấu ;. Một cách khác là cho từng uset vào nhóm bằng lệnh: # usermod ?g tên-nhóm tên-user hay sửa tập tin /etc/passwd cho từng user, trong đó thay lại định danh nhóm trong dòng khai báo người dùng.

Hủy group

Xóa trong tập tin /etc/group hay dùng lệnh : #groupdel tên-nhóm

Sửa đổi các thuộc tính của một nhóm người dùng

Trong một số trường hợp cần phải thay đổi một số thông tin về nhóm người dùng bằng lệnh groupmod với cú pháp như sau: #roupmod [tùy-chọn] <tên-nhóm>

Thông tin về các nhóm xác định qua tham số tên-nhóm được điều chỉnh.

Các tùy chọn của lệnh:

- g, gid thay đổi giá trị chỉ số của nhóm người dùng.
- n, group_name thay đổi tên nhóm người dùng.

4.7 Xác định người dùng đang đăng nhập (lệnh who)

Lệnh who là một lệnh đơn giản, cho biết được hiện tại có những ai đang đăng nhập trên hệ thống với cú pháp như sau: **#who [tùy-chọn]**

Các tuỳ chọn là:

- H, --heading : hiển thị tiêu đề của các cột trong nội dung lệnh.
- m : hiển thị tên máy và tên người dùng với thiết bị vào chuẩn.
- q, --count : hiển thị tên các người dùng đăng nhập và số người dùng đăng nhập.

Formatted: Bullets and Numbering

Ví dụ:

```
# who
root tty1 Nov 15 03:54
lan pts/0 Nov 15 06:07
#
```

Lệnh who hiển thị ba cột thông tin cho từng người dùng trên hệ thống. Cột đầu là tên của người dùng, cột thứ hai là tên thiết bị đầu cuối mà người dùng đó đang sử dụng, cột thứ ba hiển thị ngày giờ người dùng đăng nhập.

ẩ goài who, có thể sử dụng thêm lệnh users để xác định được những người đăng nhập trên hệ thống.

```
# users
lan root
#
```

Trong trường hợp người dùng không nhớ nổi tên đăng nhập trong một phiên làm việc (điều này nghe có vẻ như hơi vô lý nhưng là tình huống đôi lúc gặp phải), hãy sử dụng lệnh whoami và who am i. Sử dụng lệnh: **# whoami** hoặc **# who am i**

```
# whoami
lan
# who am i
may9!lan pts/0 Nov 15 06:07
```

Lệnh who am i sẽ hiện kết quả đầy đủ hơn với tên máy đăng nhập, tên người dùng đang đăng nhập, tên thiết bị và ngày giờ đăng nhập.

4.8 Để xác định thông tin người dùng với lệnh id

Cú pháp lệnh: **id [tùy-chọn] [người-dùng]**

Lệnh này sẽ đưa ra thông tin về người dùng được xác định trên dòng lệnh hoặc thông tin về người dùng hiện thời.

Các tuỳ chọn là:

- _____g, --group : chỉ hiển thị chỉ số nhóm người dùng.
- _____u, --user : chỉ hiển thị chỉ số của người dùng.
- _____- help : hiển thị trang trợ giúp và thoát.

Formatted: Bullets and Numbering

```
# id
uid=506(lan) gid=503(lan) groups=503(lan)
# id -g
503
# id -u
506
# id root
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),
3(sys),4(adm),6(disk),10(wheel)
#
```

4.9 Xác định các tiến trình đang được tiến hành (lệnh w)

Lệnh w cho phép xác định được thông tin về các quá trình đang được thực hiện trên hệ thống và những người dùng tiến hành quá trình đó.

Cú pháp lệnh: #**w [người-dùng]**

Lệnh w đưa ra thông tin về người dùng hiện thời trên hệ thống và quá trình họ đang thực hiện. ả ếu chỉ ra người dùng trong lệnh thì chỉ hiện ra các quá trình liên quan đến người dùng đó.

```
# w
```

```
root tty2 - 2:14pm 13:03 9.30s 9.10s /usr/bin/mc -P
lan pts/1 192.168.2.213 3:20pm 0.00s 0.69s 0.10s w
root pts/2 :0 3:33pm 9:32 0.41s 0.29s /usr/bin/mc –P
```

5. Quản trị tài nguyên

5.1 Quản lý tài nguyên với lệnh quota

Một công cụ tốt nhất để quản lý tài nguyên đĩa là quota. Quota được dùng để hiển thị việc sử dụng và giới hạn đĩa của người dùng. Khi được gọi, quota sẽ quét tập tin /etc/fstab và kiểm tra những file system trong tập tin này. Thông thường, quota dùng để giới hạn dung lượng đĩa cứng mà ta cấp cho người dùng.

Giới hạn về cứng và mềm

Để giúp cho việc giới hạn có hiệu quả, quota chia làm 2 loại giới hạn giới hạn cứng và giới hạn mềm.

- Giới hạn cứng: không cho phép vượt quá dung lượng đĩa cho phép. ả ếu user cố tình lưu những thông tin vào thì những thông tin trước đó có thể bị xóa và đẩy lên dần. Việc giới hạn này thật mạnh mẽ nhưng cần thiết đối với một số user.
- Giới hạn mềm: cho phép user vượt quá dung lượng cho phép, nhưng sẽ nhận một lời cảnh báo trước. Một ý kiến hay, ta cấu hình giới hạn mềm nhỏ hơn giới hạn cứng, và cấu hình khi user vượt quá dung lượng cho phép hệ thống sẽ gửi một lời cảnh báo trước khi cho phép user lưu dữ liệu.

Khi nào sử dụng quota?

Không phải ta dùng quoata cho tất cả những filesystem. Chỉ có những filesystem nào cần thiết chúng ta mới dùng quota.

Và khi đó, chúng ta vào file /etc/fstab cấu hình như sau:

```
/dev/hda3 /usr rw,usrquota,grpquota 1 3
```

Thiết lập quota

ả gười quản trị hệ thống sẽ thiết lập quota cho user trong file có tên quota.user nằm trong filesystem mà chúng ta muốn cấu hình quota. Tương tự, chúng ta cũng sẽ thiết lập quota cho nhóm trong file quota.group. ả hững tập tin này ta chúng ta sẽ tạo ra.

ả hững lệnh sau đây hướng dẫn ta cách thiết lập quota cho filesystem /usr.

```
cd /usr
touch aquota.user
chmod 600 aquota.user
touch aquota.group
```

```
chmod 600 aquota.group
```

Ta sẽ dùng lệnh edquota để thiết lập quota. Lệnh này chỉ được dùng bởi user root. Với lệnh này chúng ta có thể giới hạn dung lượng cho một hay nhiều user hoặc group cùng lúc. Ví dụ như sau:

```
edquota hv1 hv2
```

Ta có thể điều khiển lệnh quota một cách hiệu quả với những tùy chọn sau:

- g chỉnh sửa quota của group
- p sao chép quota của một user cho một user khác
- u chỉnh sửa quota cho user (mặc định của lệnh)
- t chỉnh sửa thời gian của giới hạn mềm.

Sau khi thiết lập quota, ta phải bật quota lên bằng lệnh: # **quotaon /dev/hda3**

Để bật quota kiểm tra tất cả những file system dùng lệnh: # quotaon ?a

Lệnh quotaoff có tính năng ngược lại, tắt quota trên filesystem.

Lệnh quota

Cú pháp của lệnh: **quota [tùy chọn] [user] [group]**

ẩ hững tùy chọn của lệnh quota.

- g hiển thị quota của group mà user này là một thành viên
- q chỉ hiển thị những filesystem có quota
- u hiển thị quota của user

Lệnh quotacheck

Ta có thể sử dụng lệnh quotacheck tại bất cứ lúc nào để kiểm tra việc sử dụng đĩa hiện hành.

5.2 Lệnh quản lý đĩa với lệnh du và df

Xem dung lượng đĩa đó sử dụng với lệnh du

Linux cho phép người dùng xem thông tin về dung lượng đĩa đó được sử dụng bằng lệnh du với cú pháp : # **du [tùy-chọn]... [file]...**

Lệnh du liệt kê kích thước (tính theo kilobytes) của mỗi file thuộc vào hệ thống file cú chứa file được chỉ trong lệnh.

Các tuỳ chọn là:

- a liệt kê kích thước của tất cả các file có trong hệ thống file lưu trữ file.
- b, --bytes hiển thị kích thước theo byte.
- c, --total hiển thị cả tổng dung lượng được sử dụng trong hệ thống file.

– D, --dereference-args liên kết đến nếu chúng nằm trên các thư mục khác.

– h, --human-readable hiển thị kích thước các file kèm theo đơn vị tính (ví dụ: 1K, 234M, 2G...).

– k, --kilobytes hiển thị kích thước tính theo kilobytes.

– L, --dereference tính cả kích thước của các file được liên kết tới.

– l, --count-links tính kích thước các file nhiều lần nếu được liên kết cứng.

– m, --megabytes tính kích thước theo megabytes.

– S, --separate-dirs không hiển thị kích thước của thư mục con.

– s đưa ra kích thước của hệ thống file có lưu trữ file.

– x, --one-file system bỏ qua các thư mục trên các hệ thống file khác.

– - help hiển thị trang trợ giúp và thoát.

Chú ý: lệnh *du* không cho phép có nhiều tùy chọn trên cùng một dòng lệnh. Ví dụ: lệnh sau cho biết kích cỡ của các file trong thư mục */usr/doc/test:*

```
# du /usr/doc/test
28    ./TODO/1.0_to_1.5
24    ./TODO/lib++
16    ./TODO/unreleased
12    ./TODO/unstable
144   ./TODO
44    ./code
160   ./languages
56    ./licences
532   .
```

ả hìn vào màn hình có thể biết được kích thước của file./TODO/1.0_to_1.5 là 28 KB, file./TODO/lib++ là 24 KB,..., và kích thước của thư mục hiện thời là 532 KB.

Kiểm tra dung lượng đĩa trống với lệnh df

Cú pháp lệnh: # df [tùy-chọn]... [file]...

Lệnh này hiển thị dung lượng đĩa còn trống trên hệ thống file chứa file. ả ếu không có tham số file thì lệnh này hiển thị dung lượng đĩa còn trống trên tất cả các hệ thống file được kết nối.

Các tùy chọn:

– a, --all bao gồm cả các file hệ thống có dung lượng là 0 block.

– - block-size thiết lập lại độ lớn của khối là cỡ byte.

– k, --kilobytes hiển thị dung lượng tính theo kilobytes.

– l, --local giới hạn danh sách các file cục bộ trong hệ thống.

– m, --megabytes hiển thị dung lượng tính theo megabytes.

– t, --type=kiểu giới hạn danh sách các file hệ thống thuộc kiểu.

– T, --print-type hiển thị các kiểu của file hệ thống.

– - help đưa ra trang trợ giúp và thoát.

Để chỉ ra được dung lượng đĩa còn trống trong Linux không phải là điều dễ làm. ả gười dùng có thể sử dụng lệnh df để làm được điều này, tuy nhiên kết quả của lệnh này chỉ cho biết dung lượng đĩa đã được sử dụng và dung lượng đĩa còn trống của từng hệ thống file. ả ếu muốn biết tổng dung lượng đĩa còn trống là bao nhiêu, sẽ phải cộng dồn dung lượng đĩa còn trống của từng hệ thống file.

Ví dụ, lệnh: # df /mnt/floppy

sẽ cho kết quả như sau trên màn hình (dòng đầu tiên là tên cột):

```
Filesystem      1k-blocks Used    Available  Use% Mounted on
/dev/hda2       2174808   1378228     686104       67%   /
none            0           0         0             -     /proc
none            0           0         0             -     /dev/pts
automount(pid411)0          0         0             -     /misc
/dev/fd0        1423      249         1174       18%      /mnt/floppy
```

có thể xác định được, đĩa mềm đã được sử dụng 18%, như vậy là còn 82% (tức là còn 1174 KB) dung lượng đĩa chưa được sử dụng.

– Cột Filesystem chứa tên của thiết bị đĩa, cột 1k-blocks chứa dung lượng của thiết bị.

– Cột Used chứa dung lượng đĩa đã được sử dụng.

– Cột Available chứa dung lượng đĩa còn trống,

– Cột Use% chứa % dung lượng đĩa đã sử dụng

– Cột Mounted on chứa điểm kết gắn của thiết bị.

Cách nhanh nhất để biết được dung lượng đĩa còn trống bao nhiêu là phải xác định được tên của một thư mục bất kỳ có trong đĩa đó, sử dụng lệnh df với tham số file là tên của thư mục. Sau đó đọc nội dung cột Available trên màn hình hiển thị để biết dung lượng đĩa còn trống. Chẳng hạn, trên đĩa cứng đang sử dụng có thư mục /etc, khi đó gõ lệnh:

```
# df /etc
```

kết quả hiển thị lên màn hình như sau cho biết đĩa còn có 466252 khối rỗi :

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda1	1984240	1417192	466252	75%	/

6 Truyền thông trong Linux

6.1. Lệnh đặt tên máy

Lệnh **#Hostname name,** nếu ta muốn đặt tên đầy đủ(full domain name).

```
Hostname name.domainname
```

Thông tin về tên máy nằm trong tập tin /etc/hosts bao gồm các thông tin sau:

```
Địa chỉ ip tên máy
may12
```

ẩ goài ra ta còn xem file mô tả thông tin về đường mạng /etc/networks

```
loopback 127.0.0.0
merlin-net 147.154.12.0
BNR 47.0.0.0
```

6.2. Lệnh ifconfig

Khi sử dụng lệnh *ifconfig* kết quả thu được là:

```
eth0 Link encap:Ethernet  HWaddr 00:02:55:07:63:07
     inet addr:203.113.130.201
      Bcast:203.113.130.223  Mask:255.255.255.224
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:3912830 errors:84463 dropped:0 overruns:0 frame:0
     TX packets:2402090 errors:0 dropped:0 overruns:0 carrier:0
     collisions:84463 txqueuelen:100
     RX bytes:2767096664 (2638.9 Mb)  TX bytes:1265930467 (1207.2 Mb)
     Interrupt:29

eth1 Link encap:Ethernet  HWaddr 00:05:1C:98:05:B1
     inet addr:10.10.0.10  Bcast:10.10.255.255  Mask:255.255.0.0
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:15389731 errors:0 dropped:0 overruns:0 frame:0
     TX packets:7768909 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:100
     RX bytes:2578998337 (2459.5 Mb)  TX bytes:1471928637 (1403.7 Mb)
lo   Link encap:Local Loopback
     inet addr:127.0.0.1  Mask:255.0.0.0
     UP LOOPBACK RUNNING  MTU:16436  Metric:1
     RX packets:45868 errors:0 dropped:0 overruns:0 frame:0
     TX packets:45868 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:0
     RX bytes:5338927 (5.0 Mb)  TX bytes:5338927 (5.0 Mb)
```

Trong trường hợp này ta thấy máy hiện tại có 2 card mạng và được gán các địa chỉ tương ứng như trên.

Muốn chỉ xem các thông tin về một card mạng nào đó thôi ta dùng lệnh:

```
# ifconfig eth0
```

Muốn kích hoạt một card mạng ta dùng lệnh

```
# ifconfig eth0 up
```

Muốn tắt một card mạng ta dùng lệnh

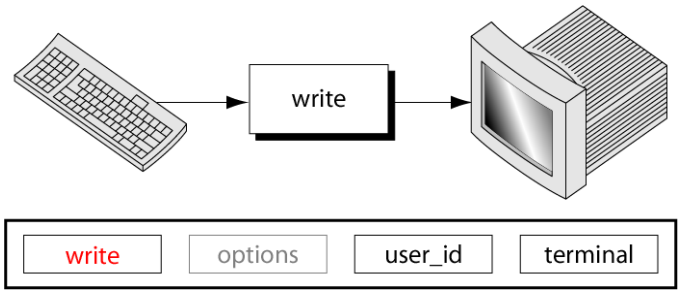
```
# ifconfig eth0 down
```

Muốn đặt lại địa chỉ cho một card mạng ta dùng lệnh:

```
# ifconfig eth0 172.16.9.15 netmask 255.255.0.0
```

6.3 Lệnh write

Lệnh write được dùng để trao đổi giữa những người hiện đang cùng làm việc trong hệ thống.



Thông thường, một người dùng muốn liên hệ với người dùng khác, cần sử dụng lệnh who: **$who** hiện thông tin như sau:

```
user1 tty17      Oct 15 10:20
user2 tty43      Oct 15 8:25
user4 tty52      Oct 15 12:20
```

trong đó có tên người dùng, số hiệu terminal, ngày giờ vào hệ thống.

Sau đó sử dụng lệnh write để chuyển thông báo cho nhau.

```
$write <tên người dùng>    [<tên trạm cuối>]
```

cần gửi thông báo đến người dùng user1 có tên user2 sẽ gõ:

```
$write user2 tty43
```

ả ếu người dùng user2 hiện không làm việc thì trên màn hình người dùng user1 sẽ hiện ra: "user2 is not logged in" và hiện lại dấu mời shell.

ả ếu người dùng user2 đang làm việc, máy người dùng user2 sẽ phát ra tiếng chuông và trên màn hình hiện ra:

```
Message from user1 on tty17 at <giờ, phút>
```

Cùng lúc đó, tại máy của user1 màn hình trắng để hiện những thông tin gửi tới người dùng user2. ả gười gửi gõ thông báo của mình theo quy tắc:

Kết thúc một dòng bằng cụm -o,

Kết thúc dòng cuối cùng (hết thông báo) bằng cụm -oo.

Để kết thúc kết nối với người dùng user2, người dùng user1 gõ ctrl-d.

Để từ chối mọi việc nhận thông báo từ người khác, sử dụng lệnh không nhận thông báo:

```
$mesg n     (n - no)
```

Một người khác gửi thông báo đến người này sẽ nhận được việc truy nhập không cho phép permission denied.

Để tiếp tục cho phép người khác gửi thông báo đến, sử dụng lệnh:

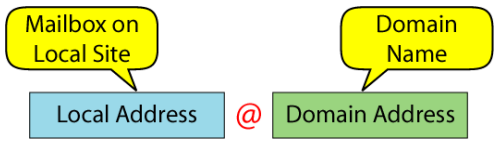
```
$mesg y     (y - yes)
```

6.4 Lệnh mail

Lệnh mail cho phép gửi thư điện tử giữa các người dùng, song hoạt động theo chế độ off-line (gián tiếp). Khi dùng lệnh write để truyền thông cho nhau thì đòi hỏi hai người gửi và nhận đồng thời đang làm việc và cùng chấp nhận cuộc trao đổi đó.

Cách thức sử dụng mail là khác hẳn: một trong hai người gửi hoặc nhận có thể không đăng nhập vào hệ thống. Để đảm bảo cách thức truyền thông gián tiếp (còn gọi là off-line) như vậy, hệ thống tạo ra cho mỗi người dùng một hộp thư riêng.

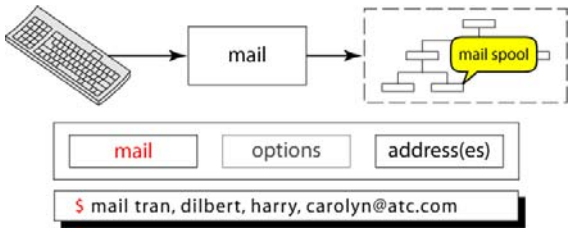
Khi một người dùng lệnh mail gửi thư đến một người khác thì thư được tự động cho vào hộp thư của người nhận và người nhận sau đó cũng dùng lệnh mail để xem trong hộp thư có thư mới hay không. Không những thế mail còn cho phép sử dụng trên mạng internet (địa chỉ mail thường dưới dạng *tên-login@máy.mạng.lĩnh-vực.quốc-gia*).



Lệnh mail chỉ yêu cầu người gửi (hoặc người nhận) login trong hệ thống. Việc nhận và gửi thư được tiến hành từ một người dùng. Thư gửi đi cho người dùng khác, được lưu tại hộp thư của hệ thống.

Tại thời điểm login hệ thống, người dùng có thể thấy được có thư mới khi trên màn hình xuất hiện dòng thông báo "you have mail".

Lệnh mail trong Uả IX gồm 2 chức năng: gửi thư và quản lý thư. Tương ứng, có hai chế độ làm việc với lệnh mail: mode lệnh (command mode) quản trị thư và mode soạn (compose mode) cho phép tạo thư.



Mode soạn

Mode soạn làm việc trực tiếp với một thư và gửi ngay cho người khác. Mode soạn thực chất là sử dụng lệnh mail có tham số:

```
$mail tên_người_nhận>        Ví dụ, $mail user2
```

Lệnh này cho phép soạn và gửi thư cho người nhận có tên được chỉ.

Sau khi gõ lệnh, màn hình bị xóa và con trỏ soạn thảo nhấp nháy ở góc trên, trái để người dùng gõ nội dung thư.

Để kết thúc soạn thư, hãy gõ ctrl-d, màn hình của mail biến mất và dấu mời của shell lại xuất hiện.

Chú ý: Dạng sau đây được dùng để gửi thư đã soạn trong nội dung một file nào đó (chú ý dấu "<" chỉ dẫn thiết bị vào chuẩn là nội dung file thay vì cho bàn phím):

```
$mail tên_người_nhận < tên_file_nội_dung_thư
```

Ví dụ

```
$ mail user2 < thu1
```

ả ội dung thư từ File thu1 được gửi cho người nhận user2, dấu mời của shell lại hiện ra. Cách làm trên đây hay được sử dụng trong gửi / nhận thư điện tử hoặc liên kết truyền thông vì cho phép tiết kiệm được thời gian kết nối vào hệ thống, đặc biệt chi phí phải trả khi kết nối là đáng kể.

Mode lệnh

ả hư đã nói sử dụng mode lệnh của mail để quản lý hộp thư. Vào mail theo mode lệnh khi dùng lệnh mail không tham số:

```
$mail
```

Sau khi gõ lệnh, màn hình mail ở mode lệnh được hiện ra với dấu mời của mode lệnh. (phổ biến là dấu chấm hỏi "?") Tại đây người dùng sử dụng các lệnh của mail quản lý hệ thống thư của mình.

- Cần trợ giúp gõ dấu chấm hỏi (màn hình có hai dấu ??): ? màn hình hiện ra dạng sau:

<số>	Hiện thư số <số>
(dấu cách)	Hiện thư ngay phía trước
+	Hiện thư ngay tiếp theo
l cmd	thực hiện lệnh cmd
dq	xóa thư hiện thời và ra khỏi mail
m user	gửi thư hiện thời cho người dùng
s tên-file	ghi thư hiện thời vào file có tên
r [tên-file]	trả lời thư hiện thời (có thể từ file)
d <số>	xóa thư số
u	khôi phục thư hiện thời
u <số>	khôi phục thư số
m <user> ...	chuyển tiếp thư tới các người dùng khác
q	ra khỏi mail

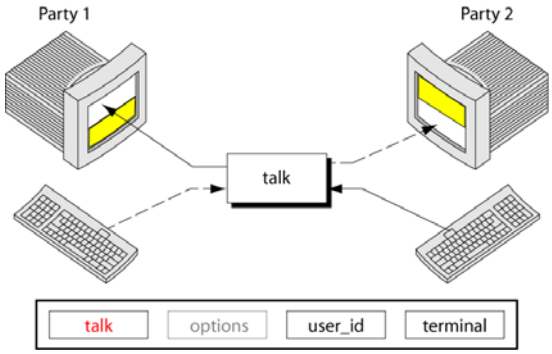
- Thực hiện các lệnh theo chỉ dẫn trên đây để quản trị được hộp thư của cá nhân.

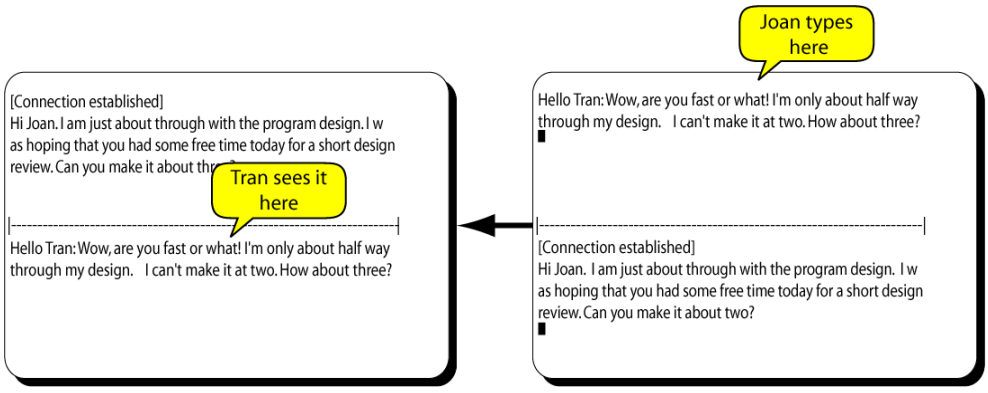
Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

6.5 Lệnh talk

Trong Linux cho phép sử dụng lệnh talk thay thế cho lệnh write.




Joan types here
[Connection established]
Hi Joan. I am just about through with the program design. I w
as hoping that you had some free time today for a short design
review. Can you make it about thr
Tran sees it here
Hello Tran: Wow, are you fast or what! I'm only about half way
through my design. I can't make it at two. How about three?
Hello Tran: Wow, are you fast or what! I'm only about half way
through my design. I can't make it at two. How about three?
[Connection established]
Hi Joan. I am just about through with the program design. I w
as hoping that you had some free time today for a short design
review. Can you make it about two?

TÀI LIỆU THAM KHẢO

[1] Mendel Cooper; *Advanced Bash Script Guide Shell;* 16 June 2002.

[2] Ellie Quigley; *UNIX Shells by Example Fourth Edition,* Prentice Hall PTR, September 24, 2004.

[3] Steve D. Pate; *UNIX Filesystems: Evolution, Design, and Implementation(VERITAS Series); 2003.*

[4] Kirk Bauer, *Automating UNIX and Linux Administration;* 2003.

[5] Kurt Wall, Mark Watson, and Mark Whitis; *Linux Programming*;1999.

[6] Roderick W. Smith;*Advanced Linux Networking*; Addison Wesley; June 11, 2002.

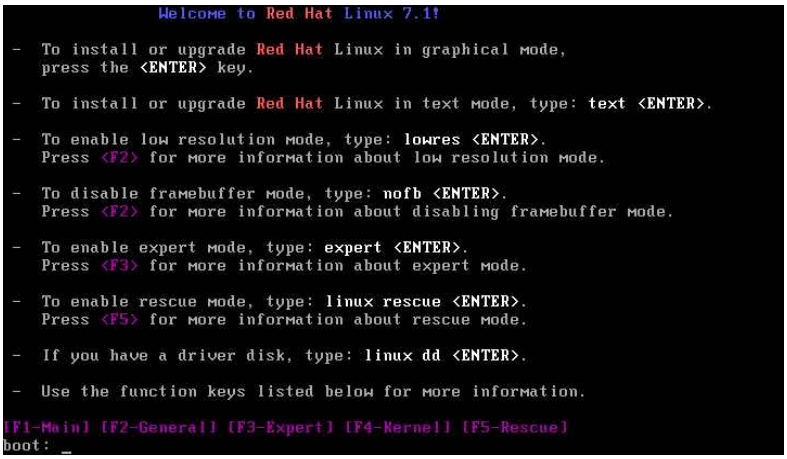
[7] Giáo trình hệ điều hành Unix, Đại học Công nghệ - Đại học Quốc gia Hà ả ội; 2004.

PHỤ LỤC

1. Giới thiệu một số phiên bản hệ điều hành Linux thông dụng hiện nay và cách cài đặt

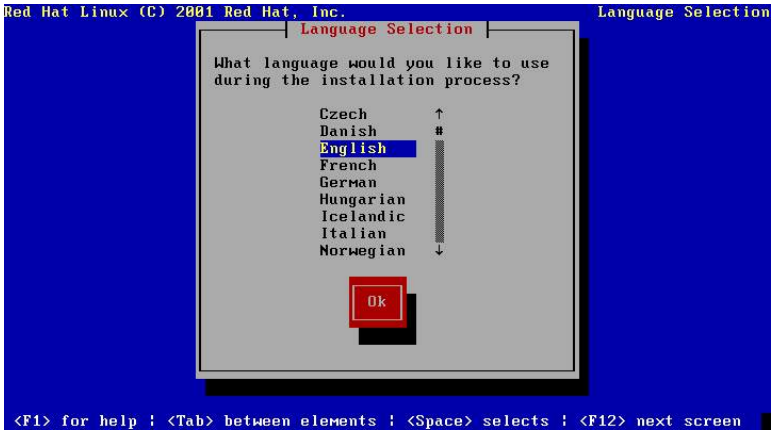
1.1 Hướng dẫn cài đặt hệ điều hành Redhat Linux 7.1

Ta cài Linux từ CDROM. Redhat phiên bản 7.1 gồm có 5 đĩa. Ta chỉ cần 2 đĩa Disk 1 va Dick 2 là đủ và cài trên máy hoàn toàn mới, chưa có cài hệ điều hành nào. Giả sử máy mới có **một ổ cứng chưa định dạng**. Thiết lập CMOS để máy khởi động từ CDROM. Cho đĩa CDROM Linux 1 vào để khởi động. Khi khởi động xong, màn hình hiện lên như sau :

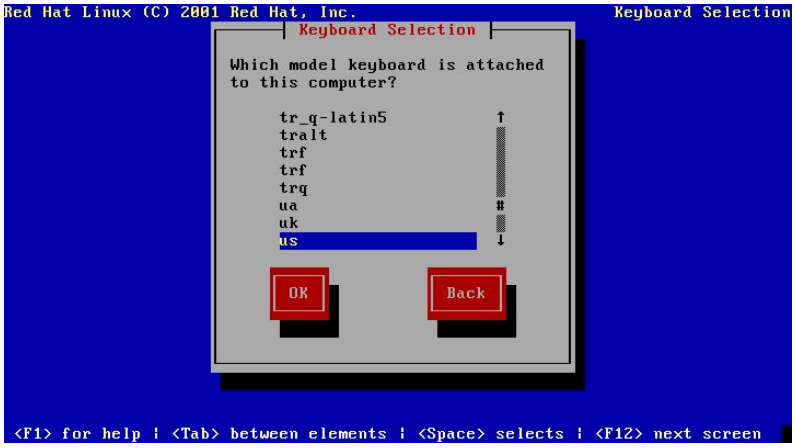


Từ đây, cho phộp ta cài đặt theo nhiều loại giao diện, ta gõ vào **text** để cài trong chế độ text. Màn hình tiếp theo như sau:

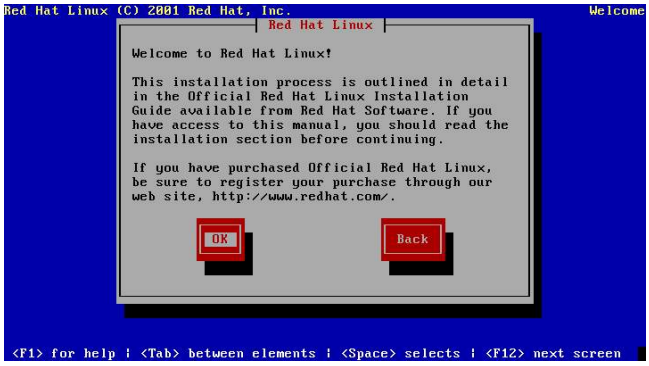
Sau đó hộp thoại chọn ngôn ngữ hiện lên cho ta chọn ngôn ngữ nào tuỳ ý. Ở đây ta chọn English**.**



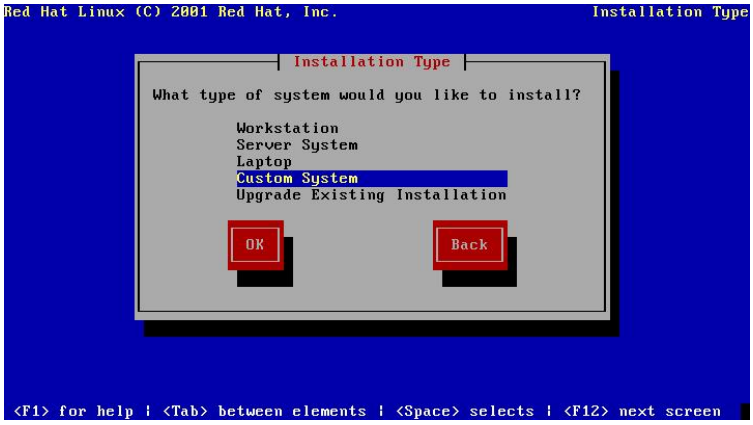
Tiếp theo chọn **us**



Chương trình hiện lên bảng thông báo Red Hat Linux: Welcome to Red Hat Linux, nhấn **OK** để bỏ qua.

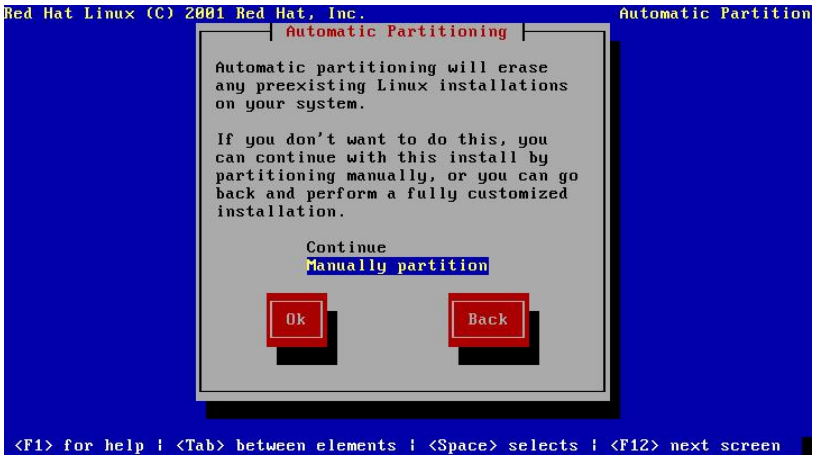


Khi hộp thoại Installation Type xuất hiện, chọn Custom System, trong trường hợp này nó sẽ cho ta chọn cài các phần mềm theo ý của ta. ẩ ếu ta chọn các tuỳ chọn khác như Server System, Workstation. Chọn Upgrade Existing Installation sẽ cho phép ta nâng cấp hệ điều hành Linux.

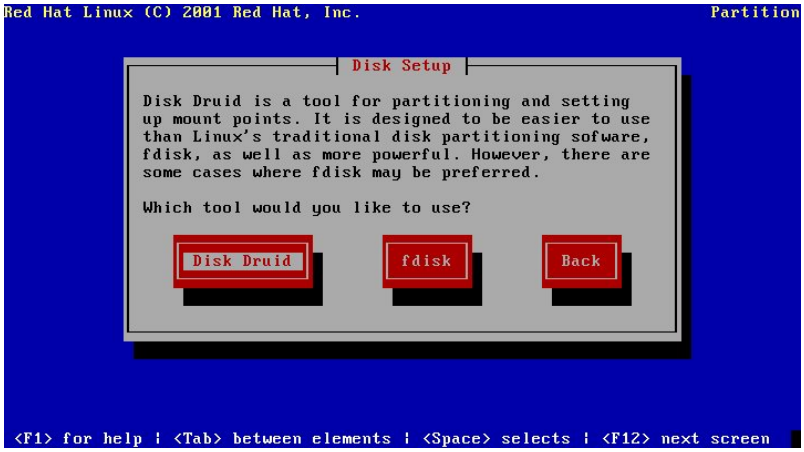


Do đĩa cứng mới hoàn toàn chưa định dạng nên xuất hiện thông báo sau. Ta chọn Initilize

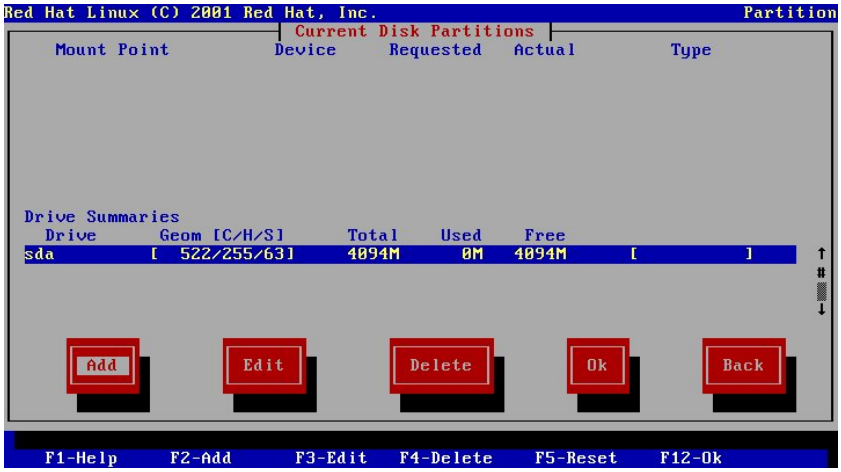
Khi gặp bảng Automatic Partitioning, chọn Manual partition và OK



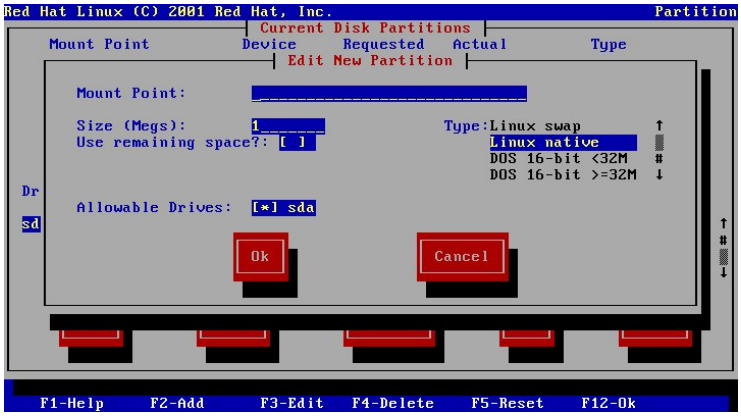
Tiếp đến chọn Disk Druid



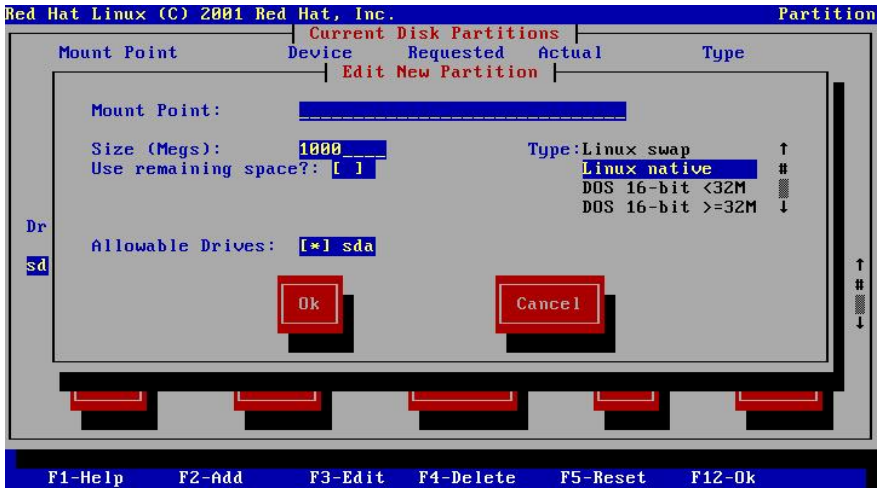
Chương trình định dạng đĩa của Linux mở ra.Theo vệt sáng ta thấy có 1 đĩa cứng gắn ở IDE1 (sda), dung lượng là 4094 MB



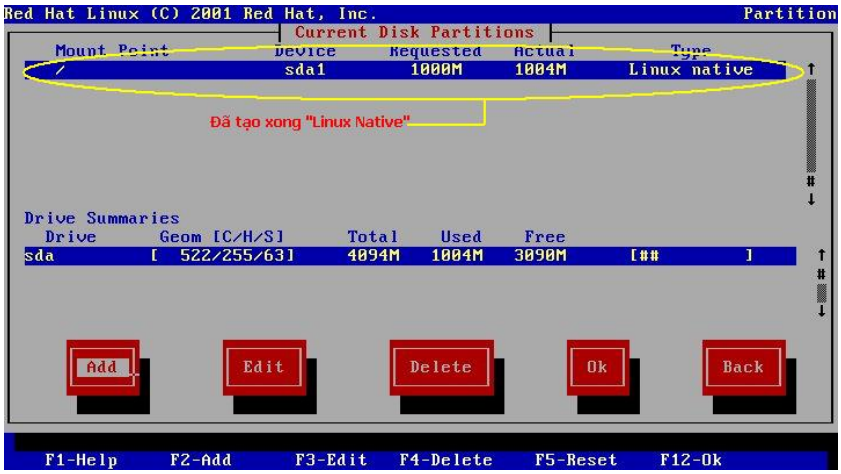
Đưa vệt sáng đến nút Add, nhấn Eả TER để tạo partition mới, cửa sổ Edit ả ew Partition xuất hiện



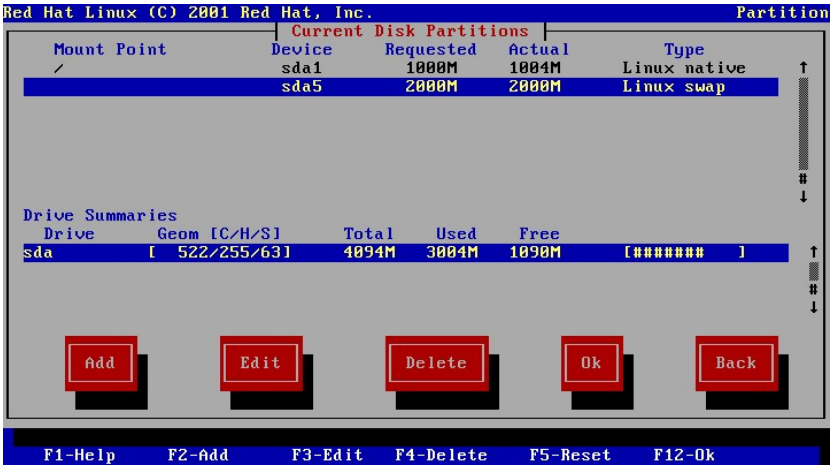
Trước tiên, ta tạo Linux ả ative, ở đây ta chọn tạm 1000 MB, nhấn TAB để chuyển qua lại và thiết lập thụng số. Thêm dấu / vào mục Mount Point.



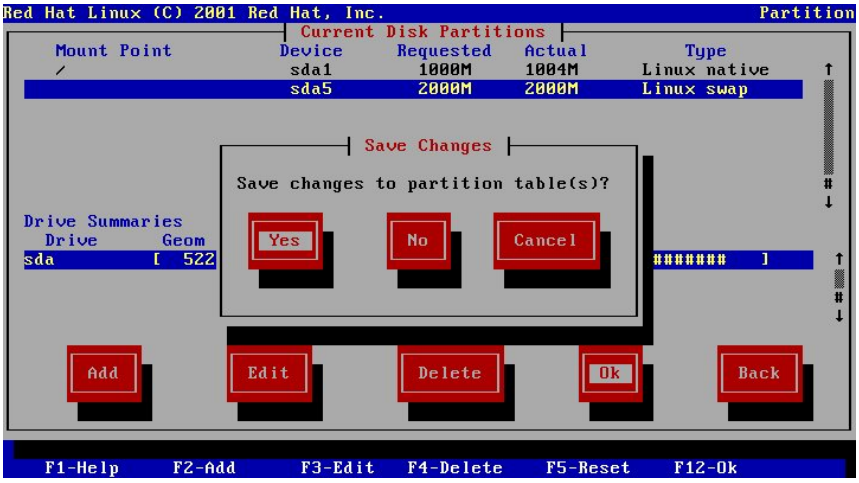
Sau đó ta sẽ thấy partition 1 đó tạo xong



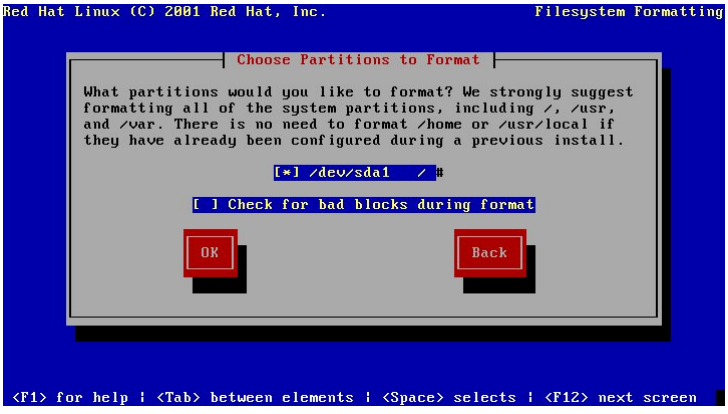
ả hấn Add để tạo tiếp Swap partition:



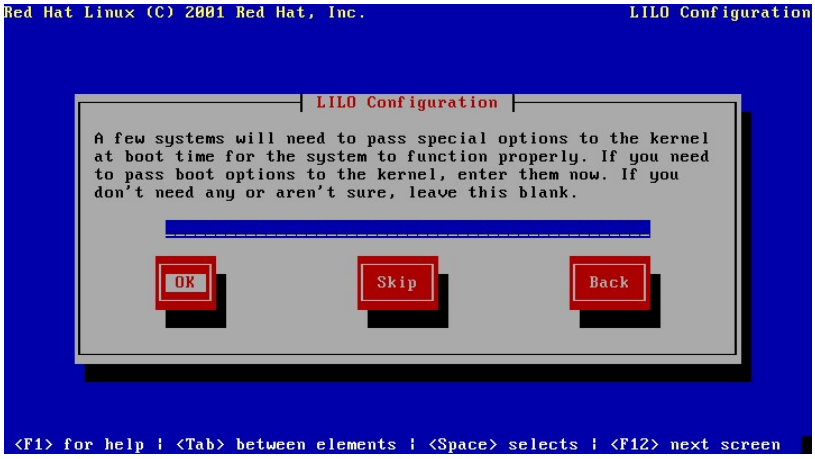
ả hấn OK để kết thúc, CT sẽ hỏi có muốn lưu lại có thay đổi không. Chọn Yes



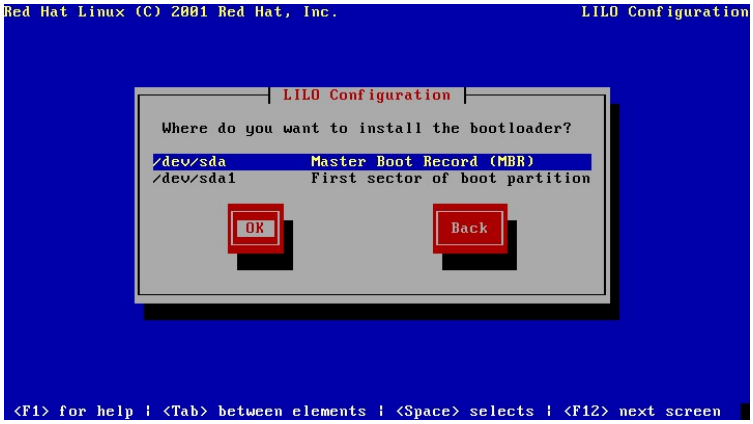
ả hấn tiếp OK để tiến hành format.



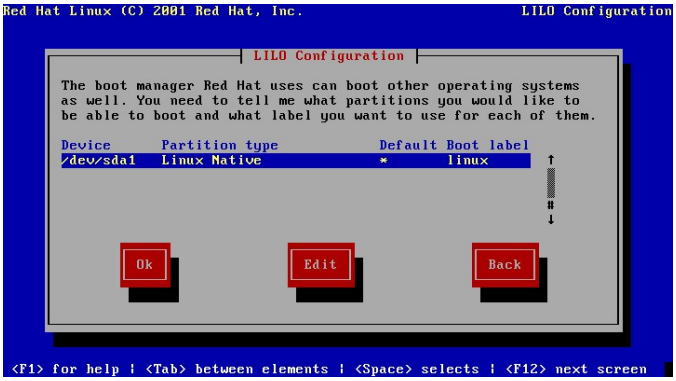
ả hấn OK để tạo boot LILO. Trong các phiên bản sau này cũng có thêm một chương trình GRUB có vai trò tương tự như LILO nhưng cú giao diện đồ hoạ.



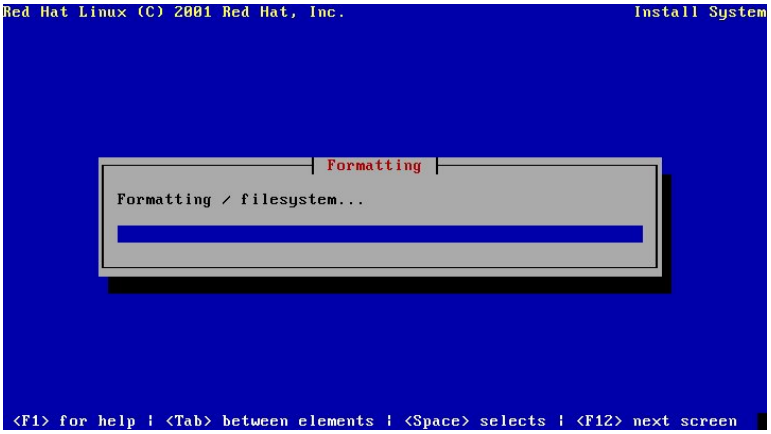
Mặc định dựng Master Boot Record để khởi động Linux, nhấn OK.



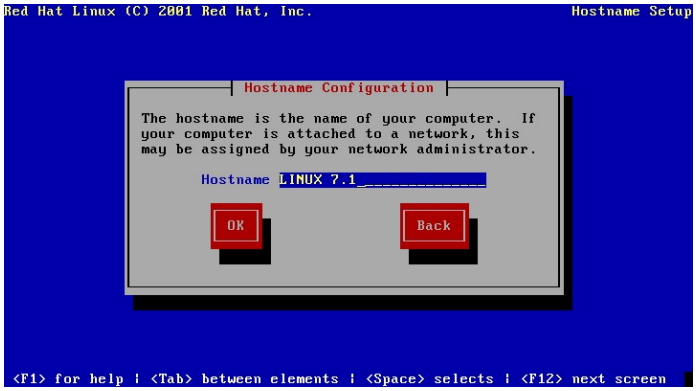
ả hấn tiếp OK.



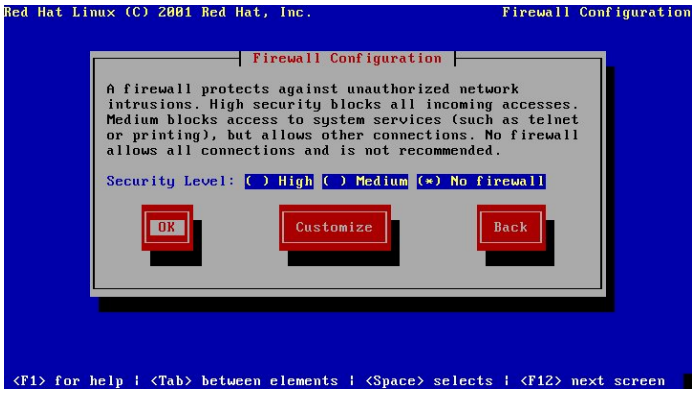
ả hấn OK để format bắt đầu.



Sau đó đặt tên cho máy

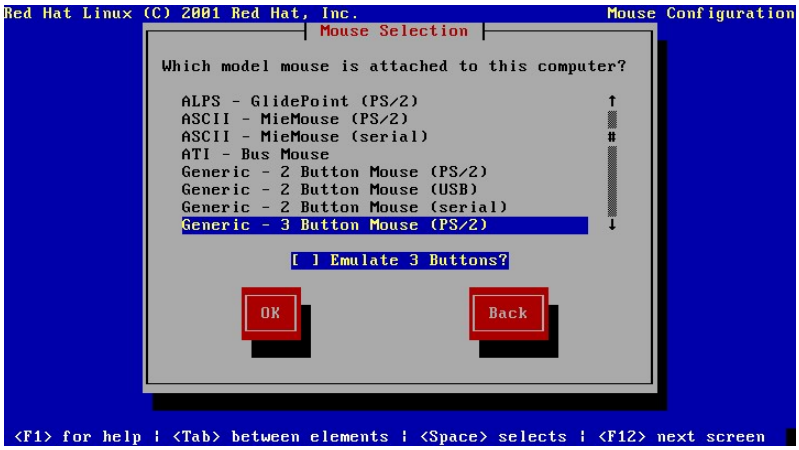


Thiết lập cấu hình cho Firewall, chọn ả o firewall , nhấn OK

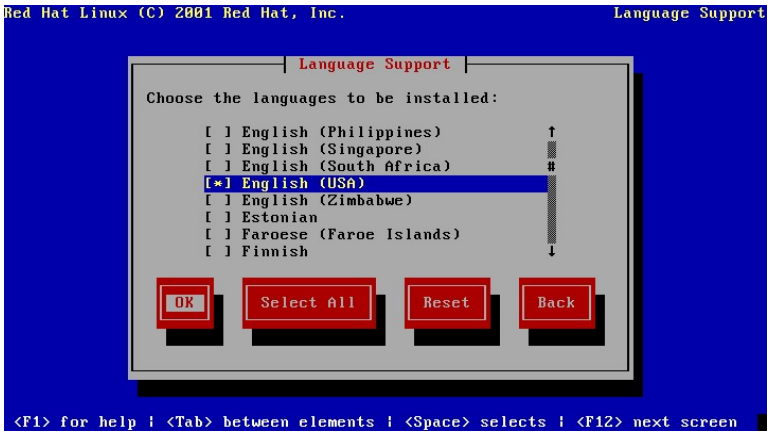


Tiếp theo đó, thiết lập các cấu hình về phần cứng.

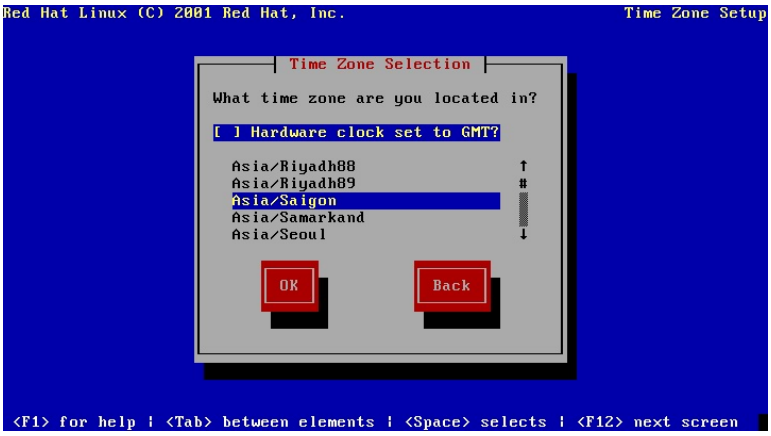
Lưu ý: ẩ những minh hoạ dưới đây là được thiết lập theo cấu hình phần cứng máy hiện tại. các máy khác có thể khác. Chọn loại chuột:



Chọn ngôn ngữ làm việc



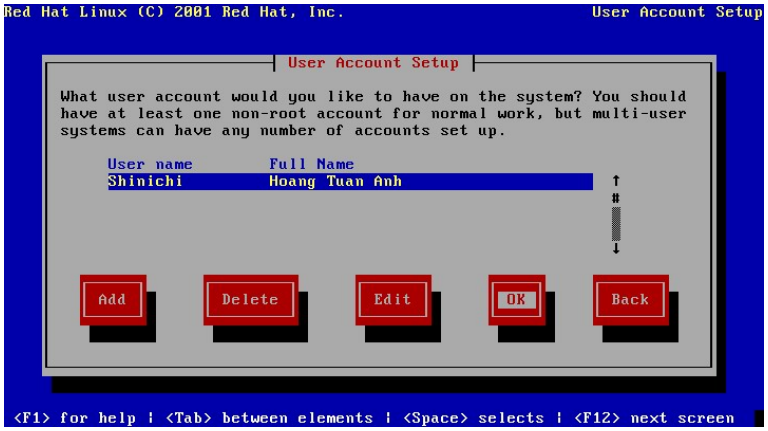
Xác định múi giờ làm việc, chọn Asia/Saigon



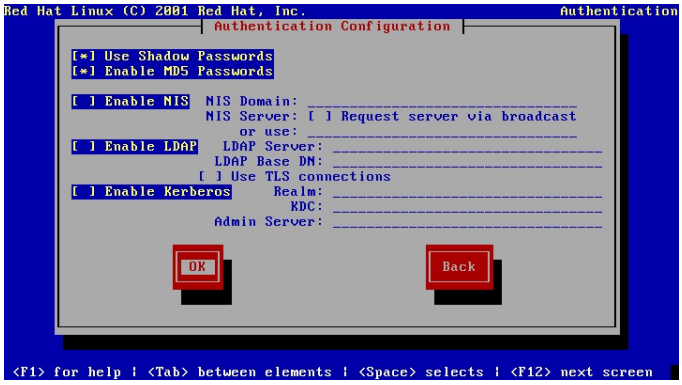
Thiết lập mật khẩu cho tài khoản người quản trị hệ thống (root)

Mật khẩu của root phải có ít nhất là 6 ký tự

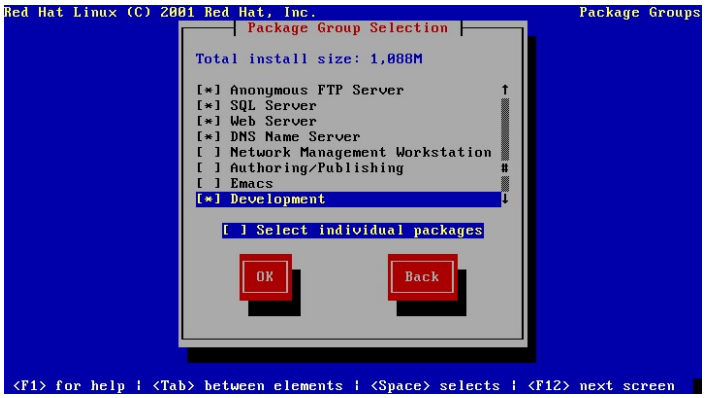
ả hấn OK nếu không có thay đổi.

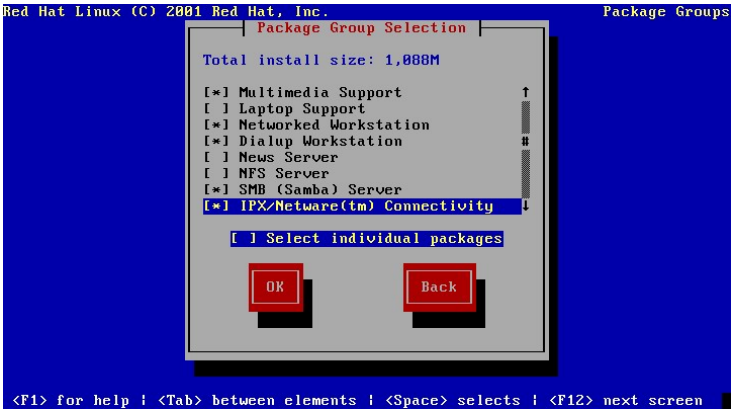


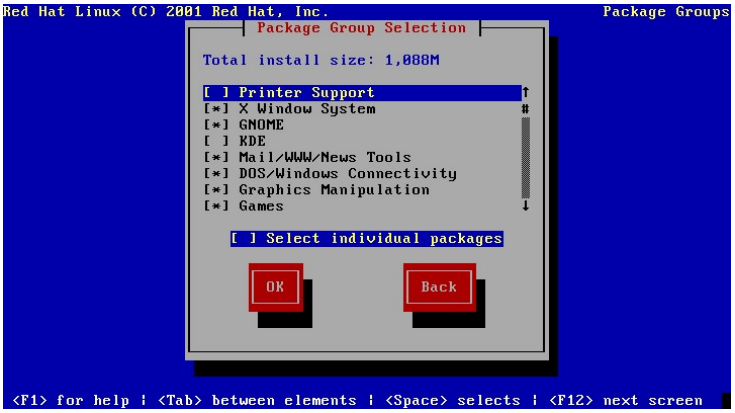
Sau khi tạo xong nhấn OK.



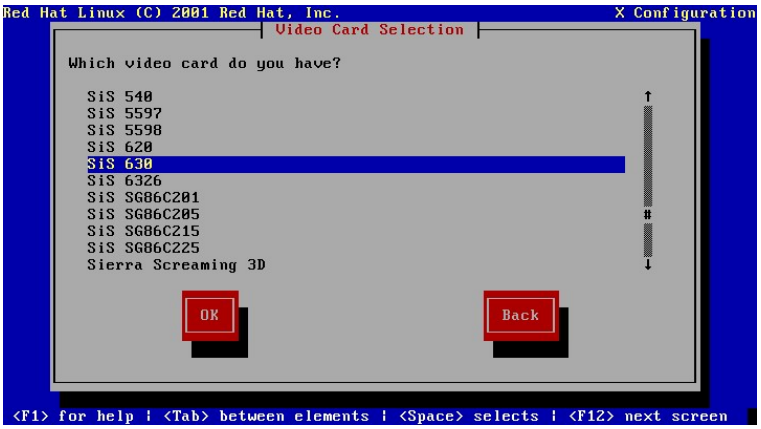
Sau đó chọn các gói cài đặt (tùy ý), các phần mà ta thấy cần thiết.



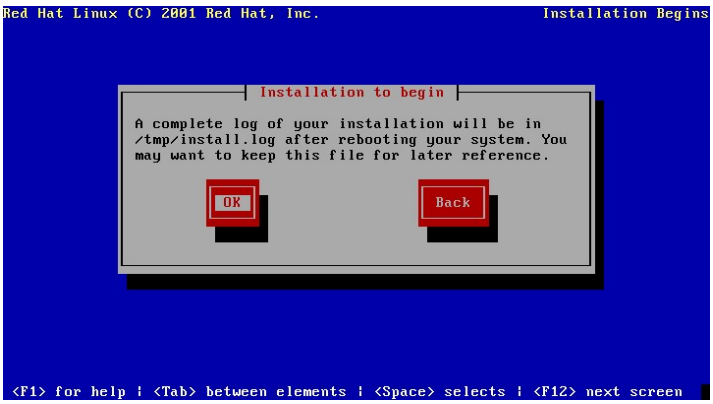




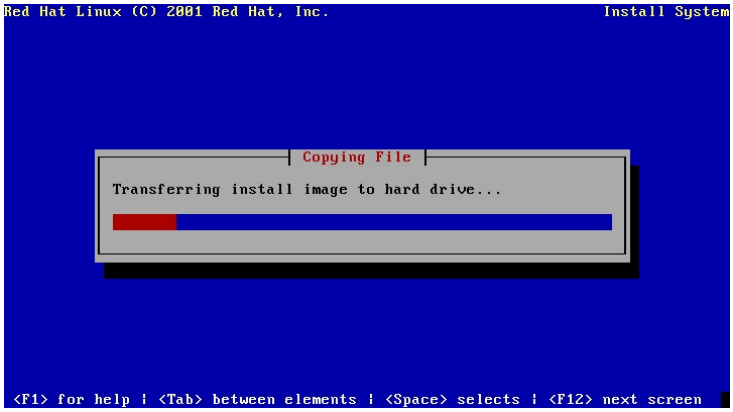
ả hấn OK khi chọn xong. Chọn loại card màn hình thích hợp.

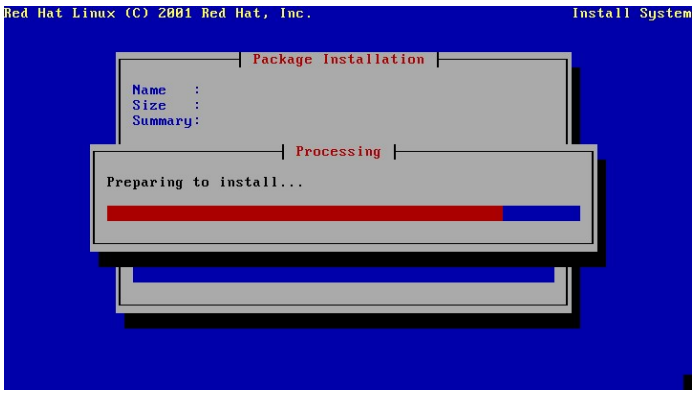


ả hấn OK để cài đặt

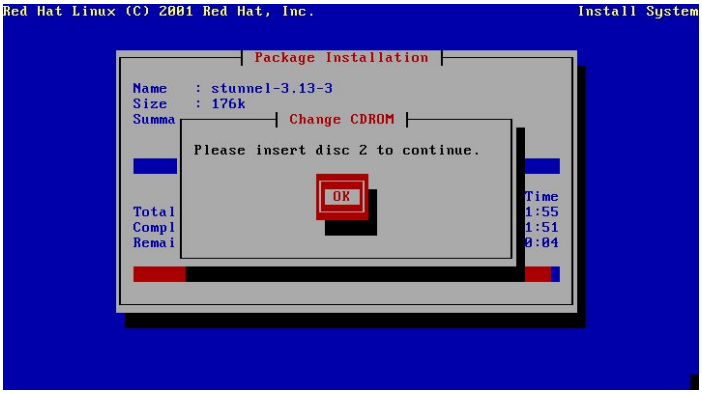


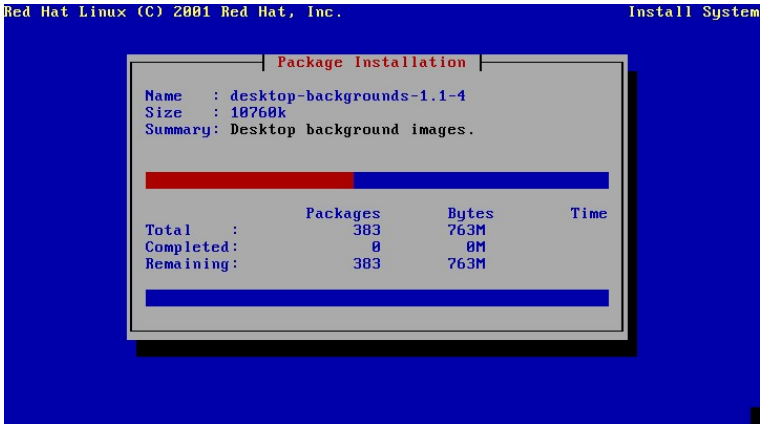
Quá trình cài đặt bắt đầu

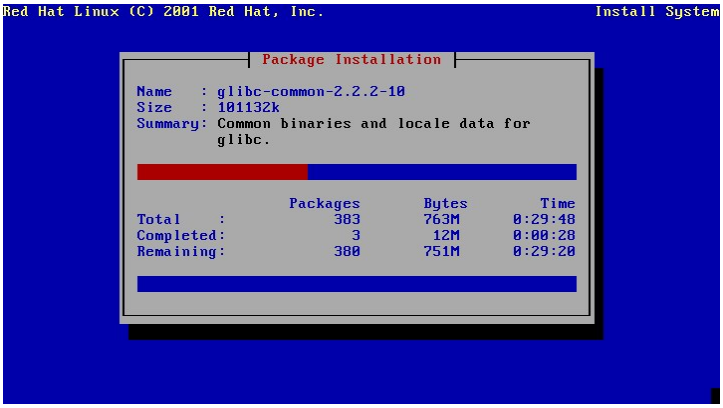


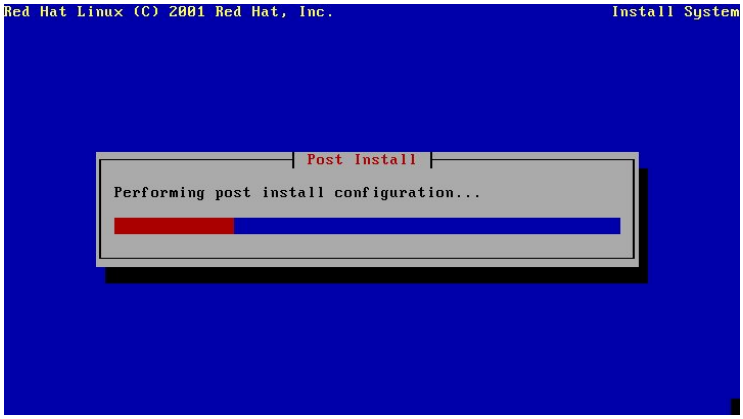


Trong quá trình cài, chương trình yêu cầu đưa đĩa 2 vào

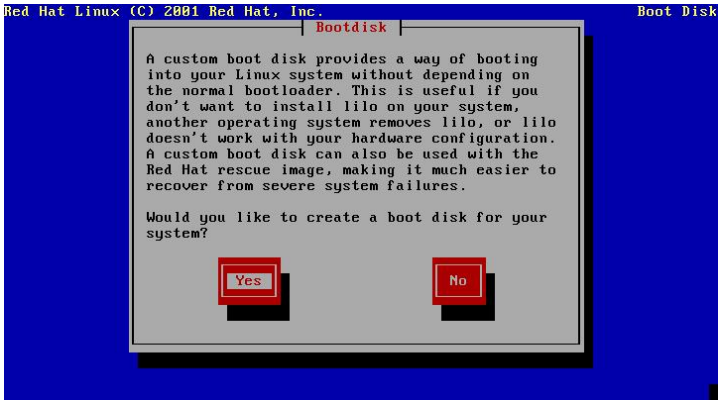

Red Hat Linux (C) 2001 Red Hat, Inc.
Install System
Package Installation
Name : stunnel-3.13-3
Size : 176k
Summa
Change CDROM
Please insert disc 2 to continue.
OK
Time
Total 1:55
Compl 1:51
Remai 0:04


Red Hat Linux (C) 2001 Red Hat, Inc.
Install System
Package Installation
Name : desktop-backgrounds-1.1-4
Size : 10760k
Summary: Desktop background images.
Packages Bytes Time
Total : 383 763M
Completed: 0 0M
Remaining: 383 763M

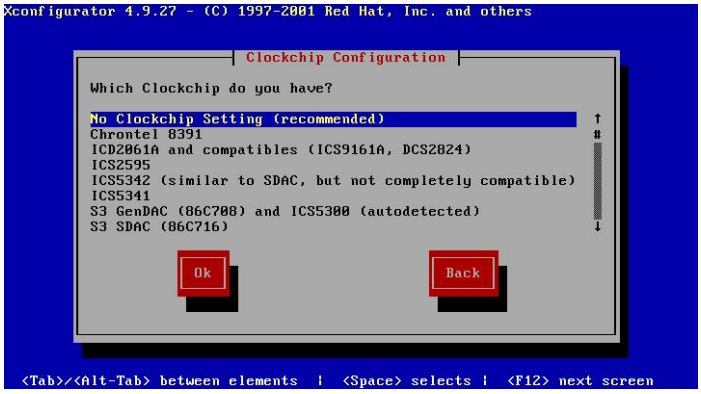

Red Hat Linux (C) 2001 Red Hat, Inc.
Install System
Package Installation
Name : glibc-common-2.2.2-10
Size : 101132k
Summary: Common binaries and locale data for glibc.
Packages Bytes Time
Total : 383 763M 0:29:48
Completed: 3 12M 0:00:28
Remaining: 380 751M 0:29:20

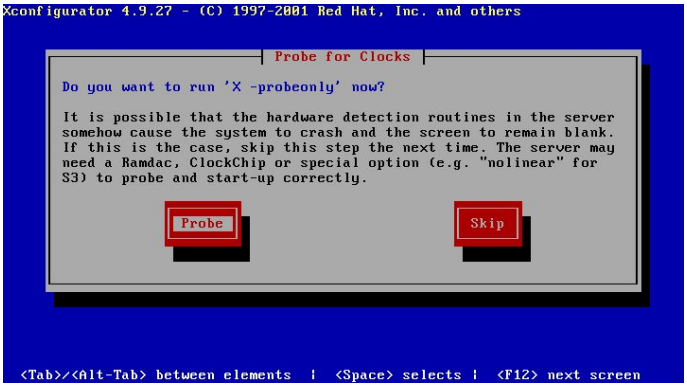


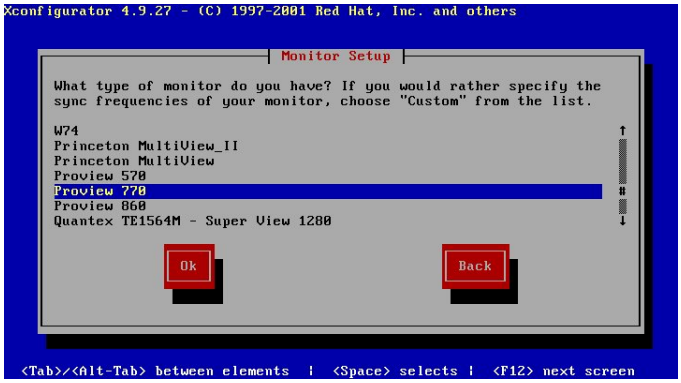
Tiếp theo chương trình sẽ hỏi có muốn tạo đĩa boot hay không,

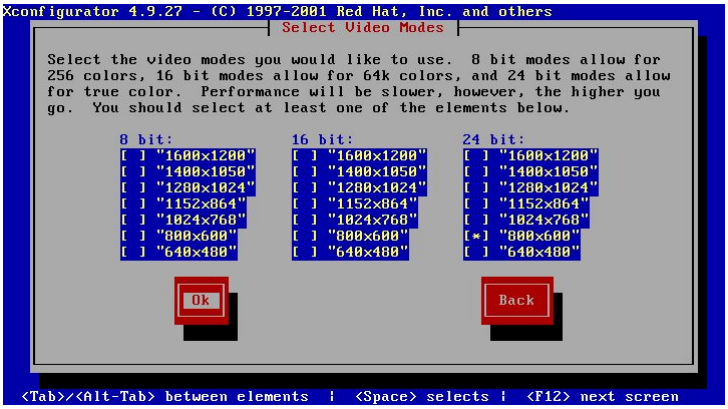


Sau đố thiết lập các thông số khác nữa là hoàn tất

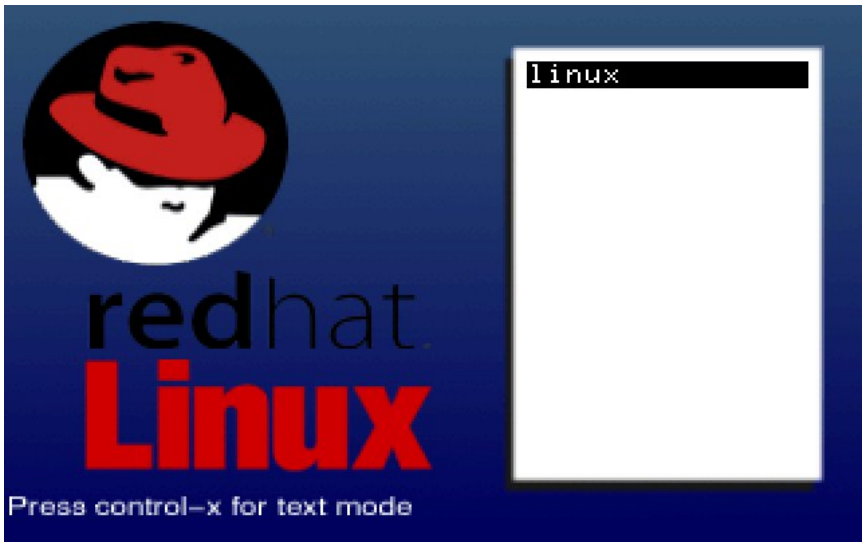








Sau khi khởi động máy, chúng ta thấy biểu tượng sau:



1.2 Hướng dẫn sử dụng hệ điều hành Ubuntu và các phiên bản của nó

2. Cài đặt WEBMIN

3. Cài đặt WEBSERVER

4. Cài đặt FILE SERVER

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

==========

HÀ QUANG THỤY
NGUYỄN TRÍ THÀNH

Giáo trình:

HỆ ĐIỀU HÀNH UNIX - LINUX

Dành cho sinh viên ngành Công nghệ thông tin,
Điện tử - Viễn thông, Toán tin ứng dụng

HÀ NỘI - 2004

MỤC LỤC

LỜI GIỚI THIỆU

Trong hơn mười năm trở lại đây hệ điều hành Linux đã

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ LINUX

1.1. Giới thiệu về UNIX và Linux

1.1.1. Xuất xứ, quá trình tiến hóa và một số đặc trưng của hệ điều hành UNIX

Năm 1965, Viện công nghệ Massachusetts (MIT: Massachusetts Institute of Technology) và Phòng thí nghiệm Bell của hãng AT&T thực hiện dự án xây dựng một hệ điều hành có tên gọi là Multics (MULTiplexed Information and Computing Service) với mục tiêu: tạo lập được một hệ điều hành phủ trên vùng lãnh thổ rộng (hoạt động trên tập các máy tính được kết nối), đa người dùng, có năng lực cao về tính toán và lưu trữ. Dự án nói trên thành công ở mức độ hết sức khiêm tốn và người ta đã biết đến một số khiếm khuyết khó khắc phục của Multics.

Năm1969, Ken Thompson, một chuyên viên tại phòng thí nghiệm Bell, người đã tham gia dự án Multics, cùng Dennics Richie viết lại hệ điều hành đa-bài toán trên máy PDP-7 với tên là UNICS (UNiplexed Information and Computing Service) từ một câu gọi đùa của một đồng nghiệp. Trong hệ điều hành UNICS, một số khởi thảo đầu tiên về Hệ thống file đã được Ken Thompson và Dennis Ritchie thực hiện. Đến năm 1970 hệ điều hành được viết trên assembler cho máy PDP-11/20 và mang tên là UNIX.

Năm 1973, Riche và Thompson viết lại nhân của hệ điều hành UNIX trên ngôn ngữ C, và hệ điều hành đã trở nên dễ dàng cài đặt tới các loại máy tính khác nhau; tính chất như thế được gọi là tính khả chuyển (portable) của UNIX. Trước đó, khoảng năm 1971, hệ điều hành được thể hiện trên ngôn ngữ B (mà dựa trên ngôn ngữ B, Ritche đã phát triển thành ngôn ngữ C).

Hãng AT&T phổ biến chương trình nguồn UNIX tới các trường đại học, các công ty thương mại và chính phủ với giá không đáng kể.

Năm 1982, hệ thống UNIX-3 là bản UNIX thương mại đầu tiên của AT&T.

Năm 1983, AT&T giới thiệu Hệ thống UNIX-4 phiên bản thứ nhất trong đó đã có trình soạn thảo **vi**, thư viện quản lý màn hình được phát triển từ Đại học Tổng hợp California, Berkley.

Giai đoạn 1985-1987, UNIX-5 phiên bản 2 và 3 tương ứng được đưa ra vào các năm 1985 và 1987. Trong giai đoạn này, có khoảng 100000 bản UNIX đã được phổ biến trên thế giới, cài đặt từ máy vi tính đến các hệ thống lớn.

Đầu thập kỷ 1990. UNIX-5 phiên bản 4 được đưa ra như là một chuẩn của UNIX. Đây là sự kết hợp của các bản sau:

- AT&T UNIX-5 phiên bản 3,
- Berkley Software Distribution (BSD),
- XENIX của MicroSoft
- SUN OS

Có thể tìm thấy các nội dung liên quan tới một số phiên bản mới của UNIX tại địa chỉ website http://problem.rice.edu/.

Các nhóm nhà cung cấp khác nhau về UNIX đang hoạt động trong thời gian hiện nay được kể đến như sau:

- Unix International (viết tắt là UI). UI là một tổ chức gồm các nhà cung cấp thực hiện việc chuyển nhượng hệ thống UNIX-5 và cung cấp bản AT&T theo các

nhu cầu và thông báo phát hành mới, chẳng hạn như điều chỉnh bản quyền. Giao diện đồ họa người dùng là Open Look.

- Open Software Foundation (OSF). OSF được hỗ trợ bởi IBM, DEC, HP ... theo hướng phát triển một phiên bản của Unix nhằm tranh đua với hệ thống UNIX-5 phiên bản 4. Phiên bản này có tên là OSF/1 với giao diện đồ họa người dùng được gọi là MOTIF.
- Free SoftWare Foundation (FSF): một cộng đồng do Richard Stallman khởi xướng năm 1984 chủ trương phát hành các phần mềm sử dụng tự do, trên cơ sở một hệ điều hành thuộc loại UNIX.

Bảng sau đây liệt kê một số cài đặt UNIX khá phổ biến (thường thấy có chữ X ở cuối tên gọi của Hệ điều hành):

Tên hệ	*Nhà cung cấp*	*Nền phát triển*
AIX	International Business Machines	AT&T System V
A/UX	Apple Computer	AT&T System V
Dynix	Sequent	BSD (Berkeley SoftWare Distribution)
HP-UX	Hewlett-Packard	BSD
Irix	Silicon Graphics	AT&T System V
Linux	Free SoftWare Foundation	
NextStep	Next	BSD
OSF/1	Digital Equipment Corporation	BSD
SCO UNIX	Santa Cruz Operation	AT&T System V
Solaris	Sun Microsystems	AT&T System V
SunOS	Sun Microsystems	BSD UNIX
Ultrix	Digital Equipment Corporation	BSD UNIX
Unicos	Cray	AT&T System V
UnixWare	Novell	AT&T System V
XENIX	MicroSoft	AT&T System III-MS

Dưới đây liệt kê một số đặc trưng của hệ điều hành UNIX:

- Hệ điều hành được viết trên ngôn ngữ bậc cao; bởi vậy, rất dễ đọc, dễ hiểu, dễ thay đổi để cài đặt trên loại máy mới (tính dễ mang chuyển, như đã nói),
- Có giao diện người dùng đơn giản đủ năng lực cung cấp các dịch vụ mà người dùng mong muốn (so sánh với các hệ điều hành có từ trước đó thì giao diện của UNIX là một tiến bộ vượt bậc),
- Thỏa mãn nguyên tắc xây dựng các chương trình phức tạp từ những chương trình đơn giản hơn: trước hết có các môđun cơ bản nhất của nhân sau đó phát triển để có toàn bộ hệ điều hành,
- Sử dụng duy nhất một hệ thống File có cấu trúc cho phép dễ dàng bảo quản và sử dụng hiệu quả,

- Sử dụng phổ biến một dạng đơn giản trình bày nội tại của File như một dòng các byte cho phép dễ dàng khi viết các chương trình ứng dụng truy nhập, thao tác với các dữ liệu trong File,
- Có kết nối đơn giản với thiết bị ngoại vi: các file thiết bị đã được đặt sẵn trong Hệ thống File tạo ra một kết nối đơn giản giữa chương trình người dùng với các thiết bị ngoại vi,
- Là hệ điều hành đa người dùng, đa quá trình, trong đó mỗi người dùng có thể thực hiện các quá trình của mình một cách độc lập.
- Mọi thao tác vào - ra của hệ điều hành được thực hiện trên hệ thống File: mỗi thiết bị vào ra tương ứng với một file. Chương trình người dùng làm việc với file đó mà không cần quan tâm cụ thể tên file đó được đặt cho thiết bị nào trong hệ thống.
- Che khuất cấu trúc máy đối với người dùng, đảm bảo tính độc lập tương đối của chương trình đối với dữ liệu và phần cứng, tạo điều kiện thuận lợi hơn cho người lập trình khi viết các chương trình chạy UNIX với các điều kiện phần cứng hoàn toàn khác biệt nhau.

1.1.2. Giới thiệu sơ bộ về Linux

Linus Tovalds (một sinh viên Phần lan) đưa ra nhân (phiên bản đầu tiên) cho hệ điều hành Linux vào tháng 8 năm 1991 trên cơ sở cải tiến một phiên bản UNIX có tên Minix do Giáo sư Andrew S. Tanenbaum xây dựng và phổ biến. Nhân Linux tuy nhỏ song là tự đóng gói. Kết hợp với các thành phần trong hệ thống GNU, hệ điều hành Linux đã được hình thành. Và cũng từ thời điểm đó, theo tư tưởng GNU, hàng nghìn, hàng vạn chuyên gia trên toàn thế giới (những người này hình thành nên cộng đồng Linux) đã tham gia vào quá trình phát triển Linux và vì vậy Linux ngày càng đáp ứng nhu cầu của người dùng.
Dưới đây là một số mốc thời gian quan trọng trong quá trình hình thành và phát triển hệ điều hành Linux.

- Sau ba năm nhân Linux ra đời, đến ngày 14-3-1994, hệ điều hành Linux phiên bản 1.0 được phổ biến. Thành công lớn nhất của Linux 1.0 là nó đã hỗ trợ giao thức mạng TCP/IP chuẩn UNIX, sánh với giao thức socket BSD- tương thích cho lập trình mạng. Trình điều khiển thiết bị đã được bổ sung để chạy IP trên một mạng Ethernet hoặc trên tuyến đơn hoặc qua modem. Hệ thống file trong Linux 1.0 đã vượt xa hệ thống file của Minix thông thường, ngoài ra đã hỗ trợ điều khiển SCSI truy nhập đĩa tốc độ cao. Điều khiển bộ nhớ ảo đã được mở rộng để hỗ trợ điều khiển trang cho các file swap và ánh xạ bộ nhớ của file đặc quyền (chỉ có một ánh xạ bộ nhớ chỉ đọc được thi hành trong Linux 1.0).
- Vào tháng 3-1995, nhân 1.2 được phổ biến. Điều đáng kể của Linux 1.2 so với Linux 1.0 ở chỗ nó hỗ trợ một phạm vi rộng và phong phú phần cứng, bao gồm cả kiến trúc tuyến phần cứng PCI mới. Nhân Linux 1.2 là nhân kết thúc dòng nhân Linux chỉ hỗ trợ PC.

  Một điều cần lưu ý về các đánh chỉ số các dòng nhân (hệ điều hành) Linux. Hệ thống chỉ số được chia thành một số mức, chẳng hạn hai mức như 2.4 hoặc ba mức như 2.2.5. Trong cách đánh chỉ số như vậy, quy ước rằng với các chỉ số từ mức thứ hai trở đi, nếu là số chẵn thì dòng nhân đó đã khá ổn định và tương đối hoàn thiện, còn nếu là số lẻ thì dòng nhân đó vẫn đang được phát triển tiếp.

- Tháng 6-1996, nhân Linux 2.0 được phổ biến. Có hai đặc trưng nổi bật của Linux 2.0 là hỗ trợ kiến trúc phức hợp, bao gồm cả cổng Alpha 64-bit đầy đủ, và hỗ trợ kiến trúc đa bộ xử lý. Phân phối nhân Linux 2.0 cũng thi hành được trên bộ xử lý Motorola 68000 và kiến trúc SPARC của SUN. Các thi hành của Linux dựa trên vi nhân GNU Mach cũng chạy trên PC và PowerMac.
- Tới năm 2000, nhân Linux 2.4 được phổ biến. Một trong đặc điểm được quan tâm của nhân này là nó hỗ trợ mã ký tự Unicode 32 bít, rất thuận lợi cho việc xây dựng các giải pháp toàn diện và triệt để đối với vấn đề ngôn ngữ tự nhiên trên phạm vi toàn thế giới.

Vấn đề phân phối và giấy phép Linux

Về lý thuyết, mọi người có thể khởi tạo một hệ thống Linux bằng cách tiếp nhận bản mới nhất các thành phần cần thiết từ các site ftp và biên dịch chúng. Trong thời kỳ đầu tiên, người dùng Linux phải tiến hành toàn bộ các thao tác này và vì vậy công việc là khá vất vả. Tuy nhiên, do có sự tham gia đông đảo của các cá nhân và nhóm phát triển Linux, đã tiến hành thực hiện nhiều giải pháp nhằm làm cho công việc khởi tạo hệ thống đỡ vất vả. Một trong những giải pháp điển hình nhất là cung cấp tập các gói chương trình đã tiền dịch, chuẩn hóa.

Những tập hợp như vậy hay những bản phân phối là lớn hơn nhiều so với hệ thống Linux cơ sở. Chúng thường bao gồm các tiện ích bổ sung cho khởi tạo hệ thống, các thư viện quản lý, cũng như nhiều gói đã được tiền dịch, sẵn sàng khởi tạo của nhiều bộ công cụ UNIX dùng chung, chẳng hạn như phục vụ tin, trình duyệt web, công cụ xử lý, soạn thảo văn bản và thậm chí các trò chơi.

Cách thức phân phối ban đầu rất đơn giản song ngày càng được nâng cấp và hoàn thiện bằng phương tiện quản lý gói tiên tiến. Các bản phân phối ngày nay bao gồm các cơ sở dữ liệu tiến hóa gói, cho phép các gói dễ dàng được khởi tạo, nâng cấp và loại bỏ.

Nhà phân phối đầu tiên thực hiện theo phương châm này là Slakware, và chính họ là những chuyển biến mạnh mẽ trong cộng đồng Linux đối với công việc quản lý gói khởi tạo Linux.

Tiện ích quản lý gói RPM (RedHat Package Manager) của công ty RedHat là một trong những phương tiện điển hình.

Nhân Linux là phần mềm tự do được phân phối theo Giấy phép sở hữu công cộng phần mềm GNU GPL.

Các thành phần tích hợp Hệ điều hành Linux

Linux sử dụng rất nhiều thành phần từ Dự án phần mềm tự do GNU, từ hệ điều hành BSD của Đại học Berkeley và từ hệ thống X-Window của MIT.

Thư viện hệ thống chính của Linux được bắt nguồn từ Dự án GNU, sau đó được rất nhiều người trong cộng đồng Linux phát triển tiếp, những phát triển tiếp theo như vậy chủ yếu liên quan tới việc giải quyết các vấn đề như thiếu vắng địa chỉ (lỗi trang), thiếu hiệu quả và gỡ rối. Một số thành phần khác của Dự án GNU, chẳng hạn như trình biên dịch GNU C (gcc), vốn là chất lượng cao nên được sử dụng nguyên xy trong Linux.

Các tool quản lý mạng được bắt nguồn từ mã 4.3BSD song sau đó đã được cộng đồng Linux phát triển, chẳng hạn như thư viện toán học đồng xử lý dấu chấm động Intel và các trình điều khiển thiết bị phần cứng âm thanh PC. Các tool quản lý mạng này sau đó lại được bổ sung vào hệ thống BSD.

Hệ thống Linux được duy trì gần như bởi một mạng lưới không chặt chẽ các nhà phát triển phần mềm cộng tác với nhau qua Internet, mạng lưới này gồm các nhóm nhỏ và cá nhân

chịu trách nhiệm duy trì tính toàn vẹn của từng thành phần. Một lượng nhỏ các site phân cấp ftp Internat công cộng đã đóng vai trò nhà kho theo chuẩn de facto để chứa các thành phần này. Tài liệu Chuẩn phân cấp hệ thống file (File System Hierarchy Standard) được cộng đồng Linux duy trì nhằm giữ tính tương thích xuyên qua sự khác biệt rất lớn giữa các thành phần hệ thống.

Một số đặc điểm chính của Linux

Dưới đây trình bày một số đặc điểm chính của của hệ điều hành Linux hiện tại:

- Linux tương thích với nhiều hệ điều hành như DOS, MicroSoft Windows ...:
- Cho phép cài đặt Linux cùng với các hệ điều hành khác trên cùng một ổ cứng. Linux có thể truy nhập đến các file của các hệ điều hành cùng một ổ đĩa. Linux cho phép chạy mô phỏng các chương trình thuộc các hệ điều hành khác.
- Do giữ được chuẩn của UNIX nên sự chuyển đổi giữa Linux và các hệ UNIX khác là dễ dàng.
- Linux là một hệ điều hành UNIX tiêu biểu với các đặc trưng là đa người dùng, đa chương trình và đa xử lý.
- Linux có giao diện đồ hoạ (GUI) thừa hưởng từ hệ thống X-Window. Linux hỗ trợ nhiều giao thức mạng, bắt nguồn và phát triển từ dòng BSD. Thêm vào đó, Linux còn hỗ trợ tính toán thời gian thực.
- Linux khá mạnh và chạy rất nhanh ngay cả khi nhiều quá trình hoặc nhiều cửa sổ.
- Linux được cài đặt trên nhiều chủng loại máy tính khác nhau như PC, Mini và việc cài đặt khá thuận lợi. Tuy nhiên, hiện nay chưa xuất hiện Linux trên máy tính lớn (mainframe).
- Linux ngày càng được hỗ trợ bởi các phần mềm ứng dụng bổ sung như soạn thảo, quản lý mạng, quản trị cơ sở dữ liệu, bảng tính ...
- Linux hỗ trợ tốt cho tính toán song song và máy tính cụm (PC-cluster) là một hướng nghiên cứu triển khai ứng dụng nhiều triển vọng hiện nay.
- Là một hệ điều hành với mã nguồn mở, được phát triển qua cộng đồng nguồn mở (bao gồm cả Free Software Foundation) nên Linux phát triển nhanh. Linux là một trong một số ít các hệ điều hành được quan tâm nhiều nhất trên thế giới hiện nay.
- Linux là một hệ điều hành hỗ trợ đa ngôn ngữ một cách toàn diện nhất. Do Linux cho phép hỗ trợ các bộ mã chuẩn từ 16 bit trở lên (trong đó có các bộ mã ISO10646, Unicode) cho nên việc bản địa hóa trên Linux là triệt để nhất trong các hệ điều hành.

Tuy nhiên cũng tồn tại một số khó khăn làm cho Linux chưa thực sự trở thành một hệ điều hành phổ dụng, dưới đây là một số khó khăn điển hình:

- Tuy đã có công cụ hỗ trợ cài đặt, tuy nhiên, việc cài đặt Linux còn tương đối phức tạp và khó khăn. Khả năng tương thích của Linux với một số loại thiết bị phần cứng còn thấp do chưa có các trình điều khiển cho nhiều thiết bị,
- Phần mềm ứng dụng chạy trên nền Linux tuy đã phong phú song so với một số hệ điều hành khác, đặc biệt là khi so sánh với MS Windows, thì vẫn còn có khoảng cách.

Với sự hỗ trợ của nhiều công ty tin học hàng đầu thế giới (IBM, SUN, HP ...) và sự tham gia phát triển của hàng vạn chuyên gia trên toàn thế giới thuộc cộng đồng Linux, các khó khăn của Linux chắc chắn sẽ nhanh chóng được khắc phục.

Chính vì lẽ đó đã hình thành một số nhà cung cấp Linux trên thế giới. Bảng dưới đây là tên của một số nhà cung cấp Linux có tiếng nhất và địa chỉ website của họ.

Đáng chú ý nhất là Red Hat Linux (tại Mỹ) và Red Flag Linux (tại Trung Quốc). Red Hat được coi là lâu đời và tin cậy, còn Red Flag là một công ty Linux của Trung quốc, có quan hệ với cộng đồng Linux Việt nam và chúng ta có thể học hỏi một cách trực tiếp kinh nghiệm cho quá trình đưa Linux vào Việt nam.

Tên công ty	***Địa chỉ website***
Caldera OpenLinux	www.caldera.com
Corel Linux	www.corel.com
Debian GNU/Linux	www.debian.com
Linux Mandrake	www.mandrake.com
Red Hat Linux	www.redhat.com
Red Flag Linux	www.redflag-linux.com
Slackware Linux	www.slackware.com
SuSE Linux	www.suse.com
TurboLinux	www.turbolinux.com

1.2. Sơ bộ về các thành phần của Linux

Hệ thống Linux, được thi hành như một hệ điều hành UNIX truyền thống, gồm shell và ba thành phần (đã dạng mã chương trình) sau đây:

- Nhân hệ điều hành chịu trách nhiệm duy trì các đối tượng trừu tượng quan trọng của hệ điều hành, bao gồm bộ nhớ ảo và quá trình. Các mô đun chương trình trong nhân được đặc quyền trong hệ thống, bao gồm đặc quyền thường trực ở bộ nhớ trong.
- Thư viện hệ thống xác định một tập chuẩn các hàm để các ứng dụng tương tác với nhân, và thi hành nhiều chức năng của hệ thống nhưng không cần có các đặc quyền của mô đun thuộc nhân. Một hệ thống con điển hình được thi hành dựa trên thư viên hệ thống là hệ thống file Linux.
- Tiện ích hệ thống là các chương trình thi hành các nhiệm vụ quản lý riêng rẽ, chuyên biệt. Một số tiện ích hệ thống được gọi ra chỉ một lần để khởi động và cấu hình phương tiện hệ thống, một số tiện ích khác, theo thuật ngữ UNIX được gọi là trình chạy ngầm (daemon), có thể chạy một cách thường xuyên (thường theo chu kỳ), điều khiển các bài toán như hưởng ứng các kết nối mạng mới đến, tiếp nhận yêu cầu logon, hoặc cập nhật các file log.

Tiện ích (hay lệnh) có sẵn trong hệ điều hành (dưới đây tiện ích được coi là lệnh thường trực). Nội dung chính yếu của tài liệu này giới thiệu chi tiết về một số lệnh thông dụng nhất của Linux. Hệ thống file sẽ được giới thiệu trong chương 3. Trong các chương sau có đề cập tới nhiều nội dung liên quan đến nhân và shell, song dưới đây là một số nét sơ bộ về chúng.

1.2.1. Sơ bộ về nhân

Nhân (còn được gọi là hệ lõi) của Linux, là một bộ các môdun chương trình có vai trò điều khiển các thành phần của máy tính, phân phối các tài nguyên cho người dùng (các quá trình người dùng). Nhân chính là cầu nối giữa chương trình ứng dụng với phần cứng.

Người dùng sử dụng bàn phím gõ nội dung yêu cầu của mình và yêu cầu đó được nhân gửi tới shell: Shell phân tích lệnh và gọi các chương trình tương ứng với lệnh để thực hiện.

Một trong những chức năng quan trọng nhất của nhân là giải quyết bài toán lập lịch, tức là hệ thống cần phân chia CPU cho nhiều quá trình hiện thời cùng tồn tại. Đối với Linux, số lượng quá trình có thể lên tới con số hàng nghìn. Với số lượng quá trình đồng thời nhiều như vậy, các thuật toán lập lịch cần phải đủ hiệu quả: Linux thường lập lịch theo chế độ Round Robin (RR) thực hiện việc luân chuyển CPU theo lượng tử thời gian.

Thành phần quan trọng thứ hai trong nhân là hệ thống các môđun chương trình (được gọi là lời gọi hệ thống) làm việc với hệ thống file. Linux có hai cách thức làm việc với các file: làm việc theo byte (kí tự) và làm việc theo khối. Một đặc điểm đáng chú ý là file trong Linux có thể được nhiều người cùng truy nhập tới nên các lời gọi hệ thống làm việc với file cần đảm bảo việc file được truy nhập theo quyền và được chia xẻ cho người dùng.

1.2.2. Sơ bộ về shell

Một số nội dung chi tiết về shell (còn được gọi là ***hệ vỏ***) trong Linux được trình bày trong chương "Lập trình trên shell". Những nội dung trình bày dưới đây cung cấp một cách nhìn sơ bộ về shell và vai trò của nó trong hoạt động chung của hệ điều hành.

Người dùng mong muốn máy tính thực hiện một công việc nào đó thì cần gõ lệnh thể hiện yêu cầu của mình để hệ thống đáp ứng yêu cầu đó. Shell là bộ dịch lệnh và hoạt động như một kết nối trung gian giữa nhân với người dùng: Shell nhận dòng lệnh do người dùng đưa vào; và từ dòng lệnh nói trên, nhân tách ra các bộ phận để nhận được một hay một số lệnh tương ứng với các đoạn văn bản có trong dòng lệnh. Một lệnh bao gồm tên lệnh và tham số: từ đầu tiên là tên lệnh, các từ tiếp theo (nếu có) là các tham số. Tiếp theo, shell sử dụng nhân để khởi sinh một quá trình mới (khởi tạo quá trình) và sau đó, shell chờ đợi quá trình con này tiến hành, hoàn thiện và kết thúc. Khi shell sẵn sàng tiếp nhận dòng lệnh của người dùng, một dấu nhắc shell (còn gọi là dấu nhắc nhập lệnh) xuất hiện trên màn hình.

Linux có hai loại shell phổ biến là: C-shell (dấu nhắc %), Bourne-shell (dấu nhắc $) và một số shell phát triển từ các shell nói trên (chẳng hạn, TCshell - tcsh với dấu nhắc ngầm định > phát triển từ C-shell và GNU Bourne - bash với dấu nhắc bash # phát triển từ Bourne-shell). Dấu mời phân biệt shell nói trên không phải hoàn toàn rõ ràng do Linux cho phép người dùng thay đổi lại dấu nhắc shell nhờ việc thay giá trị các biến môi trường **PS1** và **PS2**. Trong tài liệu này, chúng ta sử dụng kí hiệu "hàng rào #" để biểu thị dấu nhắc shell.

C-shell có tên gọi như vậy là do cách viết lệnh và chương trình lệnh Linux tựa như ngôn ngữ C. Bourne-shell mang tên tác giả của nó là Steven Bourne. Một số lệnh trong C-shell (chẳng hạn lệnh **alias**) không còn có trong Bourne-shell và vì vậy để nhận biết hệ thống đang làm việc với shell nào, chúng ta gõ lệnh:

```
# alias
```

Nếu một danh sách xuất hiện thì shell đang sử dụng là C-shell; ngược lại, nếu xuất hiện thông báo "Command not found" thì shell đó là Bourne-shell.

Lệnh được chia thành 3 loại lệnh:

- Lệnh thường trực (có sẵn của Linux). Tuyệt đại đa số lệnh được giới thiệu trong tài liệu này là lệnh thường trực. Chúng bao gồm các lệnh được chứa sẵn trong shell và các lệnh thường trực khác.
- File chương trình ngôn ngữ máy: chẳng hạn, người dùng viết trình trên ngôn ngữ C qua bộ dịch ***gcc*** (bao gồm cả trình kết nối ***link***) để tạo ra một chương trình trên ngôn ngữ máy.

- File chương trình shell (Shell Scrip).

Khi kết thúc một dòng lệnh cần gõ phím ENTER để shell phân tích và thực hiện lệnh.

1.3. Giới thiệu về sử dụng lệnh trong Linux

Như đã giới thiệu ở phần trên, Linux là một hệ điều hành đa người dùng, đa nhiệm, được phát triển bởi hàng nghìn chuyên gia Tin học trên toàn thế giới nên hệ thống lệnh cũng ngày càng phong phú; đến thời điểm hiện nay Linux có khoảng hơn một nghìn lệnh.

Tuy nhiên chỉ có khoảng vài chục lệnh là thông dụng nhất đối với người dùng. Tài liệu này cũng hạn chế giới thiệu khoảng vài chục lệnh đó. Chúng ta đừng e ngại về số lượng lệnh được giới thiệu chỉ chiếm một phần nhỏ trong tập hợp lệnh bởi vì đây là những lệnh thông dụng nhất và chúng cung cấp một phạm vi ứng dụng rộng lớn, đủ thỏa mãn yêu cầu của chúng ta.

Cũng như đã nói ở trên, người dùng làm việc với máy tính thông qua việc sử dụng trạm cuối: người dùng đưa yêu cầu của mình bằng cách gõ "lệnh" từ bàn phím và giao cho hệ điều hành xử lý.

Khi cài đặt Linux lên máy tính cá nhân thì máy tính cá nhân vừa đóng vai trò trạm cuối, vừa đóng vai trò máy tính xử lý.

Dạng tổng quát của lệnh Linux có thể được viết như sau:

```
# <Tên lệnh> [<các tham số>]
```

trong đó:

- Tên lệnh là một dãy ký tự, không có dấu cách, biểu thị cho một lệnh của Linux hay một chương trình. Người dùng cần hệ điều hành đáp ứng yêu cầu gì của mình thì phải chọn đúng tên lệnh. Tên lệnh là bắt buộc phải có khi gõ lệnh.
- Các tham số có thể có hoặc không có, được viết theo quy định của lệnh mà chúng ta sử dụng, nhằm cung cấp thông tin về các đối tượng mà lệnh tác động tới. Ý nghĩa của các dấu [, <, >,] được giải thích ở phần quy tắc viết lệnh.

Các tham số được phân ra thành hai loại: tham số khóa (sau đây thường dùng là "tùy chọn") và tham số vị trí. Tham số vị trí thường là tên file, thư mục và thường là các đối tượng chịu sự tác động của lệnh. Khi gõ lệnh, tham số vị trí được thay bằng những đối tượng mà người dùng cần hướng tác động tới. Tham số khóa chính là những tham số điều khiển hoạt động của lệnh theo các trường hợp riêng. Trong Linux, tham số khóa thường bắt đầu bởi dấu trừ "**-**" hoặc hai dấu trừ liên tiếp "**--**". Khi gõ lệnh, cũng giống như tên lệnh, tham số khóa phải được viết chính xác như trình bày trong mô tả lệnh. Một lệnh có thể có một số hoặc rất nhiều tham số khóa. Phụ thuộc vào yêu cầu cụ thể của mình, người dùng có thể chọn một hoặc một số các tham số khóa khi gõ lệnh. Trong mô tả lệnh, thường xuất hiện thuật ngữ ***tùy-chọn***. Tùy chọn lệnh (thực chất là ***tham số khóa***) cho phép điều chỉnh hoạt động của lệnh trong Linux, làm cho lệnh có tính phổ dụng cao. Tuỳ chọn lệnh cho phép lệnh có thể đáp ứng ý muốn của người dùng đối với hầu hết (tuy không phải lúc nào cũng vậy) các tình huống đặt ra cho thao tác ứng với lệnh.

- Ký hiệu "[redacted] biểu thị việc gõ phím hết dòng <Enter>. Để kết thúc một yêu cầu, người dùng nhất thiết phải gõ phím "[redacted]

Ví dụ, khi người dùng gõ lệnh xem thông tin về các file:

```
# ls -l g[redacted]
```

trong lệnh này:

- **ls** là tên lệnh thực hiện việc đưa danh sách các tên file/ thư mục con trong một thư mục,
- **-l** là tham số khóa, cho biết yêu cầu xem đầy đủ thông tin về các đối tượng hiện ra. Chú ý, trong tham số khóa chữ cái (chữ "l") phải đi ngay sau dấu trừ "-". Tương ứng với lệnh **ls** còn có các tham số khóa **-a**, **-L**, ... và chúng cũng là các tùy chọn lệnh. Trong một số tham số khóa có nhiều chữ cái thay cho một dấu "**-**" là hai dấu "**--**" ở đầu tham số. Ví dụ, như trường hợp tham số **--file** của lệnh **date**.
- g* là tham số vị trí chỉ rõ người dùng cần xem thông tin về các file có tên gọi bắt đầu là chữ cái "g".

Trong tài liệu này, quy ước rằng khi viết một lệnh (trong mô tả lệnh và gõ lệnh) thì không cần phải viết dấu "↵" ở cuối dòng lệnh đó, song luôn ghi nhớ rằng phím ENTER ("↵" là bắt buộc khi gõ lệnh.

☞ ***Lưu ý:***

- Linux (và UNIX nói chung) được xây dựng trên ngôn ngữ lập trình C, vì vậy khi gõ lệnh phải phân biệt chữ thường với chữ hoa. Ngoại trừ một số ngoại lệ, trong Linux chúng ta thấy phổ biến là:
  - ❖ Các tên lệnh là chữ thường,
  - ❖ Một số tham số có thể là chữ thường hoặc chữ hoa (ví dụ, trong lệnh **date** về thời gian hệ thống thì hai tham số **-r** và **-R** có ý nghĩa hoàn toàn khác nhau). Tên các biến môi trường cũng thường dùng chữ hoa.
- Trong tài liệu này, tại những dòng văn bản diễn giải, sử dụng cách viết tên lệnh, các tham số khóa bằng kiểu chữ không chân, đậm như **date**, **-R**, **-r** ...
- Linux phân biệt siêu người dùng (tiếng Anh là superuser hoặc root, còn được gọi là ***người quản trị*** hay ***người dùng tối cao*** hoặc ***siêu người dùng***) với người dùng thông thường. Trong tập hợp lệnh của Linux, có một số lệnh mà chỉ siêu người dùng mới được phép sử dụng còn người dùng thông thường thì không được phép (ví dụ như lệnh **adduser** thực hiện việc bổ sung thêm người dùng). Mặt khác trong một số lệnh, với một số tham số khóa thì chỉ siêu người dùng được phép dùng, còn với một số tham số khác thì mọi người dùng đều được phép (ví dụ như lệnh **passwd** thay đổi mật khẩu người dùng).
- Một dòng lệnh có thể có nhiều hơn một lệnh, trong đó lệnh sau được ngăn cách bởi với lệnh đi ngay trước bằng dấu ";" hoặc dấu "|". Ví dụ về một số dòng lệnh dạng này:

```
# ls -l; date
# head Filetext | sort >temp
```

- Sau khi người dùng gõ xong dòng lệnh, shell tiếp nhận dòng lệnh này và phân tích nội dung văn bản của lệnh. Nếu lệnh được gõ đúng thì nó được thực hiện; ngược lại, trong trường hợp có sai sót khi gõ lệnh thì shell thông báo về sai sót và dấu nhắc shell lại hiện ra để chờ lệnh tiếp theo của người dùng. Về phổ biến, nếu như sau khi người dùng gõ lệnh, không thấy thông báo sai sót hiện ra thì có nghĩa lệnh đã được thực hiện một cách bình thường.

Trước khi đi vào nội dung chi tiết các lệnh thông dụng, chúng ta xem xét về một số quy định dùng trong mô tả lệnh được trình bày trong tài liệu này.

1.3.1. Các quy ước khi viết lệnh

Trong tài liệu này, các lệnh được trình bày theo một bộ quy tắc cú pháp nhất quán. Bộ quy tắc này cho phép phân biệt trong mỗi lệnh các thành phần nào là bắt buộc phải có, các thành phần nào có thể có hoặc không ... Dưới đây là nội dung của các quy tắc trong bộ quy tắc đó.

- Tên lệnh là bắt buộc, phải là từ đầu tiên trong bất kỳ lệnh nào, phải được gõ đúng như khi mô tả lệnh.
- Tên khái niệm được nằm trong cặp dấu ngoặc quan hệ (< và >) biểu thị cho một lớp đối tượng và là tham số bắt buộc phải có. Khi gõ lệnh thì tên khái niệm (có thể được coi là "tham số hình thức") phải được thay thế bằng một từ (thường là tên file, tên thư mục ... và có thể được coi là "tham số thực sự") để chỉ đối tượng liên quan đến thao tác của lệnh.

Ví dụ, mô tả cú pháp của lệnh **more** xem nội dung file là

```
# more <file>
```

thì từ **more** là tên lệnh, còn <file> là tham số trong đó ***file*** là tên khái niệm và là tham số bắt buộc phải có. Lệnh này có tác động là hiện lên màn hình theo cách thức cuộn nội dung của file với tên đã chỉ trong lệnh.
Để xem nội dung file có tên là ***temp***, người dùng gõ lệnh:

```
# more temp
```

Như vậy, tên lệnh **more** được gõ đúng như mô tả cú pháp (cả nội dung và vị trí) còn "***file***" đã được thay thế bằng từ "***temp***" là tên file mà người dùng muốn xem nội dung.

- Các bộ phận nằm giữa cặp dấu ngoặc vuông [và] là có thể gõ hoặc không gõ cũng được.

Ví dụ, cú pháp của lệnh **halt** là

```
# halt [tùy-chọn]
```

Với các tùy chọn là **-w**, **-n, -d**, **-f**, **-i** mã mỗi tùy chọn cho một cách thức hoạt động khác nhau của lệnh **halt.** Lệnh **halt** có tác động chính là làm ngừng hoạt động của hệ điều hành, tuy nhiên khi người dùng muốn có một cách hoạt động nào đó của lệnh này thì sẽ chọn một (hoặc một số) tuỳ chọn lệnh tương ứng. Một số cách gõ lệnh **halt** của người dùng như sau đây là đúng cú pháp:

```
# halt
# halt -w
# halt -n
# halt -f
```

- Các giá trị có trong cặp | và | trong đó các bộ phận cách nhau bằng dấu sổ đứng "|" cho biết cần chọn một và chỉ một trong các giá trị nằm giữa hai dấu ngoặc đó.

Ví dụ, khi giới thiệu về tùy chọn lệnh của lệnh **tail** xem phần cuối nội dung của file, chúng ta thấy:

```
-f, --follow[={tên | đặc tả}]
```

Như vậy, sau tham số khóa **--follow**, nếu xuất hiện thêm dấu bằng "=" thì phải có hoặc ***tên*** hoặc ***đặc tả***. Đây là trường hợp các chọn lựa "loại trừ nhau".

- Dấu ba chấm ... thể hiện việc lặp lại thành phần cú pháp đi ngay trước dấu này, việc lặp lại đó có thể từ không đến nhiều lần (không kể chính thành phần cú pháp đó). Cách thức này thường được dùng với các tham số như tên file.

Ví dụ, mô tả lệnh **chown** như sau:

```
chown [tùy-chọn] <chủ>[,[nhóm]]<file>...
```

Như vậy trong lệnh **chown** có thể không có hoặc có một số tùy chọn lệnh và có từ một đến nhiều tên file.

- Các bộ phận trong mô tả lệnh, nếu không nằm trong các cặp dấu [ ], <>, { } thì khi gõ lệnh thực sự phải gõ y đúng như khi mô tả (chú ý, quy tắc viết tên lệnh là một trường hợp riêng của quy tắc này).
- Việc kết hợp các dấu ngoặc với nhau cho phép tạo ra cách thức sử dụng quy tắc tổ hợp các tham số trong lệnh. Ví dụ, lệnh **more** bình thường có cú pháp là:

```
# more <file>
```

có nghĩa là thay <file> bằng tên file cần xem nội dung, nếu kết hợp thêm dấu ngoặc vuông [và], tức là có dạng sau (chính là dạng tổng quát của lệnh **more**):

```
# more [<file>]
```

thì <file> nói chung phải có trong lệnh **more**, tuy nhiên trong một số trường hợp có thể bỏ qua tham số file.

☞ ***Lưu ý:***

- Đối với nhiều lệnh, cho phép người dùng gõ ***tham số khóa kết hợp*** tương ứng với *tùy_chọn* trong mô tả lệnh. Tham số khóa kết hợp được viết theo cách **-<xâu-kí-tự>**, trong đó xâu-kí-tự gồm các chữ cái trong tham số khóa. Ví dụ, trong mô tả lệnh in lịch **cal**:

```
cal [tùy-chọn] [tháng [năm] ]
```

có ba tham số khóa là **-m, -j, -y**. Khi gõ lệnh có thể gõ một tổ hợp nào đó từ ba tham số khóa này để được tình huống sử dụng lệnh theo ý muốn. Chẳng hạn, nếu gõ lệnh

```
cal -mj 3
```

thì lệnh **cal** thực hiện theo điều khiển của hai tham số khóa **-m** (chọn Thứ Hai là ngày đầu tuần, thay vì cho ngầm định là Chủ Nhật) và **-j** (hiển thị ngày trong tháng dưới dạng số ngày trong năm kể từ đầu năm). Vì vậy, khi viết [tùy-chọn] trong mô tả lệnh biểu thị cả việc sử dụng từng tùy chọn, nhiều tùy chọn hoặc kết hợp các tuỳ chọn.

- Trong một số lệnh, có hai tham số khóa cùng tương ứng với một tình huống thực hiện lệnh, trong đó một tham số gồm một kí tự còn tham số kia lại là một từ. Tham số dài một từ là tham số chuẩn của lệnh, còn tham số một kí tự là cách viết ngắn gọn. Tham số chuẩn dùng được trong mọi Linux và khi gõ phải có đủ kí tự trong từ.

  Ví dụ, khi mô tả lệnh date có tùy chọn:

- -d, --date=STRING

  như vậy hai tham số **-d** và **--date=STRING** có cùng ý nghĩa.

Ngoài những quy ước trên đây, người dùng đừng quên một quy định cơ bản là cần phân biệt chữ hoa với chữ thường khi gõ lệnh.

1.3.3. Làm đơn giản thao tác gõ lệnh

Việc sử dụng bàn phím để nhập lệnh tuy không phải là một công việc nặng nề, song Linux còn cho phép người dùng sử dụng một số cách thức để thuận tiện hơn khi gõ lệnh. Một số trong những cách thức đó là:

- Sử dụng việc khôi phục dòng lệnh,
- Sử dụng các phím đặc biệt,
- Sử dụng các kí hiệu thay thế và phím <Tab>,
- Sử dụng thay thế **alias**,
- Sử dụng chương trình lệnh.

Cách thức sử dụng chương trình lệnh (shell script) sẽ được giới thiệu chi tiết trong các chương sau. Dưới đây, chúng ta xem xét cách thức sử dụng việc khôi phục dòng lệnh, phím đặc biệt và kí hiệu thay thế.

Cơ chế khôi phục dòng lệnh

Linux cung cấp một cách thức đặc biệt là khả năng khôi phục lệnh. Tại dấu nhắc shell: Người dùng sử dụng các phím mũi tên lên/xuống ([redacted] trên bàn phím để nhận lại các dòng lệnh đã được đưa vào trước đây tại dấu nhắc shell, chọn một trong các dòng lệnh đó và biên tập lại nội dung dòng lệnh theo đúng yêu cầu mới của mình.

Ví dụ, người dùng vừa gõ xong dòng lệnh:

```
# ls -l tenfile*
```

sau đó muốn gõ lệnh **ls -l tentaptin** thì tại dấu nhắc của shell, người dùng sử dụng các phím di chuyển lên ([redacted] hoặc xuống ([redacted] để nhận được:

```
# ls -l tenfile*
```

dùng các phím tắt để di chuyển, xoá kí tự (xem phần sau) để có được:

```
# ls -l ten
```

và gõ tiếp các kí tự "taptin" để nhận được:

```
# ls -l tentaptin
```

chính là kết quả mong muốn.

Trong trường hợp số lượng kí tự thay thế là rất ít so với số lượng kí tự của toàn dòng lệnh thì hiệu quả của cách thức này rất cao.

☞ ***Lưu ý:***

- Việc nhấn liên tiếp các phím di chuyển lên ([redacted] hoặc xuống ([redacted] cho phép người dùng nhận được các dòng lệnh đã gõ từ trước mà không chỉ dòng lệnh mới được gõ. Cách thức này tương tự với cách thức sử dụng tiện ích DOSKEY trong hệ điều hành MS-DOS.

Một số phím đặc biệt khi gõ lệnh

Khi người dùng gõ lệnh có thể xảy ra một số tình huống như sau:

- Dòng lệnh đang gõ có chỗ sai sót, không đúng theo yêu cầu của người dùng vì vậy cần phải sửa lại đôi chút nội dung trên dòng lệnh đó. Trong trường hợp đó cần sử dụng các phím đặc biệt (còn gọi là phím viết tắt hay phím tắt) để di chuyển, xoá bỏ, bổ sung vào nội dung dòng lệnh.
- Sau khi sử dụng cách thức khôi phục dòng lệnh, chúng ta nhận được dòng lệnh tương tự với lệnh cần gõ và sau đó sử dụng các phím tắt để hoàn thiện lệnh.

Dưới đây giới thiệu các phím tắt và ý nghĩa của việc sử dụng chúng:

- Nhấn phím [redacted] để di chuyển con trỏ sang bên phải một vị trí
- Nhấn phím [redacted] để di chuyển con trỏ sang bên trái một vị trí
- Nhấn phím <ESC-BACKSPACE> để xoá một từ bên trái con trỏ
- Nhấn phím <ESC-D> để xoá một từ bên phải con trỏ
- Nhấn phím <ESC-F> để di chuyển con trỏ sang bên phải một từ
- Nhấn phím <ESC-B> để di chuyển con trỏ sang bên trái một từ
- Nhấn phím <CTRL-A> để di chuyển con trỏ về đầu dòng lệnh
- Nhấn phím <CTRL-E> để di chuyển con trỏ về cuối dòng
- Nhấn phím <CTRL-U> để xóa dòng lệnh

Có thể dùng phím <ALT> thay cho phím <ESC>.

Các kí hiệu mô tả nhóm file và phím <Tab>

*Khi gõ lệnh thực sự nhiều trường hợp người dùng mong muốn một tham số trong lệnh không chỉ xác định một file mà lại liên quan đến một nhóm các file mà tên gọi của các file trong nhóm có chung một tính chất nào đó. Trong những trường hợp như vậy, người dùng cần sử dụng các kí hiệu mô tả nhóm file (wildcards), chúng ta gọi là kí hiệu mô tả nhóm (còn được gọi là kí hiệu thay thế). Người ta sử dụng các kí tự *, ? và cặp hai dấu [và] để mô tả nhóm file. Các kí tự này mang ý nghĩa như sau khi viết vào tham số tên file thực sự:*

- "*" : là ký tự mô tả nhóm gồm mọi xâu kí tự (thay thế mọi xâu). Mô tả này cho một nhóm lớn nhất trong ba mô tả.
- "?" : mô tả nhóm gồm mọi xâu với độ dài không quá 1 (thay thế một kí tự). Nhóm này là tập con của nhóm đầu tiên (theo kí tự "*").
- [xâu-kí-tự] : mô tả nhóm gồm mọi xâu có độ dài 1 là mỗi kí tự thuộc xâu nói trên. Mô tả này cho một nhóm có lực lượng bé nhất trong ba mô tả. Nhóm này là tập con của nhóm thứ hai (theo kí tự "?"). Khi gõ lệnh phải gõ cả hai dấu [và]. Một dạng khác của mô tả nhóm này là [<kí_tự_1>-<kí_tự_2>] nghĩa là giữa cặp dấu ngoặc có ba kí tự trong đó kí tự ở giữa là dấu nối (dấu -) thì cách viết này tương đương với việc liệt kê mọi kí tự từ <kí_tự_1> đến <kí_tự_2>. Chẳng hạn, cách viết [a-d] tương đương với cách viết [abcd].

Ví dụ, giả sử khi muốn làm việc với tất cả các file trong một thư mục nào đó, người dùng gõ * thay thế tham số ***file*** thì xác định được các tên file sau (chúng ta viết bốn tên file trên một dòng):

info-dir	**initlog.conf**	**inittab**	**lynx.cfg**
mail.rc	**mailcap**	**minicom.users**	**motd**

mtab **mtools.conf** **services** **shadow**
shadow- **shells** **smb.conf** **sysctl.conf**
syslog.conf **temp** **termcap** **up2date.conf**
temp **termcap**

Nếu người dùng gõ s* (để chỉ các tên có chữ cái đầu là s) thay thế tham số ***file*** thì xác định được các tên file sau:

shadow **shadow-** **shells** **sysctl.conf**
syslog.conf

Nếu người dùng gõ [si]* (để chỉ các tên có chữ cái đầu là s hoặc i, chú ý dùng cả hai kí tự [và]) thay thế tham số ***file*** thì xác định các tên file sau:

info-dir **initlog.conf** **inittab** **services**
shadow **shadow-** **shells** **smb.conf**
sysctl.conf **syslog.conf**

☞ ***Lưu ý:***

- Như vậy, Linux (và UNIX nói chung) không chỉ sử dụng hai kí tự mô tả nhóm * và ? mà còn có cách thức sử dụng cặp kí tự [và].
- Cần phân biệt cặp dấu [và] được sử dụng khi người dùng gõ lệnh có ý nghĩa hoàn toàn khác với ý nghĩa của chúng khi được sử dụng trong mô tả lệnh.

Hơn thế nữa, Linux còn cung cấp cho người dùng cách thức sử dụng phím <**TAB**> để hoàn thành nốt tên file (tên thư mục) trong lệnh. Ví dụ, khi chúng ta gõ dòng lệnh

```
# ls /u<TAB>local<TAB>b<TAB>
```

thì nó cũng tương đương như gõ dòng lệnh (và đây chính là nội dung xuất hiện tại dấu nhắc shell):

```
# ls /usr/local/bin
```

với điều kiện trong thư mục **/usr** chỉ có thư mục **local** được bắt đầu bởi chữ "l" và trong thư mục **local** cũng chỉ có thư mục **bin** được bắt đầu bởi chữ "b".
Trong trường hợp nếu như một kí tự chưa đủ xác định, người dùng cần gõ thêm kí tự tiếp theo trong tên file (tên thư mục) và nhấn phím <**TAB**> để hoàn thành dòng lệnh.

1.3.4. Tiếp nối dòng lệnh

Như đã lưu ý trên đây, một dòng lệnh có thể gồm một hoặc một số lệnh, mặt khác tham số của lệnh có thể là rất dài không thể trong khuôn khổ của một dòng văn bản được. Khi gõ lệnh, nếu dòng lệnh quá dài, Linux cho phép ngắt dòng lệnh xuống dòng dưới bằng cách thêm kí tự báo hiệu chuyển dòng "\" tại cuối dòng; trong trường hợp đó, kí tự "\" phải là ký tự cuối cùng thuộc dòng lệnh trước.

Ví dụ,

```
# cd vsd\
thumuc
```

thì dòng thứ hai là phần tiếp theo của dòng thứ nhất và kết hợp cả hai dòng này thực chất là một dòng lệnh Linux.

1.4. Trang Man

Chúng ta có thể nói rằng Linux là một hệ điều hành rất phức tạp với hàng nghìn lệnh và mỗi lệnh lại có thể có tới vài hoặc vài chục tình huống sử dụng do chúng cho phép có nhiều tùy chọn lệnh. Để thuộc hết được nội dung tất cả các lệnh của Linux là một điều hết sức khó khăn, có thể nói là không thể. Linux cho phép người dùng sử dụng cách thức gọi trang Man để có được các thông tin đầy đủ giới thiệu nội dung các lệnh. Dưới đây là một số nội dung về cách thức sử dụng trang Man.

"Man" là từ viết tắt của "manual", được coi là tài liệu trực tuyến trong Linux đã lưu trữ toàn bộ các lệnh có sẵn với các thông tin tham khảo khá đầy đủ cho phép người dùng có thể mở ra để nhận được trợ giúp.

Để mở trang Man của một lệnh, chúng ta sử dụng lệnh **man** của Linux và gõ:

```
# man <tên-lệnh>
```

Nội dung của trang Man tuy không phải là quá khó hiểu, song để hiểu hết được nó cũng đòi hỏi không ít thời gian. Tuy vậy, nếu quên nội dung một lệnh nào đó thì cách tốt nhất là hãy sử dụng trang Man.

Cấu trúc chung của một trang Man như sau:

COMMAND(1) Linux Programmer's Manual COMMAND(1)

NAME
tên lệnh - khái quát tác dụng của lệnh

SYNOPSIS
cú pháp của lệnh

DESCRIPTION
mô tả cụ thể hơn về tác dụng của lệnh

OPTIONS
liệt kê các tùy chọn lệnh và tác dụng của chúng

FILES
liệt kê các file mà lệnh sử dụng hoặc tham chiếu đến

SEE ALSO
liệt kê các lệnh, các tài liệu, ..., có liên quan đến lệnh

REPORTING BUGS
địa chỉ liên hệ nếu gặp lỗi khi sử dụng lệnh

AUTHOR
tên tác giả của lệnh

Người dùng thậm chí không nhớ chính xác tên lệnh. Linux còn có một cách thức hỗ trợ người dùng có thể nhanh chóng tìm được lệnh cần sử dụng trong trường hợp chỉ nhớ những chữ cái đầu của tên lệnh, đó là cách thức sử dụng phím TAB. Trong cách thức này, người dùng chỉ cần nhớ một số chữ cái đầu tiên của tên lệnh.

Có thể trình bày cách thức đó theo cú pháp sau đây:

```
# <dãy-chữ-cái><TAB><TAB>
```

Trong đó dãy-chữ-cái có từ một đến một vài chữ cái thuộc phần đầu của tên lệnh. Chú ý rằng, các chữ cái và hai phím <TAB> phải được gõ liên tiếp nhau.

Kết hợp cách thức này với cách thức sử dụng lệnh **man** (với sự phong phú về tùy chọn của lệnh **man**) nhận được một cách thức khá tuyệt vời trợ giúp người dùng.

Ví dụ, muốn sử dụng lệnh **history** nhưng lại không nhớ chính xác tên lệnh được viết ra như thế nào mà chỉ nhớ nó được bắt đầu bởi chữ **h**, hãy gõ chữ **h** đó tại dấu nhắc shell và nhấn phím TAB hai lần, sẽ thấy một danh sách các lệnh có chữ cái đầu tiên là **h** được hiện ra trên màn hình:

```
# h<TAB><TAB>
h2ph          hboot       help          hexdump     history
hostname      htdigest    h2xs  hcc     helpme      hf77
hltest        hoststat    htpasswd      halt        hcp
helptool      hinotes     host          hpcdtoppm   hash
head          hexbin      hipstopgm     hostid      hpftodit
```

Như vậy, tất cả các lệnh có tên bắt đầu với chữ **h** được hiển thị trên màn hình và cho phép người dùng có thể xác định được lệnh cần quan tâm.

Trường hợp tồn tại một số lượng lớn các lệnh có cùng chữ cái đầu tiên mà người dùng đã gõ, thay vì hiện hết mọi tên lệnh, hệ điều hành cho ra một thông báo hỏi người dùng có muốn xem toàn bộ các lệnh đó hay không. Người dùng đáp ứng thông báo đó tuỳ theo ý muốn của mình.

Ví dụ, khi người dùng gõ nội dung như sau:

```
# p<TAB><TAB>
```

thì hệ thống đáp lại là:

```
There are 289 possibilities. Do you really wish to see them all? (y or n)
```

Người dùng gõ phím "y" nếu muốn xem, hoặc gõ "n" nếu bỏ qua.

Người dùng có thể gõ nhiều hơn một chữ cái ở đầu tên lệnh và điều đó cho phép giảm bớt số tên lệnh mà hệ thống tìm được và hiển thị. Chẳng hạn, khi biết hai chữ cái đầu là "pw" và người dùng gõ:

```
# pw<TAB><TAB>
```

thì hệ thống sẽ hiện ra danh sách các tên lệnh bắt đầu bởi "pw":

```
pwck          pwconv      pwd           pwdb_chkpwd     pwunconv
```

Trong trường hợp này, người dùng sẽ nhận biết được tên lệnh đang cần tìm thuận tiện hơn.

CHƯƠNG 2. THAO TÁC VỚI HỆ THỐNG

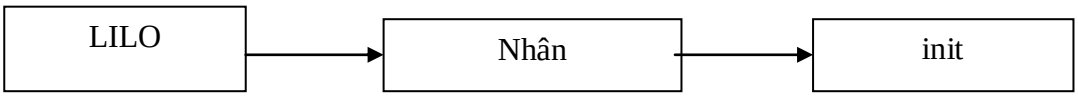
2.1. Quá trình khởi động Linux

Trong phần này, chúng ta xem xét sơ bộ quá trình khởi động hệ điều hành Linux.

Một trong những cách thức khởi động Linux phổ biến nhất là cách thức do chương trình LILO (LInux LOader) thực hiện. Chương trình LILO được nạp lên đĩa của máy tính khi cài đặt hệ điều hành Linux. LILO được nạp vào Master Boot Record của đĩa cứng hoặc vào Boot Sector tại phân vùng khởi động (trên đĩa cứng hoặc đĩa mềm). Giả sử máy tính của chúng ta đã cài đặt Linux và sử dụng LILO để khởi động hệ điều hành. LILO thích hợp với việc trên máy tính được cài đặt một số hệ điều hành khác nhau và theo đó, LILO còn cho phép người dùng chọn lựa hệ điều hành để khởi động.

Giai đoạn khởi động Linux tùy thuộc vào cấu hình LILO đã được lựa chọn trong quá trình cài đặt Linux. Trong tình huống đơn giản nhất, Linux được khởi động từ đĩa cứng hay đĩa mềm khởi động.

Quá trình khởi động Linux có thể được mô tả theo sơ đồ sau:



Theo sơ đồ này, LILO được tải vào máy để thực hiện mà việc đầu tiên là đưa nhân vào bộ nhớ trong và sau đó tải chương trình init để thực hiện việc khởi động Linux.

Nếu cài đặt nhiều phiên bản Linux hay cài Linux cùng các hệ điều hành khác (trong các trường hợp như thế, mỗi phiên bản Linux hoặc hệ điều hành khác được gán ***nhãn*** - label để phân biệt), thì thông báo sau đây được LILO đưa ra:

LILO boot:

cho phép nhập xâu là nhãn của một trong những hệ điều hành hiện có trên máy để khởi động nó. Tại thời điểm đó, người dùng cần gõ nhãn của hệ điều hành cần khởi động vào, ví dụ, gõ

LILO boot: linux

nếu chọn khởi động để làm việc trong Linux, hoặc gõ

LILO boot: dos

nếu chọn khởi động để làm việc trong MS-DOS, Windows.

☞ ***Lưu ý:***

- Nếu chúng ta không nhớ được nhãn của hệ điều hành có trong máy để chọn, hãy gõ phím <TAB> để được LILO cho biết nhãn của các hệ điều hành.

  LILO boot: <TAB>

  sẽ hiện ra danh sách các nhãn (ví dụ như): **`linux dos`** ...
  và hiện lại thông báo nói trên để ta gõ nhãn của hệ điều hành.

- LILO cũng cho phép đặt chế độ chọn ngầm định hệ điều hành để khởi động mà theo đó nếu chúng ta không có tác động gì sau thông báo chọn hệ điều hành thì LILO sẽ tự động chọn hệ điều hành ngầm định ra để khởi động. Nếu chúng ta không can thiệp vào các file tương ứng của trình LILO thì hệ điều hành Linux là hệ điều hành ngầm định.

Giả sử Linux đã được chọn để khởi động. Khi ***init*** thực hiện, chúng ta sẽ thấy một chuỗi (khoảng vài chục) dòng thông báo cho biết hệ thống phần cứng được Linux nhận diện và thiết lập cấu hình cùng với tất cả trình điều khiển phần mềm được nạp khi khởi động. Quá trình ***init*** là quá trình khởi thủy, là cha của mọi quá trình. Tại thời điểm khởi động hệ thống ***init*** thực hiện vai trò đầu tiên của mình là chạy chương trình shell trong file ***/etc/inittab*** và các dòng thông báo trên đây chính là kết quả của việc chạy chương trình shell đó. Sau khi chương trình shell trên được thực hiện xong, bắt đầu quá trình người dùng đăng nhập (login) vào hệ thống.

2.2. Thủ tục đăng nhập và các lệnh thoát khỏi hệ thống

2.2.1. Đăng nhập

Sau khi hệ thống Linux (lấy Red Hat 6.2 làm ví dụ) khởi động xong, trên màn hình xuất hiện những dòng sau:

Ret Hat Linux release 6.2 (Zoot)
Kernel 2.2.14-5.0 on an i686
May1 login:

Dòng thứ nhất và dòng thứ hai cho biết loại phiên bản Linux, phiên bản của nhân và

Chúng ta có thể thay đổi các dòng hiển thị như trình bày trên đây bằng cách sửa đổi file ***/etc/rc.d/rc.local*** *như sau:*

Thay đoạn chương trình

```
echo "" > /etc/issue
echo "$R" >> /etc/issue
echo "Kernel $(uname -r) on $a $SMP$(uname -m)" >> /etc/issue
cp -f /etc/issue /etc/issue.net
echo >> /etc/issue
```

thành

```
echo "" > /etc/issue
echo "Thông báo muốn hiển thị" >> /etc/issue
```

ví dụ sửa thành:

```
echo "" > /etc/issue
echo "This is my computer" >> /etc/issue
```

thì trên màn hình đăng nhập sẽ có dạng sau:

This is my computer
hostname login:

---

kiến trúc phần cứng có trên máy, dòng thứ ba là dấu nhắc đăng nhập để người dùng thực hiện việc đăng nhập. Chú ý là các dòng trên đây có thể thay đổi chút ít tùy thuộc vào phiên bản Linux.

Tại dấu nhắc đăng nhập, hãy nhập tên người dùng (còn gọi là tên đăng nhập): đây là tên kí hiệu đã cung cấp cho Linux nhằm nhận diện một người dùng cụ thể. Tên đăng nhập ứng với mỗi người dùng trên hệ thống là duy nhất, kèm theo một mật khẩu đăng nhập.

May1 login: root
Password:

Khi nhập xong tên đăng nhập, hệ thống sẽ hiện ra thông báo hỏi mật khẩu và di chuyển con trỏ xuống dòng tiếp theo để người dùng nhập mật khẩu. Mật khẩu khi được nhập sẽ không hiển thị trên màn hình và chính điều đó giúp tránh khỏi sự "nhòm ngó" của người khác.

Nếu nhập sai tên đăng nhập hoặc mật khẩu, hệ thống sẽ đưa ra một thông báo lỗi:

May1 login: root
Password:
Login incorrect

Máy1 login:

Nếu đăng nhập thành công, người dùng sẽ nhìn thấy một số thông tin về hệ thống, một vài tin tức cho người dùng... Lúc đó, dấu nhắc shell xuất hiện để người dùng bắt đầu phiên làm việc của mình.

May1 login: root
Password:
Last login: Fri Oct 27 14:16:09 on tty2
Root[may1 /root]#

Dãy kí tự trong dòng cuối cùng chính là ***dấu nhắc shell***. Trong dấu nhắc này, ***root*** là tên người dùng đăng nhập, ***may1*** là tên máy và **/root** tên thư mục hiện thời (vì đây là người dùng root). Khi dấu nhắc shell xuất hiện trên màn hình thì điều đó có nghĩa là hệ điều hành đã sẵn sàng tiếp nhận một yêu cầu mới của người dùng.

Dấu nhắc shell có thể khác với trình bày trên đây (Mục 2.7 cung cấp cách thay đổi dấu nhắc shell), nhưng có thể hiểu nó là chuỗi kí tự bắt đầu một dòng có chứa trỏ chuột và luôn xuất hiện mỗi khi hệ điều hành hoàn thành một công việc nào đó.

2.2.2. Ra khỏi hệ thống

Để kết thúc phiên làm việc người dùng cần thực hiện thủ tục ra khỏi hệ thống. Có rất nhiều cách cho phép thoát khỏi hệ thống, ở đây chúng ta xem xét một số cách thông dụng nhất.

- Cách đơn giản nhất để đảm bảo thoát khỏi hệ thống đúng đắn là nhấn tổ hợp phím **CTRL**+**ALT**+**DEL**. Khi đó, trên màn hình sẽ hiển thị một số thông báo của hệ thống và cuối cùng là thông báo thoát trước khi tắt máy. Cần chú ý là: Nếu đang làm việc trong môi trường X Window System, hãy nhấn tổ hợp phím **CTRL**+**ALT**+**BACKSPACE** trước rồi sau đó hãy nhấn **CTRL**+**ALT**+**DEL**.
- Cách thứ hai là sử dụng lệnh **shutdown** với cú pháp như sau:

```
shutdown [tùy-chọn] <time> [cảnh-báo]
```

Lệnh này cho phép dừng tất cả các dịch vụ đang chạy trên hệ thống.

Các tùy-chọn của lệnh này như sau:

- **-k** : không thực sự shutdown mà chỉ cảnh báo.
- **-r** : khởi động lại ngay sau khi shutdown.
- **-h** : tắt máy thực sự sau khi shutdown.
- **-f** : khởi động lại nhanh và bỏ qua việc kiểm tra đĩa.

- **-F** : khởi động lại và thực hiện việc kiểm tra đĩa.
- **-c** : bỏ qua không chạy lệnh shutdown. Trong tùy chọn này không thể đưa ra tham số thời gian nhưng có thể đưa ra thông báo giải thích trên dòng lệnh gửi cho tất cả các người dùng.
- **-t *số-giây*** *:* qui định init(8) chờ khoảng thời gian số-giây tạm dừng giữa quá trình gửi cảnh báo và tín hiệu kill, trước khi chuyển sang một mức chạy khác.

và hai tham số vị trí còn lại:

- **time** : đặt thời điểm shutdown. Tham số time có hai dạng. Dạng tuyệt đối là gg:pp (gg: giờ trong ngày, pp: phút) thì hệ thống sẽ shutdown khi đồng hồ máy trùng với giá trị tham số. Dạng tương đối là +<số> là hẹn sau thời khoảng <số> phút sẽ shutdown; coi shutdown lập tức tương đương với +0.
- **cảnh-báo** : thông báo gửi đến tất cả người dùng trên hệ thống. Khi lệnh thực hiện tất cả các máy người dùng đều nhận được cảnh báo.

Ví dụ, khi người dùng gõ lệnh:

```
shutdown +1 Sau mot phut nua he thong se shutdown!
```

trên màn hình của tất cả người dùng xuất hiện thông báo "Sau mot phut nua he thong se shutdown! " và sau một phút thì hệ thống shutdown thực sự.

- Cách thứ ba là sử dụng lệnh **halt** với cú pháp như sau:

```
halt [tùy-chọn]
```

Lệnh này tắt hẳn máy.
Các tuỳ chọn của lệnh **halt**:

- **-w** : không thực sự tắt máy nhưng vẫn ghi các thông tin lên file /var/log/wtmp (đây là file lưu trữ danh sách các người dùng đăng nhập thành công vào hệ thống).
- **-d** : không ghi thông tin lên file /var/log/wtmp. Tùy chọn -n có ý nghĩa tương tự song không tiến hành việc đồng bộ hóa.
- **-f** : thực hiện tắt máy ngay mà không thực hiện lần lượt việc dừng các dịch vụ có trên hệ thống.
- **-i** : chỉ thực hiện dừng tất cả các dịch vụ mạng trước khi tắt máy.

Chúng ta cần nhớ rằng, nếu thoát khỏi hệ thống không đúng cách thì dẫn đến hậu quả là một số file hay toàn bộ hệ thống file có thể bị hư hỏng.

☞ ***Lưu ý:***

- Có thể sử dụng lệnh **exit** để trở về dấu nhắc đăng nhập hoặc kết thúc phiên làm việc bằng lệnh **logout**.

2.2.3. Khởi động lại hệ thống

Ngoài việc thoát khỏi hệ thống nhờ các cách thức trên đây (ấn tổ hợp ba phím Ctrl+Alt+Del, dùng lệnh **shutdown** hoặc lệnh **halt**), khi cần thiết (chẳng hạn, gặp phải tình huống một trình ứng dụng chạy quẩn) có thể khởi động lại hệ thống nhờ lệnh **reboot**.
Cú pháp lệnh **reboot**:

```
reboot [tùy-chọn]
```

Lệnh này cho phép khởi động lại hệ thống. Nói chung thì chỉ siêu người dùng mới được phép sử dụng lệnh **reboot**, tuy nhiên, nếu hệ thống chỉ có duy nhất một người dùng đang làm việc thì lệnh **reboot** vẫn được thực hiện song hệ thống đòi hỏi việc xác nhận mật khẩu. Các tùy chọn của lệnh **reboot** như sau là **-w**, **-d**, **-n**, **-f**, **-i** có ý nghĩa tương tự như trong lệnh **halt**.

2.2.4. Khởi động vào chế độ đồ hoạ

Linux cho phép nhiều chế độ khởi động, những chế độ này được liệt kê trong file /etc/inittab. Dưới đây là nội dung của file này:

```
# inittab This file describes how the INIT process should set up
# the system in a certain run-level.
#
# Author: Miquel van Smoorenburg, <miquelsừdrinkel.nl.mugnet.org>
# Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this) - Đây là chế độ dừng hoạt động của hệ thống
# 1 - Single user mode - Đây là chế độ đơn người dùng, ta có thể đăng nhập vào chế độ
  này trong trường hợp muốn khắc phục một số sự cố.
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking) - Đây là chế
  độ đa người dùng, giao diện text, không hỗ trợ kết nối mạng.
# 3 - Full multiuser mode – Chế độ đa người dùng, giao diện text
# 4 – unused – Không sử dụng chế độ này
# 5 - X11 - Đây là chế độ đa người dùng, giao diện đồ hoạ
# 6 - reboot (Do NOT set initdefault to this) – Chế độ khởi động lại máy tính
#
id:3:initdefault: - Đây là chế độ ngầm định hệ thống sẽ sử dụng để khởi động

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
#ca::ctrlaltdel:/bin/echo "You can't do that"
```

```
# When our UPS tells us power has failed, assume we have a few minutes
# of power left. Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"


# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm –nodaemon
```

Trong đó chế độ khởi động số 3 là chế độ khởi động vào chế độ Text, và chế độ 5 là khở động vào chế độ đồ hoạ. Như vậy để cho máy tính khởi động vào chế độ đồ hoạ ta sửa lại dòng cấu hình

id:3:initdefault:

thành

id:5:initdefault:

Màn hình đăng nhập đồ họa

Trong Linux có một số loại giao diện đồ hoạ do một số tổ chức viết ra. Hai tổ chức nổi tiếng là GNOME (http://www.gnome.org) và KDE (http://www.kde.org) đã viết ra các giao diện đồ hoạ mang tên trùng với tổ chức đó là GNOME và KDE.

Cũng tuỳ vào việc được cài giao diện GNOME hay KDE mà khi khởi động vào chế độ đồ hoạ, máy tính có các giao diện tương ứng. Trên hình trên là giao diện GNOME mà khi

khởi động vào chế độ đồ hoạ. Mặt khác, các giao diện này liên tục được phát triển, do đó ở mỗi phiên bản sẽ có sự khác nhau. Trong giao diện đăng nhập đồ họa, hệ thống hiển thị hộp thoại cho phép người dùng nhập vào tên tài khoản; sau khi người dùng nhập tên tài khoản của mình, hệ thống hỏi tiếp mật khẩu. Nếu cả tên tài khoản và mật khẩu đều chính xác thì người dùng được phép vào hệ thống và một giao diện làm việc mới sẽ hiện ra như hình dưới.

Màn hình GNOME

2.3. Lệnh thay đổi mật khẩu

Mật khẩu là vấn đề rất quan trọng trong các hệ thống đa người dùng và để đảm bảo tính bảo mật tối đa, cần thiết phải chú ý tới việc thay đổi mật khẩu. Thậm chí trong trường hợp hệ thống chỉ có một người sử dụng thì việc thay đổi mật khẩu vẫn là rất cần thiết.

Mật khẩu là một xâu kí tự đi kèm với tên người dùng để đảm bảo cho phép một người vào làm việc trong hệ thống với quyền hạn đã được quy định. Trong quá trình đăng nhập, người dùng phải gõ đúng tên và mật khẩu, trong đó gõ mật khẩu là công việc bắt buộc phải thực hiện. Tên người dùng có thể được công khai song mật khẩu thì tuyệt đối phải được đảm bảo bí mật.

- Việc đăng ký tên và mật khẩu của siêu người dùng được tiến hành trong quá trình khởi tạo hệ điều hành Linux.
- Việc đăng ký tên và mật khẩu của một người dùng thông thường được tiến hành khi một người dùng mới đăng ký tham gia sử dụng hệ thống. Thông thường siêu người dùng cung cấp tên và mật khẩu cho người dùng mới (có thể do người dùng đề nghị)

và dùng lệnh **adduser** (hoặc lệnh **useradd**) để đăng ký tên và mật khẩu đó với hệ thống. Sau đó, người dùng mới nhất thiết cần thay đổi mật khẩu để bảo đảm việc giữ bí mật cá nhân tuyệt đối.

Lệnh **passwd** cho phép thay đổi mật khẩu ứng với tên đăng nhập người dùng. Cú pháp lệnh **passwd**:

```
passwd [tùy-chọn] [tên-người-dùng]
```

với các tùy chọn như sau:

- **-k** : thay đổi mật khẩu người dùng. Lệnh đòi hỏi phải xác nhận quyền bằng việc gõ mật khẩu đang dùng trước khi thay đổi mật khẩu. Cho phép người dùng thay đổi mật khẩu của mình độc lập với siêu người dùng.
- **-f** : đặt mật khẩu mới cho người dùng song không cần tiến hành việc kiểm tra mật khẩu đang dùng. Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-l** : khóa một tài khoản người dùng. Việc khóa tài khoản thực chất là việc dịch bản mã hóa mật khẩu thành một xâu ký tự vô nghĩa bắt đầu bởi kí hiệu "!". Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-stdin** : việc nhập mật khẩu người dùng chỉ được tiến hành từ thiết bị vào chuẩn không thể tiến hành từ đường dẫn (pipe). Nếu không có tham số này cho phép nhập mật khẩu cả từ thiết bị vào chuẩn hoặc từ đường dẫn.
- **-u** : mở khóa (tháo bỏ khóa) một tài khoản (đối ngẫu với tham số -l). Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-d** : xóa bỏ mật khẩu của người dùng. Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-S** : hiển thị thông tin ngắn gọn về trạng thái mật khẩu của người dùng được đưa ra. Chỉ siêu người dùng mới có quyền sử dụng tham số này.

Nếu tên-người-dùng không có trong lệnh thì ngầm định là chính người dùng đã gõ lệnh này.

Ví dụ khi người dùng **user1** gõ lệnh:

```
# passwd user1
```

hệ thống thông báo:

```
Changing password for user user1
New UNIX password:
```

để người dùng nhập mật khẩu mới của mình vào. Sau khi người dùng gõ xong mật khẩu mới, hệ thống cho ra thông báo:

```
BAD PASSWORD: it is derived from your password entry
Retype new UNIX password:
```

để người dùng khẳng định một lần nữa mật khẩu vừa gõ dòng trên (nhớ phải gõ lại đúng hệt như lần trước). Chớ nên quá phân vân vì thông báo ở dòng phía trên vì hầu hết khi gõ mật khẩu mới luôn gặp những thông báo kiểu đại loại như vậy, chẳng hạn như:

```
BAD PASSWORD: it is too simplistic/systematic
```

Và sau khi chúng ta khẳng định lại mật khẩu mới, hệ thống cho ra thông báo:

```
Passwd: all authentication tokens updated successfully.
```

cho biết việc thay đổi mật khẩu thành công và dấu nhắc shell lại hiện ra.
Khi siêu người dùng gõ lệnh:

```
# passwd -S root
```

sẽ hiện ra thông báo

Changing password for user root

Password set, MD5 encryption

cho biết thuật toán mã hóa mật khẩu mà Linux sử dụng là một thuật toán hàm băm có tên là MD5.

☞ ***Lưu ý:***

- Có một lời khuyên đối với người dùng là nên chọn mật khẩu không quá đơn giản quá (nhằm tránh người khác dễ dò tìm ra) hoặc không quá phức tạp (tránh khó khăn cho chính người dùng khi phải ghi nhớ và gõ mật khẩu). Đặc biệt không nên sử dụng họ tên, ngày sinh, số điện thoại ... của bản thân hoặc người thân làm mật khẩu vì đây là một trong những trường hợp mật khẩu đơn giản nhất.
- Nếu thông báo mật khẩu quá đơn giản được lặp đi lặp lại một vài lần và không có thông báo mật khẩu mới thành công đã quay về dấu nhắc shell thì nên gõ lại lệnh và chọn một mật khẩu mới phức tạp hơn đôi chút.

2.4. Lệnh xem, thiết đặt ngày, giờ hiện tại và xem lịch trên hệ thống

2.4.1 Lệnh xem, thiết đặt ngày, giờ

Lệnh **date** cho phép có thể xem hoặc thiết đặt lại ngày giờ trên hệ thống.

Cú pháp của lệnh gồm hai dạng, dạng xem thông tin về ngày, giờ:

```
date [tùy-chọn] [+định-dạng]
```

và dạng thiết đặt lại ngày giờ cho hệ thống:

```
date [tùy-chọn] [MMDDhhmm[ [CC[YY] ]-ss]]
```

Các tùy-chọn như sau:

- **-d, --date=xâu-văn-bản** : hiển thị thời gian dưới dạng ***xâu-văn-bản***, mà không lấy "thời gian hiện tại của hệ thống" như theo ngầm định; ***xâu-văn-bản*** được đặt trong hai dấu nháy đơn hoặc hai dấu nháy kép.
- **-f, --file=file-văn-bản** : giống như một tham số **--date** nhưng ứng với nhiều ngày cần xem: mỗi dòng của file-văn-bản có vai trò như một **xâu-văn-bản** trong trường hợp tham số **--date**.
- **-I, --iso-8601[=mô-tả]** : hiển thị ngày giờ theo chuẩn ISO-8601 (ví dụ: 2000-11-8).
  - -I tương đương với tham số **--iso-8601='date'**
  - Với **--iso-8601**: nếu ***mô-tả*** là '**date**' (hoặc không có) thì hiển thị ngày, nếu ***mô-tả*** là '**hours**' hiển thị ngày+giờ, nếu ***mô-tả*** là **'minutes'**: ngày+giờ+phút; nếu ***mô-tả*** là **'seconds'**: ngày + giờ + phút + giây.
- **-r, --reference= file** : hiển thị thời gian sửa đổi ***file*** lần gần đây nhất.
- **-R, --rfc-822** : hiển thị ngày theo RFC-822 (ví dụ: Wed, 8 Nov 2000 09:21:46 -0500).
- **-s, --set=xâu-văn-bản** : thiết đặt lại thời gian theo kiểu **xâu-văn-bản**.
- **-u, --utc, --universal** : hiển thị hoặc thiết đặt thời gian theo UTC (ví dụ: Wed Nov 8 14:29:12 UTC 2000).
- **--help** : hiển thị thông tin trợ giúp và thoát.

Trong dạng lệnh **date** cho xem thông tin ngày, giờ thì tham số định-dạng điều khiển cách hiển thị thông tin kết quả. Định-dạng là dãy có từ một đến nhiều cặp gồm hai kí tự, trong mỗi cặp kí tự đầu tiên là % còn kí tự thứ hai mô tả định dạng.

Do số lượng định dạng là rất nhiều vì vậy chúng ta chỉ xem xét một số định dạng điển hình (để xem đầy đủ các định dạng, sử dụng lệnh **man date**).

Dưới đây là một số định dạng điển hình:

- %% : Hiện ra chính kí tự %.
- %**a** : Hiện ra thông tin tên ngày trong tuần viết tắt theo ngôn ngữ bản địa.
- %**A** : Hiện ra thông tin tên ngày trong tuần viết đầy đủ theo ngôn ngữ bản địa.
- %**b** : Hiện ra thông tin tên tháng viết tắt theo ngôn ngữ bản địa.
- %**B** : Hiện ra thông tin tên tháng viết đầy đủ theo ngôn ngữ bản địa.

Trong dạng lệnh **date** cho phép thiết đặt lại ngày giờ cho hệ thống thì tham số [MMDDhhmm[[CC[YY] [.ss]] mô tả ngày, giờ mới cần thiết đặt, trong đó:

MM: hai số chỉ tháng,
DD: hai số chỉ ngày trong tháng,
hh: hai số chỉ giờ trong ngày,
mm: hai số chỉ phút,
CC: hai số chỉ thế kỉ,
YY: hai số chỉ năm trong thế kỉ.

Các dòng ngay dưới đây trình bày một số ví dụ sử dụng lệnh **date**, mỗi ví dụ được cho tương ứng với một cặp hai dòng, trong đó dòng trên mô tả lệnh được gõ còn dòng dưới là thông báo của Linux.

```
# date
Wed Jan 3 23:58:50 ICT 2001
# date -d='01/01/2000'
Sat Jan 1 00:00:00 ICT 2000
# date -iso-8601='seconds'
2000-12-01T00:36:41-0500
# date -d='01/01/2001'
Mon Jan 1 00:00:00 ICT 2001
# date 010323502001.50
Wed Jan 3 23:50:50 ICT 2001
# date +%a%A
Wed Wednesday
# date +%a%A%b%B
Wed Wednesday Jan January
# date +%D%%%j
01/05/01%005
```

2.4.2. Lệnh xem lịch

Lệnh **cal** cho phép xem lịch trên hệ thống với cú pháp như sau:

```
cal [tùy-chọn] [<tháng> [<năm>]]
```

nếu không có tham số, lịch của tháng hiện thời sẽ được hiển thị.

Các tuỳ-chọn là:

- **-m** : chọn ngày Thứ hai là ngày đầu tiên trong tuần (mặc định là ngày Chủ nhật).
- **-j** : hiển thị số ngày trong tháng dưới dạng số ngày trong năm (ví dụ: ngày 1/11/2000 sẽ được hiển thị dưới dạng là ngày thứ 306 trong năm 2000, số ngày bắt đầu được tính từ ngày 1/1).
- **-y** : hiển thị lịch của năm hiện thời.

Ví dụ:

```
# cal 1 2001
```

January 2001

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Khi nhập dòng lệnh trên, trên màn hình sẽ hiển thị lịch của tháng 1 năm 2001, mặc định chọn ngày chủ nhật là ngày bắt đầu của tuần. Dưới đây là ví dụ hiển thị số ngày trong tháng 3 dưới dạng số ngày trong năm 2001.

```
# cal -j 3 2001
```

March 2001

Su	Mo	Tu	We	Th	Fr	Sa
				60	61	62
63	64	65	66	67	68	69
70	71	72	73	74	75	76
77	78	79	80	81	82	83
84	85	86	87	88	89	90

2.5. Xem thông tin hệ thống

Lệnh **uname** cho phép xem thông tin hệ thống với cú pháp là:

```
uname [tùy-chọn]
```

Nếu không có tuỳ chọn thì hiện tên hệ điều hành.

Lệnh có các tùy chọn là:

- **-a, --all** : hiện tất cả các thông tin.
- **-m, --machine** : kiểu kiến trúc của bộ xử lý (i386, i486, i586, i686...).
- **-n, --nodename** : hiện tên của máy.
- **-r, --release** : hiện nhân của hệ điều hành.
- **-s, --sysname** : hiện tên hệ điều hành.
- **-p, --processor** : hiện kiểu bộ xử lý của máy chủ.

Ví dụ, nếu gõ lệnh

```
# uname -a
```

thì màn hình sẽ hiện ra như sau:

Linux linuxsrv.linuxvn.net 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown

#

Thông tin hiện ra có tất cả 6 trường là:

Tên hệ điều hành: Linux
Tên máy: linuxsrv.linuxvn.net
Tên nhân của hệ điều hành: 2.2.14-5.0
Ngày sản xuất: #1 Tue Mar 7 21:07:39 EST 2000
Kiểu kiến trúc bộ xử lý: i686
Kiểu bộ xử lý của máy chủ: unknown

Ví dụ nếu gõ lệnh:

```
# uname -spr
```

thì màn hình sẽ hiện ra như sau:

Linux 2.2.14-5.0 unknown

là tên hệ điều hành, tên nhân và kiểu bộ xử lý của máy chủ.

☞ ***Lưu ý:***

- Chúng ta làm rõ thêm nội dung lưu ý trong ***mục 1.3.1*** về tham số khóa kết hợp: Trong ví dụ trên đây khi viết tham số **-spr** là yêu cầu thực hiện lệnh **usame** với nghĩa kết hợp tình huống theo cả ba tham số khóa **-s**, **-p**, **-r**. Chú ý rằng, không thể viết **-s -p -r** thay cho **-spr** được. Như đã lưu ý ở ***mục 1.3.1*** trong nhiều lệnh của Linux cho phép viết kết hợp các tham số khóa theo cách thức như trên miễn là các tham số đó không xung khắc với nhau.

2.6. Thay đổi nội dung dấu nhắc shell

Trong Linux có hai loại dấu nhắc: dấu nhắc cấp một (dấu nhắc shell) xuất hiện khi nhập lệnh và dấu nhắc cấp hai (dấu nhắc nhập liệu) xuất hiện khi lệnh cần có dữ liệu được nhập từ bàn phím và tương ứng với hai biến nhắc tên là **PS1** và **PS2**.

PS1 là biến hệ thống tương ứng với dấu nhắc cấp 1: Giá trị của **PS1** chính là nội dung hiển thị của dấu nhắc shell. Để nhận biết thông tin hệ thống hiện tại, một nhu cầu đặt ra là cần thay đổi giá trị của các biến hệ thống **PS1** và **PS2**.

Linux cho phép thay đổi giá trị của biến hệ thống **PS1** bằng lệnh gán trị mới cho nó. Lệnh này có dạng:

```
# PS1='<dãy kí tự>'
```

Năm (5) kí tự đầu tiên của lệnh gán trên đây (**PS1**=') phải được viết liên tiếp nhau. Dãy kí tự nằm giữa cặp hai dấu nháy đơn (có thể sử dụng cặp hai dấu kép ") và không được phép chứa dấu nháy. Dãy kí tự này bao gồm các cặp kí tự điều khiển và các kí tự khác, cho phép có thể có dấu cách. Cặp kí tự điều khiển gồm hai kí tự, kí tự đầu tiên là dấu số xuôi "\" còn kí tự thứ hai nhận một trong các trường hợp liệt kê trong bảng dưới đây. Bảng dưới đây giới thiệu một số cặp ký tự điều khiển có thể được sử dụng khi muốn thay đổi dấu nhắc lệnh:

Cặp ký tự điều khiển	*Ý nghĩa*

\!	Hiển thị thứ tự của lệnh trong lịch sử
\#	Hiển thị thứ tự của lệnh
\$	Hiển thị dấu đô-la ($). Đối với siêu người dùng (super user), thì hiển thị dấu số hiệu (#)
\\	Hiển thị dấu sổ (\)
\d	Hiển thị ngày hiện tại
\h	Hiển thị tên máy (hostname)
\n	Ký hiệu xuống dòng
\s	Hiển thị tên hệ shell
\t	Hiển thị giờ hiện tại
\u	Hiển thị tên người dùng
\W	Hiển thị tên thực sự của thư mục hiện thời (ví dụ thư mục hiện thời là /mnt/hda1 thì tên thực sự của nó là /hda1)
\w	Hiển thị tên đầy đủ của thư mục hiện thời (ví dụ **/mnt/hda1**)

Ví dụ, hiện thời dấu nhắc shell có dạng:

```
root[may1 /hda1]#
```

Sau khi gõ lệnh

```
root@may1 /hda1]# PS1='[\h@\u \w : \d]\$'
```

thì dấu nhắc shell được thay đổi là:

```
[may1@root /mnt/hda1 : Fri Oct 27 ]#
```

ngoài việc đổi thứ tự giữa tên người dùng và máy còn cho chúng ta biết thêm về ngày hệ thống quản lý và tên đầy đủ của thư mục hiện thời.
Linux cung cấp cách thức hoàn toàn tương tự như đối với biến **PS1** để thay đổi giá trị biến hệ thống **PS2** tương ứng với dấu nhắc cấp hai.

2.7. Lệnh gọi ngôn ngữ tính toán số học

Linux cung cấp một ngôn ngữ tính toán với độ chính xác tùy ý thông qua lệnh **bc**. Khi yêu cầu lệnh này, người dùng được cung cấp một ngôn ngữ tính toán (và cho phép lập trình tính toán có dạng ngôn ngữ lập trình C) hoạt động theo thông dịch. Trong ngôn ngữ lập trình được cung cấp (tạm thời gọi là ngôn ngữ ***bc***), tồn tại rất nhiều công cụ hỗ trợ tính toán và lập trình tính toán: kiểu phép toán số học phong phú, phép toán so sánh, một số hàm chuẩn, biến chuẩn, cấu trúc điều khiển, cách thức định nghĩa hàm, cách thức thay đổi độ chính xác, đặt lời chú thích ... Chỉ cần sử dụng một phần nhỏ tác động của lệnh **bc**, chúng ta đã có một "máy tính số bấm tay" hiệu quả.

Cú pháp lệnh **bc**:

```
bc [tùy-chọn] [file...]
```

với các tuỳ chọn sau đây:

- **-l, --mathlib** : thực hiện phép tính theo chuẩn thư viện toán học (ví dụ: 5/5=1.00000000000000000000).
- **-w, --warn** : khi thực hiện phép tính không tuân theo chuẩn POSIX (POSIX là một chuẩn trong Linux) thì một cảnh báo xuất hiện.

- **-s, --standard** : thực hiện phép tính chính xác theo chuẩn của ngôn ngữ POSIX **bc.**
- **-q, --quiet** : không hiện ra lời giới thiệu về phần mềm GNU khi dùng **bc.**

Tham số file là tên file chứa chương trình viết trên ngôn ngữ ***bc***, khi lệnh **bc** thực hiện sẽ tự động chạy các file chương trình này (Nếu có nhiều tham số thì có nghĩa sẽ chạy nhiều chương trình liên tiếp nhau).

Dưới đây là một ví dụ sử dụng lệnh **bc** ở dạng đơn giản nhất.

Khi gõ lệnh tại dấu nhắc:

```
# bc -l
```

màn hình xuất hiện lời giới thiệu về GNU khi dùng **bc** và ngôn ngữ ***bc*** được kích hoạt để phục vụ người dùng.

```
bc 1.05
Copyright 1991, 1992, 1993, 1994, 1997, 1998 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
5^3
125
12+12+78*7-62/4
554.50000000000000000000
a=4
a^a
256
a*78
312
b=45
a*b
180
a/b
.08888888888888888888
a%b
.00000000000000000040
```

ở đây * là phép nhân, ^ là phép tính luỹ thừa, / là phép chia lấy thương, % là chia lấy phần dư.

☞ ***Lưu ý:***

- Ngôn ngữ lập trình tính toán ***bc*** là một ngôn ngữ rất mạnh có nội dung hết sức phong phú cho nên trong khuôn khổ của tài liệu này không thể mô tả hết các nội dung của ngôn ngữ đó được. Chúng ta cần sử dụng lệnh **man bc** để nhận được thông tin đầy đủ về lệnh **bc** và ngôn ngữ tính toán ***bc***.
- Ở đây trình bày sơ bộ một số yếu tố cơ bản nhất của ngôn ngữ đó (***bt*** là viết tắt của biểu thức, ***b*** là viết tắt của biến):

  Các phép tính: - bt: lấy đối; ++ b, --b, b ++, b --: phép toán tăng, giảm b; các phép toán hai ngôi cộng +, trừ -, nhân *, chia /, lấy phần dư %, lũy thừa nguyên

bậc ^; gán =; gán sau khi thao tác <thao tác>=; các phép toán so sánh <, <=, >, >=, bằng ==, khác != ...

Phép so sánh cho 1 nếu đúng, cho 0 nếu sai.

Bốn biến chuẩn là ***scale*** số lượng chữ số phần thập phân; ***last*** giá trị tính toán cuối cùng; ***ibase*** cơ số hệ đếm đối với input và ***obase*** là cơ số hệ đếm với output (ngầm định hai biến này có giá trị 10).

Các hàm chuẩn sin s (bt); cosin c (bt); arctg a (bt); lôgarit tự nhiên l (bt); mũ cơ số tự nhiên e (bt); hàm Bessel bậc nguyên n của bt là j (n, bt).

CHƯƠNG 3. HỆ THỐNG FILE

3.1 Tổng quan về hệ thống file

3.1.1. Một số khái niệm

Người dùng đã từng làm việc với hệ điều hành DOS/Windows thì rất quen biết với các khái niệm: file (tập tin), thư mục, thư mục hiện thời ... Để đảm bảo tính hệ thống và thuận tiện cho người dùng chưa từng làm việc thành thạo với một hệ điều hành nào khác, chương này vẫn giới thiệu về các khái niệm này một cách sơ bộ.

Một đối tượng điển hình trong các hệ điều hành đó là ***file***. File là một tập hợp dữ liệu có tổ chức được hệ điều hành quản lý theo yêu cầu của người dùng. Cách tổ chức dữ liệu trong file thuộc về chủ của nó là người đã tạo ra file. File có thể là một văn bản (trường hợp đặc biệt là chương trình nguồn trên C, PASCAL, shell script ...), một chương trình ngôn ngữ máy, một tập hợp dữ liệu ... Hệ điều hành tổ chức việc lưu trữ nội dung file trên các thiết bị nhớ lâu dài (chẳng hạn đĩa từ) và đảm bảo các thao tác lên file. Chính vì có hệ điều hành đảm bảo các chức năng liên quan đến file nên người dùng không cần biết file của mình lưu ở vùng nào trên đĩa từ, bằng cách nào đọc/ghi lên các vùng của đĩa từ mà vẫn thực hiện được yêu cầu tìm kiếm, xử lý lên các file.

Hệ điều hành quản lý file theo tên gọi của file (tên file) và một số thuộc tính liên quan đến file. Trước khi giới thiệu một số nội dung liên quan đến tên file và tên thư mục, chúng ta giới thiệu sơ bộ về khái niệm thư mục.

Để làm việc được với các file, hệ điều hành không chỉ quản lý nội dung file mà còn phải quản lý các thông tin liên quan đến các file. Thư mục (directory) là đối tượng được dùng để chứa thông tin về các file, hay nói theo một cách khác, thư mục chứa các file. Các thư mục cũng được hệ điều hành quản lý trên vật dẫn ngoài và vì vậy, theo nghĩa này, thư mục cũng được coi là file song trong một số trường hợp để phân biệt với "file" thư mục, chúng ta dùng thuật ngữ ***file thông thường***. Khác với file thông thường, hệ điều hành lại quan tâm đến nội dung của thư mục.

Một số nội dung sau đây liên quan đến tên file (bao gồm cả tên thư mục):

- Tên file trong Linux có thể dài tới 256 ký tự, bao gồm các chữ cái, chữ số, dấu gạch nối, gạch chân, dấu chấm. Tên thư mục/file trong Linux có thể có nhiều hơn một dấu chấm, ví dụ: **This_is.a.VERY_long.filename**. Nếu trong tên file có dấu chấm "." thì xâu con của tên file từ dấu chấm cuối cùng được gọi là phần mở rộng của tên file (hoặc file). Ví dụ, tên file trên đây có phần mở rộng là **.filename**. Chú ý rằng khái niệm phần mở rộng ở đây không mang ý nghĩa như một số hệ điều hành khác (chẳng hạn như MS-DOS).

☞ ***Lưu ý:***

- Chúng ta nên lưu ý rằng, không phải ký tự nào cũng có nghĩa. Nếu có hai file chỉ khác nhau ở ký tự cuối cùng, thì đối với Linux, đó là hai file có thể trùng tên. Bởi lẽ, Linux chỉ lấy 32 hay 64 ký tự đầu tiên trong tên file mà thôi (tùy theo phiên bản Linux), phần tên file còn lại dành cho chủ của file, Linux theo dõi thông tin, nhưng thường không xem các ký tự đứng sau ký tự thứ 33 hay 65 là quan trọng đối với nó.

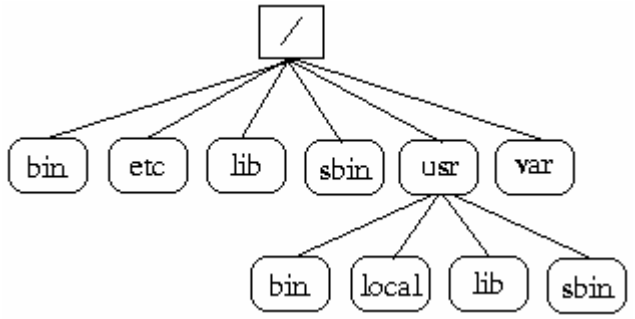
- Xin nhắc lại lưu ý về phân biệt chữ hoa và chữ thường đối với tên thư mục/file, ví dụ hai file **FILENAME.tar.gz** và **filename.tar.gz** là hai file khác nhau.

- Nếu trong tên thư mục/file có chứa khoảng trống, sẽ phải đặt tên thư mục/file vào trong cặp dấu nháy kép để sử dụng thư mục/file đó. Ví dụ, để tạo thư mục có tên là "My document" chẳng hạn, hãy đánh dòng lệnh sau:

```
# mkdir "My document"
```

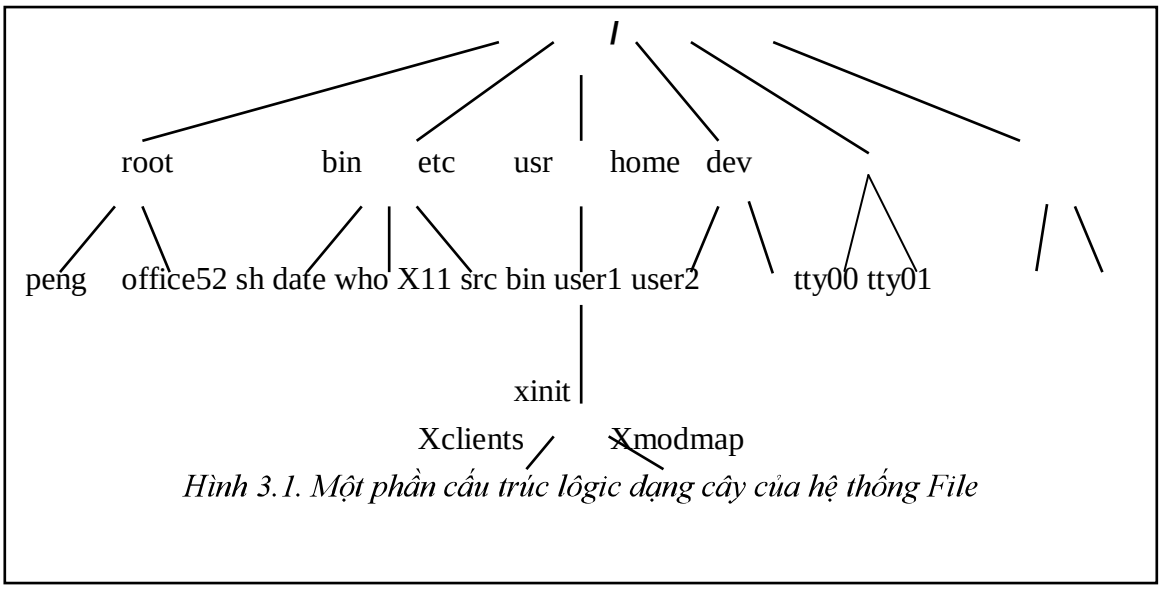
- Một số ký tự sau không được sử dụng trong tên thư mục/file: !, *, $, &, # ...
- Khi sử dụng chương trình **mc** (Midnight Commander), việc hiển thị tên file sẽ bổ sung một kí tự theo nghĩa: dấu "*" cho file khả thi trong Linux, dấu "~" cho file sao lưu, dấu "." cho file ẩn, dấu "@" cho file liên kết...

Tập hợp tất cả các file có trong hệ điều hành được gọi là ***hệ thống file*** là một hệ thống thống nhất. Bởi chính từ cách thức sử dụng thư mục, hệ thống file được tổ chức lôgic theo dạng hình cây: Hệ thống file được xuất phát từ một thư mục gốc (được kí hiệu là "**/**") và cho phép tạo ra thư mục con trong một thư mục bất kỳ. Thông thường, khi khởi tạo Linux đã có ngay hệ thống file của nó. Hình 3.1. cho minh họa một phần trong cây lôgic của hệ thống file.



Để chỉ một file hay một thư mục, chúng ta cần đưa ra một đường dẫn, ví dụ để đường dẫn xác định file **`Xclients`** trong hình 3.1. chúng ta viết như sau:

/etc/X11/xinit/Xclients



Hình 3.1. Một phần cấu trúc lôgic dạng cây của hệ thống File

Đường dẫn này cho biết **Xclients** nằm trong **xinit**, **xinit** nằm trong **X11**, **X11** nằm trong **etc** và **etc** nằm trong gốc **/**.

Tên file thường là tham số thực sự khi gõ lệnh và công việc gõ lệnh trở nên rất nặng nề đối với người dùng nếu như trong lệnh phải gõ một đường dẫn dài theo dạng trên (được biết với tên gọi là ***đường dẫn tuyệt đối***). Vì vậy, Linux (cũng như nhiều hệ điều hành khác) sử dụng khái niệm ***thư mục hiện thời*** của mỗi người dùng làm việc trong hệ thống. Thư mục hiện thời là một thư mục trong hệ thống file mà hiện thời "người dùng đang ở đó".

Qua thư mục hiện thời, Linux cho phép người dùng chỉ một file trong lệnh ngắn gọn hơn nhiều. Ví dụ, nếu thư mục hiện thời là thư mục ***xinit*** thì để chỉ file đã nói, người dùng chỉ cần viết **Xclients** hoặc **./Xclients** trong đó kí hiệu "**.**" để chỉ thư mục hiện thời. Đường dẫn được xác định qua thư mục hiện thời được gọi là ***đường dẫn tương đối***.

Khi một người dùng đăng nhập vào hệ thống, Linux luôn chuyển người dùng vào thư mục riêng, và tại thời điểm đó thư mục riêng là thư mục hiện thời của người dùng. Thư mục riêng của siêu người dùng là **/root**, thư mục riêng của người dùng có tên là ***user1*** là **/home/user1** ... Linux cho phép dùng lệnh **cd** để chuyển sang thư mục khác (lấy thư mục khác làm thư mục hiện thời). Hai dấu chấm ".." được dùng để chỉ thư mục ngay trên thư mục hiện thời (cha của thư mục hiện thời).

Linux còn cho phép ghép một hệ thống file trên một thiết bị nhớ (đĩa mềm, vùng đĩa cứng chưa được đưa vào hệ thống file) thành một thư mục con trong hệ thống file của hệ thống bằng lệnh ***mount***. Các hệ thống file được ghép thuộc vào các kiểu khác nhau.

Hai mục tiếp theo (3.1.2 và 3.1.3.) giới thiệu những nội dung sâu hơn về hệ thống file Linux.

3.1.2. Sơ bộ kiến trúc nội tại của hệ thống file

Trên đĩa từ, hệ thống file được coi là dãy tuần tự các khối lôgic mỗi khối chứa hoặc 512B hoặc 1024B hoặc bội của 512B là cố định trong một hệ thống file. Trong hệ thống file, các khối dữ liệu được địa chỉ hóa bằng cách đánh chỉ số liên tiếp, mỗi địa chỉ được chứa trong 4 byte (32 bit).

Cấu trúc nội tại của hệ thống file bao gồm 4 thành phần kế tiếp nhau: Boot block (dùng để khởi động hệ thống), Siêu khối (Super block), Danh sách inode và Vùng dữ liệu.

Dưới đây, chúng ta xem xét sơ lược nội dung các thành phần cấu trúc nội tại một hệ thống file.

Siêu khối

Siêu khối chứa nhiều thông tin liên quan đến trạng thái của hệ thống file. Trong siêu khối có các trường sau đây:

- Kích thước của danh sách inode (khái niệm inode sẽ được giải thích trong mục sau): định kích cỡ vùng không gian trên Hệ thống file quản lý các inode.
- Kích thước của hệ thống file.

Hai kích thước trên đây tính theo đơn vị dung lượng bộ nhớ ngoài,

- Một danh sách chỉ số các khối rỗi (thường trực trên siêu khối) trong hệ thống file.

Chỉ số các khối rỗi thường trực trên siêu khối được dùng để đáp ứng nhu cầu phân phối mới. Chú ý rằng, danh sách chỉ số các khối rỗi có trên siêu khối chỉ là một bộ phận của tập tất cả các khối rỗi có trên hệ thống file.

- Chỉ số của khối rỗi tiếp theo trong danh sách các khối rỗi.

Chỉ số khối rỗi tiếp theo dùng để hỗ trợ việc tìm kiếm tiếp các khối rỗi: bắt đầu tìm từ khối có chỉ số này trở đi. Điều đó có nghĩa là mọi khối có chỉ số không lớn hơn chỉ số này hoặc có trong danh sách các khối rỗi thường trực hoặc đã được cấp phát cho một file nào đó. Nhiều thao tác tạo file mới, xoá file, thay đổi nội dung file v.v. cập nhật các thông tin này.

- Một danh sách các inode rỗi (thường trực trên siêu khối) trong hệ thống file.

Danh sách này chứa chỉ số các inode rỗi được dùng để phân phối ngay được cho một file mới được khởi tạo. Thông thường, danh sách này chỉ chứa một bộ phận các inode rỗi trên hệ thống file.

- Chỉ số inode rỗi tiếp theo trong danh sách các inode rỗi.

Chỉ số inode rỗi tiếp theo định vị việc tìm kiếm tiếp thêm inode rỗi: bắt đầu tìm từ inode có chỉ số này trở đi. Điều đó có nghĩa là mọi inode có chỉ số không lớn hơn chỉ số này hoặc có trong danh sách các inode rỗi thường trực hoặc đã được tương ứng với một file nào đó.

Hai tham số trên đây tạo thành cặp xác định được danh sách các inode rỗi trên hệ thống file các thao tác tạo file mới, xoá file cập nhật thông tin này.

- Các trường khóa (lock) danh sách các khối rỗi và danh sách inode rỗi: Trong một số trường hợp, chẳng hạn khi hệ thống đang làm việc thực sự với đĩa từ để cập nhật các danh sách này, hệ thống không cho phép cập nhật tới hai danh sách nói trên.
- Cờ chỉ dẫn về việc siêu khối đã được biến đổi: Định kỳ thời gian siêu khối ở bộ nhớ trong được cập nhật lại vào siêu khối ở đĩa từ và vì vậy cần có thông tin về việc siêu khối ở bộ nhớ trong khác với nội dung ở bộ nhớ ngoài: nếu hai bản không giống nhau thì cần phải biến đổi để chúng được đồng nhất.
- Cờ chỉ dẫn rằng hệ thống file chỉ có thể đọc (cấm ghi): Trong một số trường hợp, hệ thống đang cập nhật thông tin từ bộ nhớ ngoài thì chỉ cho phép đọc đối với hệ thống file,
- Số lượng tổng cộng các khối rỗi trong hệ thống file,
- Số lượng tổng cộng các inode rỗi trong hệ thống file,
- Thông tin về thiết bị,
- Kích thước khối (đơn vị phân phối dữ liệu) của hệ thống file. Hiện tại kích thước phổ biến của khối là 1KB.

Trong thời gian máy hoạt động, theo từng giai đoạn, nhân sẽ đưa siêu khối lên đĩa nếu nó đã được biến đổi để phù hợp với dữ liệu trên hệ thống file.

Một trong khái niệm cốt lõi xuất hiện trong hệ thống file đó là inode. Các đối tượng liên quan đến khái niệm này sẽ được trình bày trong các mục tiếp theo.

Inode

Mỗi khi một quá trình khởi tạo một file mới, nhân hệ thống sẽ gán cho nó một inode chưa sử dụng. Để hiểu rõ hơn về inode, chúng ta xem xét sơ lược mối quan hệ liên quan giữa file dữ liệu và việc lưu trữ trên vật dẫn ngoài đối với Linux.

Nội dung của file được chứa trong vùng dữ liệu của hệ thống file và được phân chia các khối dữ liệu (chứa nội dung file) và hình ảnh phân bố nội dung file có trong một inode tương ứng. Liên kết đến tập hợp các khối dữ liệu này là một inode, chỉ thông qua inode mới có thể làm việc với dữ liệu tại các khối dữ liệu: Inode chứa dựng thông tin về tập hợp các khối dữ liệu nội dung file. Có thể quan niệm rằng, tổ hợp gồm inode và tập các khối dữ liệu như vậy là một file vật lý: inode có thông tin về file vật lý, trong đó có địa chỉ của các khối nhớ chứa nội dung của file vật lý. Thuật ngữ inode là sự kết hợp của hai từ index với node và được sử dụng phổ dụng trong Linux.

Các inode được phân biệt nhau theo chỉ số của inode: đó chính là số thứ tự của inode trong danh sách inode trên hệ thống file. Thông thường, hệ thống dùng 2 bytes để lưu trữ chỉ số của inode. Với cách lưu trữ chỉ số như thế, không có nhiều hơn 65535 inode trong một hệ thống file.

Như vậy, một file chỉ có một inode song một file lại có một hoặc một số tên file. Người dùng tác động thông qua tên file và tên file lại tham chiếu đến inode (tên file và chỉ số inode là hai trường của một phần tử của một thư mục). Một inode có thể tương ứng với một hoặc nhiều tên file, mỗi tương ứng như vậy được gọi là một liên kết. Inode được lưu trữ tại vùng danh sách các inode.

Trong quá trình làm việc, Linux dùng một vùng bộ nhớ, được gọi là bảng inode (trong một số trường hợp, nó còn được gọi tường minh là bảng sao in-core inode) với chức năng tương ứng với vùng danh sách các inode có trong hệ thống file, hỗ trợ cho quá trình truy nhập dữ liệu trong hệ thống file. Nội dung của một in-core inode không chỉ chứa các thông tin trong inode tương ứng mà còn được bổ sung các thông tin mới giúp cho quá trình xử lý inode.

Chúng ta xem xét cấu trúc nội tại của một inode để thấy được sự trình bày nội tại của một file. Inode bao gồm các trường thông tin sau đây:

- Kiểu file. Trong Linux phân loại các kiểu file: file thông thường (regular), thư mục, đặc tả kí tự, đặc tả khối và ống dẫn FIFO (pipes). Linux quy định trường kiểu file có giá trị 0 tương ứng đó là inode chưa được sử dụng.
- Quyền truy nhập file. Trong Linux, file là một tài nguyên chung của hệ thống vì vậy quyền truy nhập file được đặc biệt quan tâm để tránh những trường hợp truy nhập không hợp lệ. Đối với một inode, có 3 mức quyền truy nhập liên quan đến các đối tượng:
  - ❖ mức chủ của file (đối tượng này được ký hiệu là ***u***: từ chữ user),
  - ❖ mức nhóm người dùng của chủ nhân của file (đối tượng này được ký hiệu là ***g***: từ chữ group),
  - ❖ mức người dùng khác (đối tượng này được ký hiệu là ***a***: từ chữ all).

Quyền truy nhập là đọc, ghi, thực hiện hoặc một tổ hợp nào đó từ nhóm gồm 3 quyền trên. Chú ý rằng, quyền thực hiện đối với một thư mục tương ứng với việc cho phép tìm một tên file có trong thư mục đó.

- Số lượng liên kết đối với inode: Đây chính là số lượng các tên file trên các thư mục được liên kết với inode này,

- Định danh chủ nhân của inode,
- Định danh nhóm chủ nhân: xác định tên nhóm người dùng mà chủ file là một thành viên của nhóm này,
- Độ dài của file tính theo byte,
- Thời gian truy nhập file:
  - thời gian file được sửa đổi muộn nhất,
  - thời gian file được truy nhập muộn nhất,
  - thời gian file được khởi tạo,
- Bảng địa chỉ chứa các địa chỉ khối nhớ chứa nội dung file. Bảng này có 13 phần tử địa chỉ, trong đó có 10 phần tử trực tiếp, 1 phần tử gián tiếp bậc 1, 1 phần tử gián tiếp bậc 2 và một phần tử gián tiếp bậc 3 (chi tiết có trong phần sau).

Nội dung của file thay đổi khi có thao tác ghi lên nó; nội dung của một inode thay đổi khi nội dung của file thay đổi hoặc thay đổi chủ hoặc thay đổi quyền hoặc thay đổi số liên kết.

Ví dụ về nội dung một inode như sau:

type regular
perms rwxr-xr-x
links 2
owner 41CT
group 41CNTT
size 5703 bytes
accessed Sep 14 1999 7:30 AM
modified Sep 10 1999 1:30 PM
inode Aug 1 1995 10:15 AM
Các phần tử địa chỉ dữ liệu

Bản sao in-core inode còn bổ sung thêm trường trạng thái của in-core inode.

- Trường trạng thái của in-core inode có các thông tin sau:
  - inode đã bị khoá,
  - một quá trình đang chờ đợi khi inode tháo khóa,
  - in-core inode khác với inode do sự thay đổi dữ liệu trong inode,
  - in-core inode khác với inode do sự thay đổi dữ liệu trong file,
  - số lượng các tên file nối với file đang được mở,
  - số hiệu thiết bị lôgic của hệ thống file chứa file nói trên
  - chỉ số inode: dùng để liên kết với inode trên đĩa,
  - các móc nối tới các in-core inode khác. Trong bộ nhớ trong, các in-core inode được liên kết theo một hàng băm và một danh sách tự do. Trong danh sách hàng băm các in-core inode hòa hợp theo số hiệu thiết bị lôgic và số hiệu inode.

Trong quá trình hệ thống làm việc, nảy sinh khái niệm inode tích cực nếu như có một quá trình đang làm việc với inode đó (như mở file).

Một inode thuộc vào danh sách các inode rỗi khi không có file vật lý nào tương ứng với inode đó.

Bảng chứa địa chỉ khối dữ liệu của File trong UNIX

Bảng chứa địa chỉ khối dữ liệu của file gồm 13 phần tử với 10 phần tử trực tiếp và 3 phần tử gián tiếp: Mỗi phần tử có độ dài 4 bytes, chứa một số hiệu của một khối nhớ trên đĩa. Mỗi phần tử trực tiếp trỏ tới 1 khối dữ liệu thực sự chứa nội dung file. Phần tử gián tiếp bậc 1 (single indirect) trỏ tới 1 khối nhớ ngoài. Khác với phần tử trực tiếp, khối nhớ ngoài này không dùng để chứa dữ liệu của file mà lại chứa danh sách chỉ số các khối nhớ ngoài và chính các khối nhớ ngoài này mới thực sự chứa nội dung file. Như vậy, nếu khối có độ dài 1KB và một chỉ số khối ngoài có độ dài 4 bytes thì địa chỉ gián tiếp cho phép định vị không gian trên đĩa lưu trữ dữ liệu của file tới 256KB (Không gian bộ nhớ ngoài trong vùng dữ liệu phải dùng tới là 257KB). Tương tự đối với các phần tử gián tiếp mức cao hơn.

Cơ chế quản lý địa chỉ file như trên cho thấy có sự phân biệt giữa file nhỏ với file lớn. File nhỏ có độ dài bé hơn và theo cách tổ chức như trên, phương pháp truy nhập sẽ cho phép tốc độ nhanh hơn, đơn giản hơn do chỉ phải làm việc với các phần tử trực tiếp. Khi xử lý, thuật toán đọc File tiến hành theo các cách khác nhau đối với các phần tử trực tiếp và gián tiếp.

Cơ chế tổ chức lưu trữ nội dung Fle như đã trình bày cho phép độ dài file có thể lên tới $(2^{24}+2^{16} + 2^{8}+10)$ khối.

Vùng dữ liệu bao gồm các khối dữ liệu, mỗi khối dữ liệu được đánh chỉ số để phân biệt. Khối trên vùng dữ liệu được dùng để chứa nội dung các file, nội dung các thư mục và nội dung các khối định vị địa chỉ của các file. Chú ý rằng, chỉ số của khối dữ liệu được chứa trong 32 bit và thông tin này xác định dung lượng lớn nhất của hệ thống file.

3.1.3. Một số thuật toán làm việc với inode

Hệ thống lời gọi hệ thống file

Khi làm việc với file thường thông qua lời gọi hệ thống. Một số lời gọi hệ thống thường gặp như mở file open, đóng file close, đọc nội dung file read, ghi nội dung file write v.v.

Bảng dưới đây thống kê các lời gọi hệ thống làm việc với hệ thống file và phân loại theo chức năng của mỗi lời gọi hệ thống (một lời gọi có thể được nhắc tới một số lần):

Thời điểm sử dụng file	Sử dụng ***namei***		gán inode	thuộc tính file	Vào-ra file	Cấu trúc hệ thống file	Quản lý cây
open ***creat*** ***dup*** ***pipe*** ***close***	***open*** ***creat*** ***chdir*** ***chroot*** ***chown*** ***chmod***	***stat*** ***link*** ***unlink*** ***mknod*** ***mount*** ***umount***	***creat*** ***mknod*** ***link*** ***unlink***	***chown*** ***chmod*** ***stat***	***read*** ***write*** ***cseek***	***mont*** ***umount***	***chdir*** ***chown***

Thuật toán hệ thống file mức thấp

Namei						
iget	***iput***	***Ialloc***	***ifree***	***alloc***	***free***	***bmap***

Thuật toán định vị buffer

getblk	***brelse***	***Bread***	***breada***	***bwrite***

Hình 3.2. Tổng thể về lời gọi hệ thống File

Chúng ta xem xét một số thuật toán làm việc với inode.

Thuật toán truy nhập tới inode (iget)

Nhiều tình huống đòi hỏi thuật toán ***iget***, chẳng hạn như, một quá trình mở một file mới hoặc tạo một file mới v.v.. Thuật toán ***iget*** cấp phát một bản in-core inode đối với một số hiệu inode. Tuy nhiên, trong trường hợp chưa có bản sao in-core inode thì dể có nội dung của nó cần phải đọc được nội dung của inode đó và cần định vị khối dữ liệu chứa inode đã cho. Công thức liên quan đến khối đĩa từ chứa inode để có thể đọc vào bộ nhớ trong như sau:

Chỉ số khối chứa inode = (số hiệu inode - 1) / (số lượng inode trong một khối nhớ) + chỉ số khối nhớ đầu tiên chứa danh sách inode trên đĩa.

Sau khi đã đọc khối đĩa chứa inode vào bộ nhớ trong, để xác định chính xác vị trí của inode, chúng ta có công thức sau:

Byte vị trí đầu tiên = ((số hiệu inode - 1) mod (số lượng inode trong một khối nhớ))*độ dài một inode

Ví dụ, nếu như mỗi inode đĩa chiếm 64 bytes, mỗi khối đĩa chứa 8 inode đĩa thì inode số 8 sẽ bắt đầu từ byte thứ 448 trên khối đĩa đầu tiên trong vùng danh sách các inode.
Để ý rằng, khi làm việc với một hệ thống file thì super block của nó luôn có mặt trong bộ nhớ trong để hệ thống có những thông tin làm việc. Chú ý rằng, trong super block có một danh sách các inode rỗi (trên nó) và một danh sách các khối rỗi.

Thuật toán ***iget*** nhận một inode để cho nó tích cực và điều đó tùy thuộc vào một số tình huống sau đây:

- Nếu inode không tồn tại trong vùng đệm mà lại không thuộc danh sách các inode rỗi trên super block thì hệ thống phải thông báo một lỗi đã được gặp. Lỗi này xảy ra do yêu cầu một inode không còn đủ vùng đệm làm việc với file nữa (tương ứng với trường hợp trong MS-DOS thông báo: too many files opened),
- inode đã có trong vùng đệm các inode trên hệ thống file (đã có in-core inode). Trong trường hợp này xử lý theo hai bước:
  + inode tương ứng đã bị khóa bởi một quá trình khác: lúc đó phải đợi cho đến khi quá trình trước đây không khóa inode nữa. Sau khi được tháo khóa inode có thể trở thành tích cực hoặc rỗi,
  + Nếu inode ở danh sách các inode rỗi thì loại bỏ nó khỏi danh sách này bằng cách đặt inode sang tích cực.
- inode không tồn tại trên vùng đệm tuy nhiên danh sách các inode rỗi khác rỗng. Khi danh sách các inode này khác rỗng, có nghĩa là có những inode không có giá trị: loại bỏ nó và đặt inode mới vào thay thế.

Thuật toán iput loại bỏ inode

Thuật toán ***iput*** có chức năng đối ngẫu với thuật toán ***iget***: cần tháo bớt sự xuất hiện của một inode, chẳng hạn khi chương trình thực hiện thao tác đóng file.

Khác với trường hợp thuật toán ***iget***, thuật toán ***iput*** không nảy sinh tình huống sai sót.

Trong thuật toán này, khi một quá trình không làm việc với một file được liên kết với một inode nữa thì một số tình huống xẩy ra:

- Hệ thống giảm số lượng file tích cực đi 1,
- Nếu số lượng file tích cực là 0 thì:

+ Nếu đó là lệnh xoá file thì trước đó hệ thống đã thực hiện thao tác giảm số liên kết với inode đi 1 và vì vậy có thể số lượng liên kết trở thành 0, có nghĩa là sự tồn tại của file vật lý không còn. Khi đó, chúng ta thực hiện việc xoá thức sự file nói trên bằng một số thao tác: giải phóng các khối dữ liệu, đặt kiểu file của inode là 0 và giải phóng inode.
+ khi số liên kết >0 thi cần cập nhật sự thay đổi của inode lên đĩa từ.

- Trong trường hợp số lượng file tích cực vẫn dương thì không thực hiện thao tác gì.

  Chú ý là trong thuật toán này có sử dụng thuật toán ***ifree***.

Thuật toán ialloc gán inode cho một file mới

Khi một file mới được xuất hiện, chẳng hạn khởi tạo file ***creat***, phải cung cấp một inode cho file và thuật toán ***ialloc*** đáp ứng đòi hỏi trên.

Hoạt động của thuật toán ***ialloc*** được giải thích như sau:

- kiểm tra danh sách inode rỗi trên super block, xảy ra một trong hai trường hợp hoặc danh sách rỗng hoặc không rỗng,
- Nếu danh sách không rỗng thì lấy một inode tiếp theo cho file, khởi tạo các giá trị ban đầu của inode đó và giảm số inode rỗi trên super trên super block.
- Nếu danh sách các inode rỗi trên super block là rỗng: tìm kiếm trên hệ thống file những inode rỗi để tải vào danh sách các inode rỗi trên super block. Nếu danh sách đó đầy hoặc không tìm thấy được nữa thì gán một inode cho file. Nếu danh sách inode rỗi trên super block là rỗng và không tìm thấy inode rỗi trên đĩa thì sẽ có thông báo lỗi.

Trên danh sách các inode rỗi, nhân lưu giữ một inode được gọi là inode nhớ, chính là inode cuối cùng được tìm thấy để sau này thuận lợi cho tìm kiếm.

Thuật toán ifree tải một inode rỗi trên đĩa vào danh sách các inode rỗi trên super block

Thuật toán namei tìm chỉ số một inode theo tên file

Thuật toán ***namei*** là một thuật toán phổ dụng, nhiều thuật toán làm việc với file phải sử dụng ***namei***. Từ tên một đường dẫn file/thư mục, thuật toán ***namei*** cho inode tương ứng.

Thuật toán cấp phát dữ liệu trên đĩa

Khi nhân muốn cấp phát một khối dữ liệu, nó sẽ cấp phát khối rỗi tiếp theo đã được ghi nhận trong super block. Khi một khối dữ liệu đã được cấp cho một file thì nó chỉ được cấp phát lại khi nó trở thành rỗi. Nếu không còn khối rỗng nào trên hệ thống file mà lại có nhu cầu cung cấp khối thì nhân sẽ thông báo lỗi.

3.1.4. Hỗ trợ nhiều hệ thống File

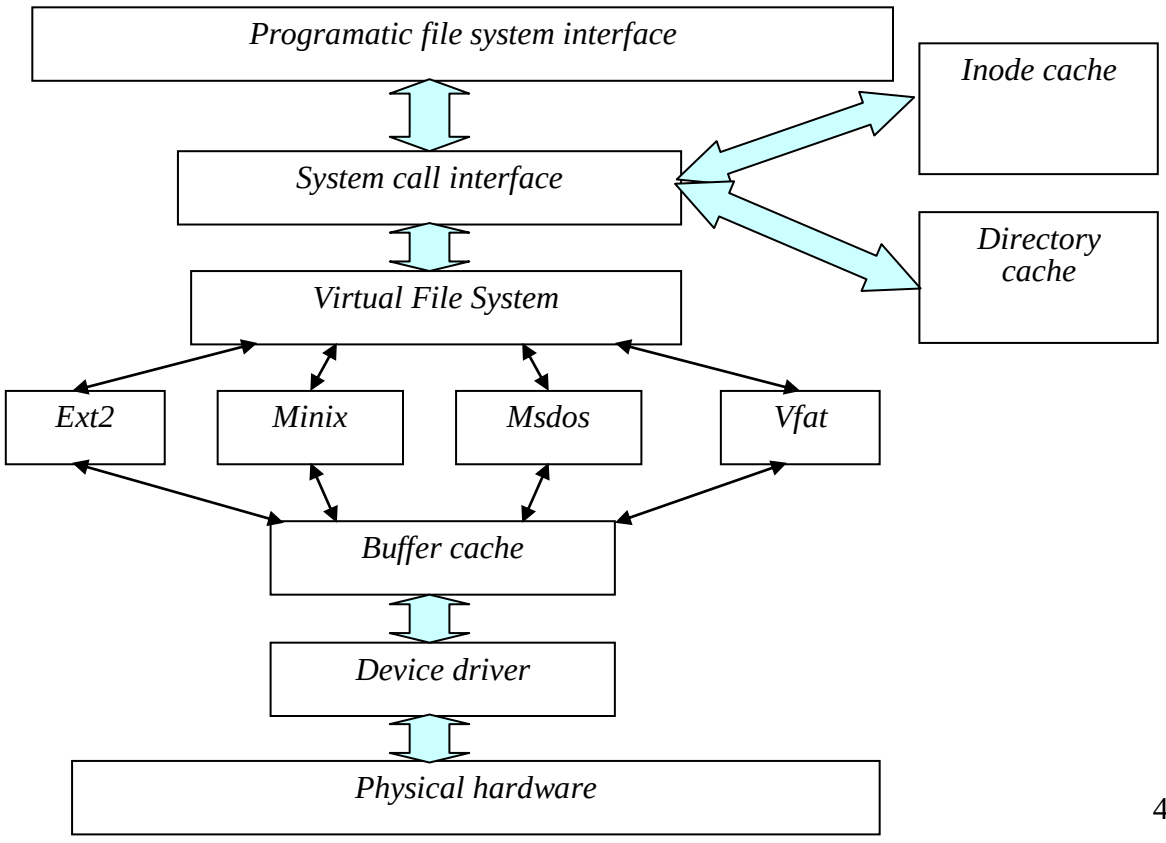
Các phiên bản đầu tiên của Linux chỉ hỗ trợ một hệ thống file duy nhất đó là hệ thống file minix. Sau đó, với sự mở rộng nhân, cộng đồng Linux đã thêm vào nó rất nhiều kiểu hệ thống file khác nhau và Linux trở thành một hệ điều hành hỗ trợ rất nhiều hệ thống file. Dưới đây là một số hệ thống file thông dụng trong các hệ điều hành khác nhau được Linux hỗ trợ.

- Hệ thống file ADFS: ADFS viết tắt của Acorn Disc Filing System là hệ thống file chuẩn trên hệ điều hành RiscOS. Với sự hỗ trợ này, Linux có thể truy cập vào các phân vùng đĩa định dạng theo hệ thống file ADFS.
- Hệ thống file AFFS: AFFS (The Amiga Fast File System) là một hệ thống file phổ biến của hệ điều hành AmigaOS phiên bản 1.3 chạy trên các máy Amiga.
- Hệ thống file CODA: CODA là một hệ thống file mạng cho phép người dùng có thể kết gán các hệ thống file từ xa và truy cập chúng như các hệ thống file cục bộ (local).
- Hệ thống file DEVPTS: Hệ thống file cho Unix98 PTYs.
- Hệ thống file EFS: Đây là một dạng hệ thống file sử dụng cho CDROM.
- Hệ thống file EXT2: Hệ thống file EXT2 (The second extended filesystem) là hệ thống được dùng chủ yếu trên các phiên bản của hệ điều hành Linux. Chúng ta sẽ trở lại ngiên cứu hệ thống file này trong các phần sau.
- Hệ thống file HFS: Đây là hệ thống file chạy trên các máy Apple Macintosh.
- Hệ thống file HPFS: HPFS là hệ thống file được sử dụng trong hệ điều hành OS/2. Linux hỗ trợ hệ thống file này ở mức chỉ đọc (read only).
- Hệ thống file ISOFS: Đây là hệ thống file được sử dụng cho các đĩa CD. Hệ thống thông dụng nhất cho các đĩa CD hiện nay là ISO 9660. Với sự hỗ trợ này, hệ thống Linux có thể truy cập dữ liệu trên các đĩa CD.
- Hệ thống file MINIX: MINIX là hệ thống file đầu tiên mà Linux hỗ trợ. Hệ thống file này được sử dụng trong hệ điều hành Minix và một số hệ thống Linux cũ.
- Hệ thống file MSDOS: Với sự hỗ trợ này, hệ thống Linux có thể truy cập được các phân vùng của hệ điều hành MSDOS. Linux cũng có thể sử dụng kiểu MSDOS để truy cập các phân vùng của Window 95/98 tuy nhiên khi đó, các ưu điểm của hệ điều hành Window sẽ không còn giá trị ví dụ như tên file chỉ tối đa 13 ký tự (kể cả mỏ rộng).
- Hệ thống file NFS: NFS (Network File System) là một hệ thống file trên mạng hỗ trợ việc truy cập dữ liệu từ xa giống như hệ thống file CODA. Với NFS, các máy chạy Linux có thể chia sẻ các phân vùng đĩa trên mạng để sử dụng như là các phân vùng cục bộ của chính máy mình.
- Hệ thống file NTFS: Với sự hỗ trợ này, hệ thống Linux có thể truy cập vào các phân vùng của hệ điều hành Microsoft Window NT.
- Hệ thống file PROC: Đây là một hệ thống file đặc biệt được Linux hỗ trợ. Hệ thống file PROC không chiếm một phân vùng nào của hệ thống và cũng không quản lý các dữ liệu lưu trữ trên đĩa. PROC hiển thị nội dung của chính nhân hệ thống. Các file trong hệ thống file PROC lưu trữ các thông tin về trạng thái hiện hành của nhân. Thông tin về mỗi một tiến trình đang thực hiện trong hệ thống được lưu trong một thư mục mang tên ứng với chỉ số process ID của tiến trình đó. Người dùng có thể sử dụng hệ thống file PROC để lấy các thông tin về nhân cũng như sửa đổi một số giá trị của nhân thông qua sửa đổi nội dung của các file trong hệ thống file này. Tuy nhiên, việc sửa đổi trực tiếp như trên tương đối nguy hiểm, dễ gây đổ vỡ hệ thống.
- Hệ thống file QNX4: Đây là hệ thống file được sử dụng trong hệ điều hành QNX 4.

- Hệ thống file ROMFS: Đây là các hệ thống file chỉ đọc (read only) được sử dụng chủ yếu cho việc khởi tạo đĩa ảo (ramdisk) trong quá trình khởi động đĩa cài đặt.
- Hệ thống file SMB: SMB (Server Mesage Block) là một giao thức của Windows dùng để chia sẻ file giữa các hệ điều hành Windows 95/98, Windows NT và OS/2 Lan Manager. Với sự hỗ trợ SMB, hệ điều hành Linux có thể chia sẻ cũng như truy cập các file nằm trên các phân vùng của một máy chạy các hệ điều hành kể trên. Nói tóm lại, SMB cũng là một dạng hỗ trợ hệ thống file mạng giúp hệ thống có thể chia sẻ với các hệ thống sử dụng chung giao thức SMB.
- Hệ thống file UMSDOS: Hệ thống file UMSDOS (Unix-like MSDOS) là hệ thống file được mở rộng từ hệ thống file MSDOS theo định hướng Unix. Hệ thống file này có một số ưu điểm so với MSDOS như là hỗ trợ tên file dài, hỗ trợ việc phân quyền, hỗ trợ các liên kết (link), hỗ trợ các file đặc biệt (device, pipe ...) và... Hệ thống file này có thể được sử dụng làm phân vùng gốc của hệ thống Linux.
- Hệ thống file VFAT: VFAT chính là hệ thống file mở rộng của hệ thống FAT. Hệ thống file này được sử dụng trong các hệ điều hành Windows 95/98.

Như vậy, ngoài khả năng hỗ trợ nhiều loại thiết bị, Linux còn có khả năng hỗ trợ nhiều kiểu hệ thống file. Bằng cách hỗ trợ nhiều kiểu hệ thống file, Linux có thể truy cập và xử lý các file của nhiều hệ điều hành khác nhau. Mặc dù có khả năng truy cập nhiều hệ thống file khác nhau, hệ thống file của Linux vẫn phải đảm bảo cung cấp cho người dùng một giao diện nhất quán đối với các file, bảo vệ các file trên các hệ thống khác nhau, tối ưu các thao tác truy cập vào thiết bị... Để thực hiện được điều này, Linux sử dụng một hệ thống file đặc biệt gọi là hệ thống file ảo VFS (Virtual File System).

Hệ thống file ảo VFS được thiết kế để cung cấp một giao diện thống nhất về các file được lưu trữ trên các thiết bị. Hình 3.3 mô tả mối quan hệ giữa VFS với các hệ thống file thực và



Hình 3.3. Hệ thống file ảo VFS

các thiết bị lưu trữ.

VFS có trách nhiệm cung cấp cho chương trình người dùng một giao diện nhất quán về hệ thống file thông qua các lệnh gọi hệ thống (system call). Mỗi khi có một yêu cầu truy cập file, VFS sẽ dựa vào các hệ thống file thực để tìm kiếm file yêu cầu trên các thiết bị vật lý. Với mỗi file tìm được, nó thực hiện thao tác mở file đó và cho tương ứng file với một cấu trúc dữ liệu gọi là i-node. VFS cung cấp rất nhiều lệnh gọi để thao tác với hệ thống file nhưng chủ yếu thuộc vào các loại sau:

- Các thao tác liên quan tới hệ thống file.
- Các thao tác liên quan tới i-node.
- Các thao tác với file đang mở.
- Các thao tác với vùng đệm dữ liệu.

3.1.5. Liên kết tượng trưng (lệnh ln)

Trong Linux có hai kiểu liên kết đó là liên kết tượng trưng (liên kết mềm) và liên kết cứng.

"Liên kết cứng" là một cách gọi khác đối với một file đang tồn tại (không có sự phân biệt giữa file gốc và file liên kết). Theo cách nói kỹ thuật, chúng cùng chia sẻ một inode và inode này chứa đựng tất cả các thông tin về file. Không thể tạo một liên kết cứng tới một thư mục.

"Liên kết tượng trưng" là một kiểu file đặc biệt, trong đó, một file liên kết thực sự tham chiếu theo tên đến một file khác. Có thể hiểu kiểu file này như là một con trỏ chỉ dẫn tới một file hoặc một thư mục, và được sử dụng để thay thế cho file hoặc thư mục được trỏ tới. Hầu hết các thao tác (như mở, đọc, ghi ...) được thực hiện trên các file liên kết, sau đó, nhân hệ thống sẽ tự động "tham chiếu" và thực hiện trên file đích của liên kết. Tuy nhiên, có một số các thao tác như xóa file, file liên kết sẽ bị xóa bỏ chứ không phải file đích của nó.

Để tạo một liên kết tượng trưng, hãy sử dụng lệnh **ln** với cú pháp như sau:

```
ln [tùy-chọn] <đích> [tên-nối]
```

Lệnh này sẽ tạo một liên kết đến thư mục/file **đích** với tên file liên kết là **tên-nối**. Nếu **tên-nối** không có, một liên kết với tên file liên kết giống như tên file **đích** sẽ được tạo ra trong thư mục hiện thời.

Các tuỳ chọn của lệnh **ln**:

- **-b, --backup[=CONTROL]** : tạo liên kết quay trở lại cho mỗi file đích đang tồn tại.
- **-f, --force** : xóa bỏ các file đích đang tồn tại.
- **-d, -F, --directory** : tạo liên kết cứng đến các thư mục (tùy chọn này chỉ dành cho người dùng có quyền quản trị hệ thống). Một số phiên bản không có tùy chọn này.
- **-n, --no-dereference** : một file bình thường được xem là đích liên kết từ một thư mục.
- **-i, interactive** : vẫn tạo liên kết dù file đích đã bị xóa bỏ.
- **-s, --symbolic** : tạo các liên kết tượng trưng.
- **--target-directory=<tên-thư-mục>** : xác định thư mục tên-thư-mục là thư mục có chứa các liên kết.
- **-v, --verbose** : hiển thị tên các file trước khi tạo liên kết.

- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ, muốn tạo liên kết đến file **/usr/doc/g77/DOC** với tên file liên kết là **g77manual.txt**, thì gõ lệnh như sau:

```
# ln -s /usr/doc/g77/DOC g77manual.txt
```

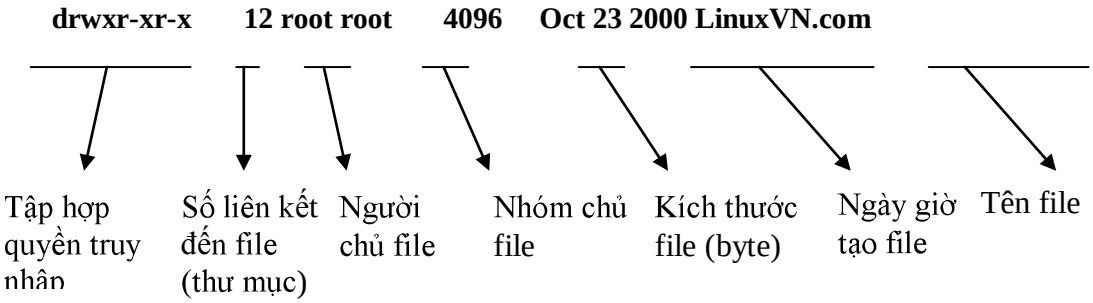
Khi chạy chương trình ***mc***, các file liên kết có tên bắt đầu bởi dấu "ừ", và khi vệt sáng di chuyển đến file liên kết thì tên file được liên kết đến sẽ hiển thị ở bên dưới.

3.2 Quyền truy nhập thư mục và file

3.2.1 Quyền truy nhập

Mỗi file và thư mục trong Linux đều có một chủ sở hữu và một nhóm sở hữu, cũng như một tập hợp các quyền truy nhập. Cho phép thay đổi các quyền truy nhập và quyền sở hữu file và thư mục nhằm cung cấp truy nhập nhiều hơn hay ít hơn.

Thông tin về một file có dạng sau (được hiện ra theo lệnh hiện danh sách file **ls -l**):



Trong đó, dãy 10 ký tự đầu tiên mô tả kiểu file và quyền truy nhập đối với tập tin đó.

Theo mặc định, người dùng tạo một file chính là người chủ (sở hữu) của file đó và là người có quyền sở hữu nó. Người chủ của file có đặc quyền thay đổi quyền truy nhập hay quyền sở hữu đối với file đó. Tất nhiên, một khi đã chuyển quyền sở hữu của mình cho người dùng khác thì người chủ cũ không được phép chuyển quyền sở hữu và quyền truy nhập được nữa.

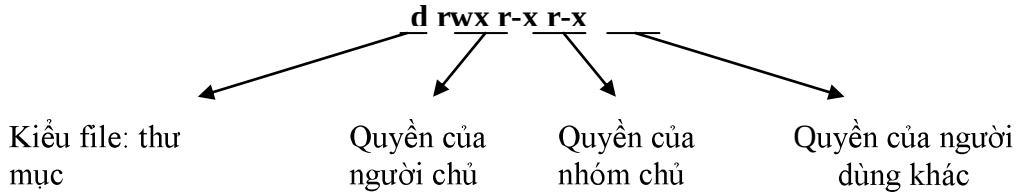
Tập hợp một chuỗi có 10 ký tự đã giới thiệu trên đây được chia ra làm 4 phần: kiểu file, các quyền truy nhập đến file của chủ sở hữu, của nhóm sở hữu và người dùng khác.

Có một số kiểu file trong Linux. Ký tự đầu tiên trong tập hợp 10 ký tự mô tả kiểu file và quyền truy nhập sẽ cho biết file thuộc kiểu nào (chữ cái đó được gọi là chữ cái biểu diễn). Bảng dưới đây sẽ liệt kê các kiểu file trong Linux:

Chữ cái biểu diễn	Kiểu file
d	Thư mục (directory)
b	File kiểu khối (block-type special file)
c	File kiểu ký tự (character-type special file)
l	Liên kết tượng trưng (symbolic link)
p	File đường ống (pipe)
s	Socket
-	File bình thường (regular file)

Chín ký tự tiếp theo trong chuỗi là quyền truy nhập được chia ra làm 3 nhóm tương ứng với quyền truy nhập của người sử hữu, nhóm sở hữu và người dùng khác.

Ví dụ, 10 ký tự đầu tiên trong dòng ví dụ ngay trước đây sẽ được phân tích thành:



Để hiểu được chính xác quyền truy nhập có ý nghĩa như thế nào đối với hệ thống máy tính, phải nhớ rằng **Linux xem mọi thứ đều là file**. Nếu cài đặt một ứng dụng, nó cũng sẽ được xem như mọi chương trình khác, trừ một điều: hệ thống nhận biết rằng một ứng dụng là một chương trình khả thi, tức là nó có thể chạy được. Một bức thư gửi cho mẹ là một dạng file văn bản bình thường, nhưng nếu thông báo cho hệ thống biết đó là một chương trình khả thi, hệ thống sẽ cố để chạy chương trình (và tất nhiên là lỗi).

Có ba loại quyền truy nhập chính đối với thư mục/file, đó là: đọc (read - r), ghi (write - w) và thực hiện (execute - x). Quyền đọc cho phép người dùng có thể xem nội dung của file với rất nhiều chương trình khác nhau, nhưng họ sẽ không thể thay đổi, sửa chữa hoặc xóa bất kỳ thông tin nào trong đó. Tuy nhiên, họ có thể sao chép file đó thành file của họ và sửa chữa file bản sao.

Quyền ghi là quyền truy nhập tiếp theo. Người sử dụng với quyền ghi khi truy nhập vào file có thể thêm thông tin vào file. Nếu có quyền ghi và quyền đọc đối với một file, có thể soạn thảo lại file đó - quyền đọc cho phép xem nội dung, và quyền ghi cho phép thay đổi nội dung file. Nếu chỉ có quyền ghi, sẽ thêm được thông tin vào file, nhưng lại không thể xem được nội dung của file.

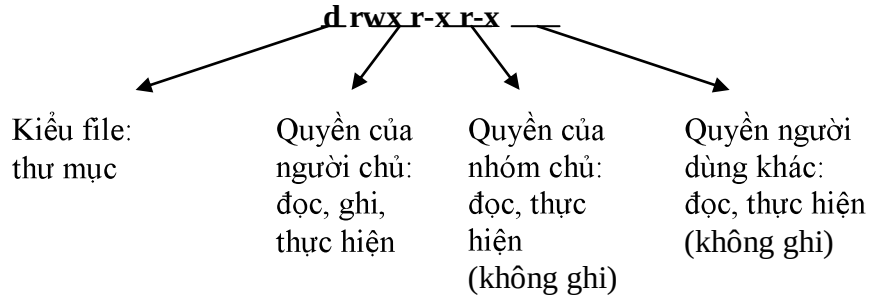
Loại quyền truy nhập thứ ba là quyền thực hiện, quyền này cho phép người dùng có thể chạy được file, nếu đó là một chương trình khả thi. Quyền thực hiện độc lập với các quyền truy nhập khác, vì thế hoàn toàn có thể có một chương trình với quyền đọc và quyền thực hiện, nhưng không có quyền ghi. Cũng có trường hợp một chương trình chỉ có quyền thực hiện, có nghĩa là người dùng có thể chạy ứng dụng, nhưng họ không thể xem được cách nó làm việc hay sao chép nó.

Bảng dưới đây giới thiệu cách ký hiệu của các quyền truy nhập:

Quyền truy nhập	Ý nghĩa
---	Không cho phép một quyền truy nhập nào
r--	Chỉ được quyền đọc
r-x	Quyền đọc và thực hiện (cho chương trình và shell script)
rw-	Quyền đọc và ghi
rwx	Cho phép tất cả các quyền truy nhập (cho chương trình)

Tuy nhiên, đối với thư mục thì chỉ có ba loại ký hiệu của các quyền truy nhập là: **---**, **r-x** và **rwx**, vì nội dung của thư mục là danh sách của các file và các thư mục con có bên trong thư mục đó. Quyền đọc một thư mục là được xem nội dung của thư mục đó và quyền thực hiện đối với một thư mục là quyền tìm được file và thư mục con có trong thư mục.

Như vậy, với ví dụ đang được xem xét, chúng ta nhận được đây là một thư mục và quyền truy nhập nó được giải thích như sau:



☞ ***Giải thích:***

Sự hạn chế trường hợp về quyền truy nhập thư mục được giải thích theo các lập luận như sau:

- Hãy hình dung, giả sử chỉ có quyền đọc trên thư mục, khi đó sẽ xem được có những file hay thư mục nào trong thư mục nhưng lại không thể xem cụ thể nội dung của một file hay thư mục có trên thư mục đó vì không tìm được nó.
- Hoặc giả sử có quyền thực hiện - quyền này sẽ cho phép tìm được file có trên thư mục - nhưng lại không có quyền đọc đối với một thư mục, vậy thì làm thế nào để biết được trong thư mục có những file nào.

3.2.2. Các lệnh cơ bản

a. Thay đổi quyền sở hữu file với lệnh *chown*

Để thay đổi quyền sở hữu đối với một file, hãy sử dụng lệnh **chown** với cú pháp như sau:

```
chown [tùy-chọn] [chủ][.nhóm] <file ...>
```

Lệnh này cho phép thay chủ sở hữu **file**. Nếu chỉ có tham số về ***chủ***, thì người dùng ***chủ*** sẽ có quyền sở hữu file và nhóm sở hữu không thay đổi. Nếu theo sau tên người **chủ** là dấu "**.**" và tên của một ***nhóm*** thì nhóm đó sẽ nhóm sở hữu file. Nếu chỉ có dấu "**.**" và ***nhóm*** mà không có tên người chủ thì chỉ có quyền sở hữu nhóm của file thay đổi, lúc này, lệnh **chown** có tác dụng giống như lệnh **chgrp** (lệnh **chgrp** được trình bày dưới đây).
Các tùy chọn của lệnh **chown**:

- **-c, --changes** : hiển thị dòng thông báo chỉ với các file mà lệnh làm thay đổi sở hữu (số thông báo hiện ra có thể ít hơn trường hợp **-v, -verbosr**).
- **-f, --silent, --quiet** : bỏ qua hầu hết các thông báo lỗi.
- **-R, --recursive** : thực hiện đổi quyền sở hữu đối với thư mục và file theo đệ quy.
- **-v, --verbose** : hiển thị dòng thông báo với mọi file liên quan mà chown tác động tới (có hoặc không thay đổi sở hữu).
- **--help** : đưa ra trang trợ giúp và thoát.

Ví dụ, thư mục **LinuxVN.com** có thông tin về các quyền truy nhập như sau:

drwxr-xr-x 12 thu root 4096 Oct 23 2000 LinuxVN.com

Người sở hữu hiện tại thư mục **LinuxVN.com** là người dùng ***thu***. Để người dùng ***lan*** là chủ sở hữu thư mục trên, hãy gõ lệnh:

```
# chown lan LinuxVN.com
```

Khi đó, nếu dùng lệnh **ls** thì thông tin về thư mục **LinuxVN.com** sẽ có dạng:

```
drwxr-xr-x 12 lan root 4096 Oct 23 2000 LinuxVN.com
```

với người sở hữu thư mục bây giờ là người dùng ***lan***.

Khi chuyển quyền sở hữu file cho một người khác, người chủ cũ mất quyền sở hữu file đó.

b. Thay đổi quyền sở hữu nhóm với lệnh *chgrp*

Các file (và người dùng) còn thuộc vào các nhóm, đây là phương thức truy nhập file thuận tiện cho nhiều người dùng nhưng không phải tất cả người dùng trên hệ thống. Khi đăng nhập, mặc định sẽ là thành viên của một nhóm được thiết lập khi siêu người dùng root tạo tài khoản người dùng. Cho phép một người dùng thuộc nhiều nhóm khác nhau, nhưng mỗi lần đăng nhập chỉ là thành viên của một nhóm.

Để thay đổi quyền sở hữu nhóm đối với một hoặc nhiều file, hãy sử dụng lệnh **chgrp** với cú pháp như sau:

```
chgrp [tùy-chọn] {nhóm|--reference=nhómR} <file...>
```

Lệnh này cho phép thay thuộc tính nhóm sở hữu của file theo tên nhóm được chỉ ra trực tiếp theo tham số **nhóm** hoặc gián tiếp qua thuộc tính nhóm của file có tên là **nhómR**.

Các tùy chọn của lệnh là (một số tương tự như ở lệnh **chown**):

- **-c, --changes** : hiển thị dòng thông báo chỉ với các file mà lệnh làm thay đổi sở hữu (số thông báo hiện ra có thể ít hơn trường hợp **-v, -verbosr**).
- **-f, --silent, --quiet** : bỏ qua hầu hết các thông báo lỗi.
- **-R, --recursive** : thực hiện đổi quyền sở hữu đối với thư mục và file theo đệ quy.
- **-v, --verbose** : hiển thị dòng thông báo với mọi file liên quan mà chgrp tác động tới (có hoặc không thay đổi sở hữu).
- **--help** : hiển thị trang trợ giúp và thoát

Tham số --reference=nhómR cho thấy cách gián tiếp thay nhóm chủ của file theo nhóm chủ của một file khác (tên là nhómR) là cách thức được ưa chuộng hơn. Tham số này là xung khắc với tham số nhóm của lệnh.

c. Thay đổi quyền truy cập file với lệnh *chmod*

Cú pháp lệnh **chmod** có ba dạng:

```
chmod [tùy-chọn] <mod [,mod]...> <file...>
chmod [tùy-chọn] <mod-hệ-8> <file...>
chmod [tùy-chọn] --reference=nhómR <file...>
```

Lệnh **chmod** cho phép xác lập quyền truy nhập theo kiểu (**mode)** trên **file**. Dạng đầu tiên là dạng xác lập tương đối, dạng thứ hai là dạng xác lập tuyệt đối và dạng cuối cùng là dạng gián tiếp chỉ dẫn theo quyền truy nhập của file **nhómR**.

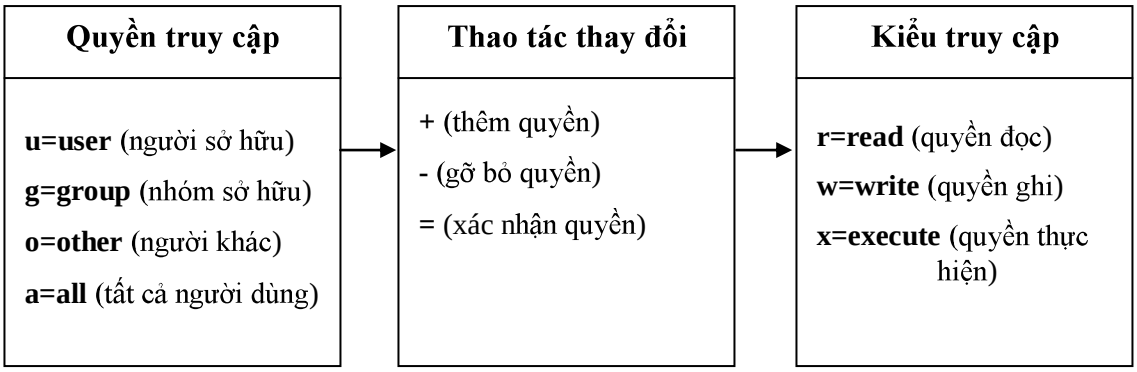
Các tùy chọn của lệnh **chmod** được liệt kê như dưới đây và có ý nghĩa tương tự các tuỳ chọn tương ứng của các lệnh **chown**, **chgrp**:

- -c, --changes
- -f, --silent, --quiet
- -v, --verbose
- -R, --recursive
- --help
- và tham số **--reference=RFILE** cũng ý nghĩa gián tiếp như trong lệnh **chgrp**.

Giải thích về hai cách xác lập quyền truy nhập file trong lệnh **chmod** như sau: xác lập tuyệt đối (dùng hệ thống mã số viết theo hệ cơ số 8 biểu diễn cho các quyền truy nhập) và xác lập tương đối (dùng các chữ cái để biểu diễn quyền truy nhập).

Cách xác lập tương đối

Cách xác lập tương đối là dễ nhớ theo ý nghĩa của nội dung các **mod** và chỉ những thay đổi thực sự mới được biểu diễn trong lệnh. Ba hộp sau đây sẽ giải thích các chữ cái biểu diễn **mod** theo cách xác lập tương đối.



Có thể kết hợp các mục từ hộp thứ nhất và hộp thứ ba với một mục từ hộp thứ hai để tạo ra một **mod**.

Ví dụ, nếu muốn thêm quyền ghi đối với file ***test*** cho tất cả người dùng trong nhóm sở hữu, hãy chọn **g** cho nhóm sở hữu, **+** cho thêm quyền truy nhập, và **w** cho quyền ghi. Lúc đó lệnh **chmod** sẽ có dạng sau:

```
chmod g+w test
```

Cách xác lập tương đối trong lệnh **chmod** gần giống như một menu có nhiều mục chọn khác nhau, cho phép kết hợp để có được sự lựa chọn theo yêu cầu.

Nếu quyết định gỡ bỏ quyền đọc và thực hiện trên file ***test*** cho những người không cùng nhóm, hãy chọn **o** cho người dùng khác, **-** để gỡ bỏ quyền truy nhập, và r,x cho quyền đọc và thực hiện. Lệnh **chmod** sẽ là:

```
chmod o-rx test
```

Cách xác lập tuyệt đối

Đối với người dùng hiểu sơ bộ về biểu diễn số trong hệ cơ số 8 thì cách xác lập tuyệt đối lại được ưa chuộng hơn.

Phần 3.2.1. cho biết biểu diễn quyền truy nhập file thông qua dãy gồm 9 vị trí dưới dạng **rwxrwxrwx**, trong đó từng cụm 3 vị trí theo thứ tự tương ứng với: chủ sở hữu, nhóm sở hữu và người dùng khác. Như vậy thuộc tính quyền truy nhập của một file có thể biểu

diễn thành 9 bít nhị phân trong đó bít có giá trị 1 thì quyền đó được xác định, ngược lại thì quyền đó bị tháo bỏ. Như vậy, chủ sở hữu tương ứng với 3 bít đầu tiên, nhóm sỡ hữu tương ứng với 3 bít giữa, người dùng khác tương ứng với 3 bít cuối. Mỗi cụm 3 bít như vậy cho một chữ số hệ 8 (nhận giá trị từ 0 đến 7) và thuộc tính quyền truy nhập tương ứng với 3 chữ số hệ 8.

Ví dụ, cặp 3 số hệ 8 là 755 tương ứng với dòng 9 bít 111101101 với 111 cho chủ sở hữu, 101 cho nhóm sở hữu, 101 cho người dùng khác. Ví dụ lệnh:

```
chmod 753 memo1
```

đặt thuộc tính quyền truy nhập đối với file memo1 là **rwxr-xr-x**. Để dễ xác lập 3 chữ số hệ 8 áp dụng cách tính như sau:

Quyền	*Chữ số hệ 8*	*Quyền*	*Chữ số hệ 8*
Chỉ đọc	4	Chỉ đọc và ghi	6
Chỉ ghi	2	Chỉ đọc và thực hiện	5
Chỉ thực hiện	1	Chỉ ghi và thực hiện	3
Không có quyền nào	0	Đọc, ghi và thực hiện	7

d. đăng nhập vào một nhóm người dùng mới với lệnh *newgrp*

Linux cho phép một người dùng có thể là thành viên của một hoặc nhiều nhóm người dùng khác nhau, trong đó có một nhóm được gọi là nhóm khởi động. Điều này được đảm bảo khi thực hiện lệnh **adduser** hoặc **usersdd**. Tuy nhiên, tại một thời điểm, một người dùng thuộc vào chỉ một nhóm. Khi một người dùng đăng nhập, hệ thống ngầm định người dùng đó là thành viên của nhóm khởi động, và có quyền truy nhập đối với những file thuộc quyền sở hữu của nhóm khởi động đó. Nếu muốn sử dụng quyền sở hữu theo các nhóm khác đối với những file thì người dùng phải chuyển đổi thành thành viên của một nhóm những nhóm đã được gắn với người dùng. Lệnh **newgr** cho phép người dùng chuyển sang nhóm người dùng khác đã gắn với mình với cú pháp:

```
newgrp [nhóm]
```

trong đó **nh□m** là một tên nhóm người dùng tồn tại trong hệ thống.

Ví dụ, một người dùng là thành viên của hai nhóm ***user*** và ***installer***, với ***user*** là nhóm khởi động. Khi đăng nhập, người dùng đó có tư cách là thành viên của nhóm ***user***. Khi mong muốn sử dụng một số các chương trình thuộc quyền sở hữu của nhóm ***installer***, người dùng cần gõ lệnh sau:

```
# newgrp installer
```

Nếu người dùng nói trên cố chuyển vào một nhóm mà người dùng đó không là thành viên, chẳng hạn dùng lệnh:

```
# newgrp hot2
```

thì Linux sẽ đưa ra một khuyến cáo thân thiện như sau:

```
newgrp: Sorry
```

3.3 Thao tác với thư mục

Như đã được giới thiệu (mục 3.1.1.), Linux tổ chức hệ thống file theo cách sử dụng các thư mục. Mục này bắt đầu bằng việc giới thiệu một số thư mục chính và tác dụng của chúng trong hệ thống Linux. Sau đó một số lệnh thao tác với thư mục cơ bản nhất được trình bày.

3.3.1 Một số thư mục đặc biệt

*** Thư mục gốc /**

Đây là thư mục gốc chứa đựng tất cả các thư mục con có trong hệ thống.

*** Thư mục */root***

Thư mục **/root** có thể được coi là "thư mục riêng" của siêu người dùng. Thư mục này được sử dụng để lưu trữ các file tạm thời, nhân Linux và ảnh khởi động, các file nhị phân quan trọng (những file được sử dụng đến trước khi Linux có thể gắn kết đến phân vùng **/user**), các file đăng nhập quan trọng, bộ đệm in cho việc in ấn, hay vùng lưu tạm cho việc nhận và gửi email. Nó cũng được sử dụng cho các vùng trống tạm thời khi thực hiện các thao tác quan trọng, ví dụ như khi xây dựng (**build**) một gói RPM từ các file RPM nguồn.

*** Thư mục */bin***

Trong Linux, chương trình được coi là khả thi nếu nó có thể thực hiện được. Khi một chương trình được biên dịch, nó sẽ có dạng là file nhị phân. Như vậy, chương trình ứng dụng trong Linux là một file nhị phân khả thi.

Chính vì lẽ đó, những nhà phát triển Linux đã quyết định phải tổ chức một thư mục "binaries" để lưu trữ các chương trình khả thi có trên hệ thống, đó chính là thư mục **/bin**.

Ban đầu, thư mục **/bin** (**bin** là viết tắt của từ **binary**) là nơi lưu trữ các file nhị phân khả thi. Nhưng theo thời gian, ngày càng có nhiều hơn các file khả thi có trong Linux, do đó, có thêm các thư mục như **/sbin**, **/usr/bin** được sử dụng để lưu trữ các file đó.

*** Thư mục */dev***

Một phần không thể thiếu trong bất kỳ máy tính nào đó là các trình điều khiển thiết bị. Không có chúng, sẽ không thể có được bất kỳ thông tin nào trên màn hình của (các thông tin có được do trình điều khiển thiết bị hiển thị đưa ra). Cũng không thể nhập được thông tin (những thông tin do trình điều khiển thiết bị bàn phím đọc và chuyển tới hệ thống), và cũng không thể sử dụng đĩa mềm của (được quản lý bởi trình điều khiển đĩa mềm).

Tất cả các trình điều khiển thiết bị đều được lưu trữ trong thư mục **/dev**.

*** Thư mục */etc***

Quản trị hệ thống trong Linux không phải là đơn giản, chẳng hạn như việc quản lý tài khoản người dùng, vấn đề bảo mật, trình điều khiển thiết bị, cấu hình phần cứng, v.v.. Để giảm bớt độ phức tạp, thư mục **/etc** đã được thiết kế để lưu trữ tất cả các thông tin hay các file cấu hình hệ thống.

*** Thư mục */lib***

Linux có một trung tâm lưu trữ các thư viện hàm và thủ tục, đó là thư mục **/lib**.

*** Thư mục */lost+found***

Một file được khôi phục sau khi có bất kỳ một vấn đề hoặc gặp một lỗi về ghi đĩa trên hệ thống đều được lưu vào thư mục này.

*** Thư mục */mnt***

Thư mục **/mnt** là nơi để kết nối các thiết bị (ví dụ đĩa cứng, đĩa mềm...) vào hệ thống file chính nhờ lệnh **mount**. Thông thường các thư mục con của **/mnt** chính là gốc của các hệ thống file được kết nối: **/mnt/floppy**: đĩa mềm, **/mnt/hda1**: vùng đầu tiên của đĩa cứng thứ nhất (**hda**), **/mnt/hdb3**: vùng thứ ba của đĩa cứng thứ 2 (**hdb**) ...

*** Thư mục */tmp***

Thư mục **/tmp** được rất nhiều chương trình trong Linux sử dụng như một nơi để lưu trữ các file tạm thời.

Ví dụ, nếu đang soạn thảo một file, chương trình sẽ tạo ra một file là bản sao tạm thời (bản nháp) của file đó và lưu vào trong thư mục **/tmp**. Việc soạn thảo thực hiện trực tiếp trên file tạm thời này và sau khi soạn thảo xong, file tạm thời sẽ được ghi đè lên file gốc. Cách thức như vậy bảo đảm sự an toàn đối với file cần soạn thảo.

*** Thư mục */usr***

Thông thường thì thư mục **/usr** là trung tâm lưu trữ tất cả các lệnh hướng đến người dùng (user-related commands). Tuy nhiên, ngày nay thật khó xác định trong thư mục này có những thứ gì, bởi vì hầu hết các file nhị phân cần cho Linux đều được lưu trữ ở đây, trong đó đáng chú ý là thư mục con **/usr/src** bao gồm các thư mục con chứa các chương trình nguồn của nhân Linux.

*** Thư mục */home***

Thư mục này chứa các thư mục cá nhân của người dùng: mỗi người dùng tương ứng với một thư mục con ở đây, tên người dùng được lấy làm tên của thư mục con.

*** Thư mục */var***

Thư mục **/var** được sử dụng để lưu trữ các file chứa các thông tin luôn luôn thay đổi, bao gồm bộ đệm in, vùng lưu tạm thời cho việc nhận và gửi thư (mail), các khóa quá trình, v.v..

*** Thư mục */boot***

Là thư mục chứa nhân của hệ thống (**Linux-*.*.**), **System.map** (file ánh xạ đến các ***driver*** để nạp các hệ thống file khác), ảnh (image) của hệ thống file dùng cho **initrd** (**ramdisk**), trình điều khiển cho các thiết bị RAID (một thiết bị gồm một mảng các ổ đĩa cứng để tăng tốc độ và độ an toàn khi ghi dữ liệu), các bản sao lưu **`boot record`** của các phân vùng đĩa khác. Thư mục này cho phép khởi động và nạp lại bất kỳ trình điều khiển nào được yêu cầu để đọc các hệ thống file khác.

*** Thư mục */proc***

Đây là thư mục dành cho nhân (**kernel**) của hệ điều hành và thực tế đây là một hệ thống file độc lập do nhân khởi tạo.

*** Thư mục */misc* và thư mục */opt***

Cho phép lưu trữ mọi đối tượng vào hai thư mục này.

*** Thư mục */sbin***

Thư mục lưu giữ các file hệ thống thường tự động chạy.

3.3.2 Các lệnh cơ bản về thư mục

*** Xác định thư mục hiện thời với lệnh *pwd***

Cú pháp lệnh:

```
pwd
```

Lệnh này cho biết hiện người dùng đang ở trong thư mục nào và hiện ra theo dạng một đường dẫn tuyệt đối.

Ví dụ, gõ lệnh **pwd** tại dấu nhắc lệnh sau khi người dùng **lan** vừa đăng nhập thì màn hình hiển thị như sau:

```
# pwd
/home/lan
#
```

*** Xem thông tin về thư mục với lệnh *ls***

Sử dụng lệnh **ls** và một số các tùy chọn của nó là có thể biết được mọi thông tin về một thư mục. Cú pháp lệnh:

```
ls [tùy-chọn] [file]...
```

Lệnh này đưa ra danh sách các file liên quan đến tham số ***file*** trong lệnh. Trường hợp phổ biến tham số file là một thư mục, tuy nhiên trong một số trường hợp khác, tham số ***file*** xác định nhóm (khi sử dụng các mô tả nhóm *, ? và cặp [và])**;** nếu không có tham số ***file***, mặc định danh sách các file có trong thư mục hiện thời sẽ được hiển thị.

Các tùy chọn của lệnh:

- **-a** : liệt kê tất cả các file, bao gồm cả file ẩn.
- **-l** : đưa ra thông tin đầy đủ nhất về các file và thư mục.
- **-s** : chỉ ra kích thước của file, tính theo khối (1 khối = 1204 byte).
- **-F** : xác định kiểu file (/ = thư mục, * = chương trình khả thi).
- **-m** : liệt kê các file được ngăn cách nhau bởi dấu ",".
- **-C** : đưa ra danh sách các file và thư mục theo dạng cột (hai thư mục gần nhau được xếp vào một cột).
- **-1** : hiển thị mỗi file hoặc thư mục trên một dòng.
- **-t** : sắp xếp các file và thư mục trong danh sách theo thứ tự về thời gian được sửa đổi gần đây nhất.
- **-x** : đưa ra danh sách các file và thư mục theo dạng cột (hai thư mục gần nhau được xếp trên hai dòng đầu của hai cột kề nhau).
- **-r** : sắp xếp danh sách hiển thị theo thứ tự ngược lại.
- **-R** : liệt kê lần lượt các thư mục và nội dung của các thư mục.

Ví dụ, lệnh

```
# ls -l
```

sẽ hiển thị danh sách đầy đủ nhất về các file và thư mục có trong thư mục hiện thời.

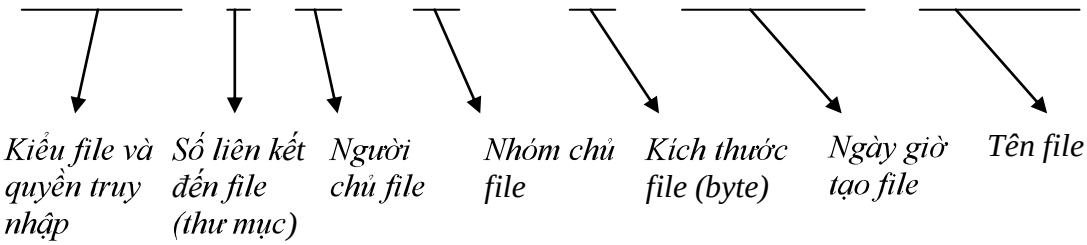
```
total 108
drwxr-xr-x 12 thu root 4096 Oct 23 2000 LinuxVN.com
drwxr-xr-x 2 root root 4096 Oct 31 2000 bin
drwxr-xr-x 2 root root 4096 Dec 11 16:54 boot
drwxr-xr-x 7 root root 36864 Dec 11 16:54 dev
drwxr-xr-x 43 root root 4096 Dec 11 16:55 etc
drwxr-xr-x 5 root root 4096 Dec 11 16:57 home
drwxr-xr-x 4 root root 4096 Oct 31 2000 lib
drwxr-xr-x 2 root root 16384 Oct 31 2000 lost+found
drwxr-xr-x 2 root root 0 Dec 11 16:54 misc
drwxr-xr-x 5 root root 4096 Oct 31 2000 mnt
drwxr-xr-x 2 root root 4096 Aug 23 12:03 opt
dr-xr-xr-x 56 root root 0 Dec 11 11:54 proc
drwxr-x--- 12 root root 4096 Dec 11 16:55 root
drwxr-xr-x 3 root root 4096 Oct 31 2000 sbin
drwxr-xr-x 3 root root 4096 Oct 31 2000 tftpboot
drwxrwxrwx 8 root root 4096 Dec 11 16:58 tmp
drwxr-xr-x 22 root root 4096 Oct 31 2000 usr
drwxr-xr-x 22 root root 4096 Oct 31 2000 var
```

Dòng đầu tiên "**`total 108`**" cho biết tổng số khối (1024 byte) trên đĩa lưu trữ các file trong danh sách (14*4+36+16=108).

Mỗi dòng tiếp theo trình bày thông tin về mỗi file hay thư mục con. Các thông tin này đã được giới thiệu trước đây.

Chẳng hạn,

drwxr-xr-x 22 root root 4096 Oct 31 2000 var



ý nghĩa của mỗi trường trên đây đã được giải thích trong mục 3.2.1.

Khi gõ lệnh:

```
# ls [is]*
```

cho danh sách các file và thư mục con có tên bắt đầu bằng hoặc chữ cái **i** hoặc chữ cái **s** có trong thư mục hiện thời:

```
info-dir        initlog.conf    inittab         services
shadow          shadow-         shells          smb.conf
```

sysctl.conf syslog.conf

*** Lệnh tạo thư mục *mkdir***

Lệnh **mkdir** tạo một thư mục với cú pháp:

```
mkdir [tùy-chọn] <thư-mục>
```

Lệnh này cho phép tạo một thư mục mới nếu thư mục đó chưa thực sự tồn tại. Để tạo một thư mục, cần đặc tả tên và vị trí của nó trên hệ thống file (vị trí mặc định là thư mục hiện thời). Nếu thư mục đã tồn tại, hệ thống sẽ thông báo cho biết.

Các tùy chọn:

- **-m, --mode=Mod** : thiết lập quyền truy nhập **Mod** như trong lệnh **chmod** nhưng không cho quyền **rwxrwxrwx**.
- **-p, --parents** : tạo các thư mục cần thiết mà không thông báo lỗi khi nó đã tồn tại.
- **--verbose** : hiển thị các thông báo cho mỗi thư mục được tạo.
- **--help** : đưa ra trang trợ giúp và thoát.

Ví dụ, nếu muốn tạo thư mục **test** trong thư mục **home**, hãy gõ lệnh sau:

```
# mkdir /home/test
```

*** Lệnh xóa bỏ thư mục *rmdir***

Như đã biết, lệnh **mkdir** để tạo ra một thư mục mới, và về đối ngẫu thì lệnh **rmdir** được dùng để xóa bỏ một thư mục. Cú pháp lệnh:

```
rmdir [tùy-chọn[ <thư-mục>
```

Có thể xóa bỏ bất kỳ thư mục nào nếu có quyền đó. Lưu ý rằng, thư mục chỉ bị xóa khi nó "rỗng", tức là không tồn tại file hay thư mục con nào trong đó.

Không có cách gì khôi phục lại các thư mục đã bị xóa, vì thế hãy suy nghĩ cẩn thận trước khi quyết định xóa một thư mục.

Các tùy chọn của lệnh:

- **--ignore-fail-on-non-empty** : bỏ qua các lỗi nếu xóa một thư mục không rỗng.
- **-p, --parents** : xóa bỏ một thư mục, sau đó lần lượt xóa bỏ tiếp các thư mục có trên đường dẫn chứa thư mục vừa xóa. Ví dụ, dòng lệnh **rmdir -p /a/b/c** sẽ tương đương với ba dòng lệnh **rmdir /a/b/c**, **rmdir /a/b**, **rmdir /a** (với điều kiện các thư mục là rỗng).
- **--verbose** : đưa ra thông báo khi xóa một thư mục.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# rmdir -p /test/test1/test2
rmdir: /: No such file or directory
#
```

Dòng lệnh trên sẽ lần lượt xóa ba thư mục **test2**, **test1**, **test** và hiển thị thông báo trên màn hình kết quả của lệnh.

*** Lệnh đổi tên thư mục *mv***

Cú pháp lệnh:

```
mv <tên-cũ> <tên-mới>
```

Lệnh này cho phép đổi tên một thư mục từ **tên-cũ** thành **tên-mới**.

Ví dụ, lệnh

```
# mv LinuxVN.com LinuxVN
```

sẽ đổi tên thư mục **LinuxVN.com** thành **LinuxVN**.

Nếu sử dụng lệnh **mv** để đổi tên một thư mục với một cái tên đã được đặt cho một file thì lệnh sẽ gặp lỗi.

Nếu tên mới trùng với tên một thư mục đang tồn tại thì nội dung của thư mục được đổi tên sẽ ghi đè lên nội dung của thư mục trùng tên.

3.4. Các lệnh làm việc với file

3.4.1 Các kiểu file có trong Linux

Mục 3.1.2. đã trình bày sơ lược về kiểu của các file. Như đã được giới thiệu, có rất nhiều file khác nhau trong Linux, nhưng bao giờ cũng tồn tại một số kiểu file cần thiết cho hệ điều hành và người dùng, dưới đây giới thiệu lại một số các kiểu file cơ bản.

- ***File người dùng (user data file):*** là các file tạo ra do hoạt động của người dùng khi kích hoạt các chương trình ứng dụng tương ứng. Ví dụ như các file thuần văn bản, các file cơ sở dữ liệu hay các file bảng tính.
- ***File hệ thống (system data file):*** là các file lưu trữ thông tin của hệ thống như: cấu hình cho khởi động, tài khoản của người dùng, thông tin thiết bị ... thường được cất trong các tệp dạng văn bản để người dùng có thể can thiệp, sửa đổi theo ý mình.
- ***File thực hiện (executable file):*** là các file chứa mã lệnh hay chỉ thị cho máy tính thực hiện. File thực hiện lưu trữ dưới dạng mã máy mà ta khó có thể tìm hiểu được ý nghĩa của nó, nhưng tồn tại một số công cụ để "hiểu" được các file đó. Khi dùng trình ứng dụng **mc** (Midnight Commander, chương 8), file thực hiện được bắt đầu bởi dấu (*) và thường có màu xanh lục.
- ***Thư mục hay còn gọi là file bao hàm (directory):*** là file bao hàm các file khác và có cấu tạo hoàn toàn tương tự như file thông thường khác nên có thể gọi là file. Trong **mc**, file bao hàm thường có màu trắng và bắt đầu bằng dấu ngã (**~**) hoặc dấu chia (**/**). Ví dụ: **/**, **/home**, **/bin**, **/usr**, **/usr/man**, **/dev** ...
- ***File thiết bị (device file):*** là file mô tả thiết bị, dùng như là định danh để chỉ ra thiết bị cần thao tác. Theo quy ước, file thiết bị được lưu trữ trong thư mục **/dev**. Các file thiết bị hay gặp trong thư mục này là **tty** (teletype - thiết bị truyền thông), **ttyS** (teletype serial - thiết bị truyền thông nối tiếp), **fd0**, **fd1**, ... (floppy disk- thiết bị ổ đĩa mềm), **hda1**, **hda2**, ... **hdb1**, **hdb2**, ... (hardisk - thiết bị ổ cứng theo chuẩn IDE; a, b,... đánh số ổ đĩa vật lý; 1, 2, 3... đánh số ổ logic). Trong **mc**, file thiết bị có màu tím và bắt đầu bằng dấu cộng (+).
- ***File liên kết (linked file):*** là những file chứa tham chiếu đến các file khác trong hệ thống tệp tin của Linux. Tham chiếu này cho phép người dùng tìm nhanh tới file thay vì tới vị trí nguyên thủy của nó. Hơn nữa, người ta có thể gắn vào đó các thông tin phụ trợ làm cho file này có tính năng trội hơn so với tính năng

nguyên thủy của nó. Ta thấy loại file này giống như khái niệm ***shortcut*** trong MS-Windows98.

Không giống một số hệ điều hành khác (như MS-DOS chẳng hạn), Linux quản lý thời gian của tệp tin qua các thông số thời gian truy nhập (accesed time), thời gian kiến tạo (created time) và thời gian sửa đổi (modified time).

3.4.2. Các lệnh tạo file

Trong Linux có rất nhiều cách để tạo file, sau đây là các cách hay được dùng.

*** Tạo file với lệnh *touch***

Lệnh **touch** có nhiều chức năng, trong đó một chức năng là giúp tạo file mới trên hệ thống: **touch** rất hữu ích cho việc tổ chức một tập hợp các file mới. Cú pháp lệnh:

```
touch <file>
```

Thực chất lệnh này có tác dụng dùng để cập nhật thời gian truy nhập và sửa chữa lần cuối của một file. Vì lý do này, các file được tạo bằng lệnh **touch** đều được sắp xếp theo thời gian sửa đổi. Nếu sử dụng lệnh **touch** đối với một file chưa tồn tại, chương trình sẽ tạo ra file đó. Sử dụng bất kỳ trình soạn thảo nào để soạn thảo file mới.

Ví dụ, dùng lệnh **touch** để tạo file **newfile:**

```
# touch newfile
```

*** Tạo file bằng cách đổi hướng đầu ra của lệnh (>)**

Cách này rất hữu ích nếu muốn lưu kết quả của một lệnh đã thực hiện.

Để gửi kết quả của một lệnh vào một file, dùng dấu ">" theo nghĩa chuyển hướng lối ra chuẩn. Ví dụ, đưa kết quả của lệnh **ls -l /bin** vào file **/home/thu/lenhls** bằng cách gõ:

```
# ls -l /bin > /home/thu/lenhls
```

Linux tự động tạo nếu file **lenhls** chưa có, trong trường hợp ngược lại, nội dung file cũ sẽ bị thế chỗ bởi kết quả của lệnh.

```
# ls -l /bin >/home/thu/lenhls
```

Nếu muốn bổ sung kết quả vào cuối file thay vì thay thế nội dung file, hãy sử dụng dấu ">>".

Ví dụ, lệnh

```
# ls -l /bin >> /home/thu/lenhls
```

đưa các dòng danh sách file trong thư mục **/bin** vào cuối nội dung của file **/home/thu/lenhls**.

*** Tạo file với lệnh *cat***

Lệnh **cat** tuy đơn giản nhưng rất hữu dụng trong Linux. Chúng ta có thể sử dụng lệnh này để lấy thông tin từ đầu vào (bàn phím...) rồi kết xuất ra file hoặc các nguồn khác (màn hình ...), hay để xem nội dung của một file ... Phần này trình bày tác dụng của lệnh **cat** đối với việc tạo file. Cú pháp lệnh:

```
cat > <file>
```

Theo ngầm định, lệnh này cho phép lấy thông tin đầu vào từ bàn phím rồi xuất ra màn hình. Soạn thảo nội dung của một file bằng lệnh cat tức là đã đổi hướng đầu ra của lệnh từ màn hình vào một file. Người dùng gõ nội dung của file ngay tại dấu nhắc màn hình và gõ

CTRL+d để kết thúc việc soạn thảo. Nhược điểm của cách tạo file này là nó không cho phép sửa lỗi, ví dụ nếu muốn sửa một lỗi chính tả trên một dòng, chỉ có cách là xóa đến vị trí của lỗi và gõ lại nội dung vừa bị xóa.

Ví dụ. tạo file **newfile** trong thư mục **/home/vd** bằng lệnh **cat.**

```
# cat > /home/vd/newfile
This is a example of cat command
#
```

Sau khi soạn thảo xong, gõ **Enter** và **CTRL+d** để trở về dấu nhắc lệnh, nếu không gõ **Enter** thì phải gõ **CTRL+d** hai lần. Có thể sử dụng luôn lệnh **cat** để xem nội dung của file vừa soạn thảo:

```
# cat /home/vd/newfile
This is a example of cat command
#
```

3.4.3 Các lệnh thao tác trên file

*** Sao chép file với lệnh *cp***

Lệnh **cp** có hai dạng như sau:

```
cp [tùy-chọn] <file-nguồn> ... <file-đích>
cp [tùy-chọn] --target-directory=<thư-mục> <file-nguồn>...
```

Lệnh này cho phép sao **file-nguồn** thành **file-đích** hoặc sao chép từ nhiều file-nguồn vào một thư mục đích (tham số <file-đích> hay <thư-mục>). Dạng thứ hai là một cách viết khác đổi thứ tự hai tham số vị trí.

Các tùy chọn:

- **-a, --archive** : giống như **-dpR** (tổ hợp ba tham số **-d**, **-p**, **-R**, như dưới đây).
- **-b, --backup[=CONTROL]** : tạo file lưu cho mỗi file đích nếu như nó đang tồn tại.
- **-d, --no-dereference** : duy trì các liên kết.
- **-f, --force** : ghi đè file đích đang tồn tại mà không nhắc nhở.
- **-i, --interactive** : có thông báo nhắc nhở trước khi ghi đè.
- **-l, --link** : chỉ tạo liên kết giữa file-đích từ file-nguồn mà không sao chép.
- **-p, --preserve** : duy trì các thuộc tính của file-nguồn sang file-đích.
- **-r** : cho phép sao chép một cách đệ quy file thông thường.
- **-R** : cho phép sao chép một cách đệ quy thư mục.
- **-s, --symbolic-link** : tạo liên kết tượng trưng thay cho việc sao chép các file.
- **-S, --suffix=<hậu-tố>** : bỏ qua các hậu tố thông thường (hoặc được chỉ ra).
- **-u, --update** : chỉ sao chép khi file nguồn mới hơn file đích hoặc khi file đích chưa có.
- **-v, --verbose** : đưa ra thông báo về quá trình sao chép.
- **--help** : hiển thị trang trợ giúp và thoát.

File đích được tạo ra có cùng kích thước và các quyền truy nhập như file nguồn, tuy nhiên file đích có thời gian tạo lập là thời điểm thực hiện lệnh nên các thuộc tính thời gian sẽ khác.

Ví dụ, lệnh

```
# cp /home/ftp/vd /home/test/vd1
```

Nếu ở vị trí đích, mô tả đầy đủ tên file đích thì nội dung file nguồn sẽ được sao chép sang file đích. Trong trường hợp chỉ đưa ra vị trí file đích được đặt trong thư mục nào thì tên của file nguồn sẽ là tên của file đích.

```
# cp /home/ftp/vd /home/test/
```

Trong ví dụ này, tên file đích sẽ là **vd** nghĩa là tạo một file mới **/home/test/vd**.
Nếu sử dụng lệnh này để sao một thư mục, sẽ có một thông báo được đưa ra cho biết nguồn là một thư mục và vì vậy không thể dùng lệnh **cp** để sao chép.

```
# cp . newdir
cp: .: omitting directory
```

Ví dụ về việc lệnh **cp** cho phép sao nhiều file cùng một lúc vào một thư mục.

```
# cp vd vd1 newdir
# pwd
/newdir
# ls -l
total 8
-rw-r--r-- 1 root ftp 15 Nov 14 11:00 vd
-rw-r--r-- 1 root ftp 12 Nov 14 11:00 vd1
```

☞ ***Lưu ý:***

- Đối với nhiều lệnh làm việc với file, khi gõ lệnh có thể sử dụng kí hiệu mô tả nhóm để xác định một nhóm file làm cho tăng hiệu lực của các lệnh đó. Ví dụ, lệnh:

  ```
  # cp * bak
  ```

  thực hiện việc sao chép mọi file có trong thư mục hiện thời sang thư mục con của nó có tên là **bak**.

Dùng lệnh

```
# cp /usr/src/linux-2.2.14/include/linux/*.h bak
```

cho phép sao chép mọi file với tên có hai kí hiệu cuối cùng là "**.h**" sang thư mục con **bak**.

Chính vì lí do nói trên, dù trong nhiều lệnh tuy không nói đến việc sử dụng kí hiệu mô tả nhóm file nhưng chúng ta có thể áp dụng chúng nếu điều đó không trái với suy luận thông thường. Do những tình huống như thế là quá phong phú cho nên không thể giới thiệu hết trong tài liệu. Chúng ta chú ý một giải pháp là mỗi khi sử dụng một lệnh nào đó, nên thử nghiệm cách thức hiệu quả này.

*** <u>Đổi tên file với lệnh *mv*</u>**

Cú pháp lệnh đổi tên file:

```
mv <tên-cũ> <tên-mới>
```

Lệnh này cho phép đổi tên file từ **tên cũ** thành **tên mới**.

Ví dụ:

```
# mv vd newfile
```

Lệnh này sẽ đổi tên file **vd** thành **newfile**. Trong trường hợp file **newfile** đã tồn tại, nội dung của file **vd** sẽ ghi đè lên nội dung của file **newfile**.

*** Xóa file với lệnh *rm***

Lệnh **rm** là lệnh rất "nguy hiểm" vì trong Linux không có lệnh khôi phục lại những gì đã xóa, vì thế hãy cẩn trọng khi sử dụng lệnh này. Cú pháp lệnh:

```
rm [tùy-chọn] <file> ...
```

Lệnh **rm** cho phép xóa bỏ một file hoặc nhiều file.

Các tùy chọn:

- **-d, --directory** : loại bỏ liên kết của thư mục, kể cả thư mục không rỗng. Chỉ có siêu người dùng mới được phép dùng tùy chọn này.
- **-f, --force** : bỏ qua các file (xác định qua tham số **file**) không tồn tại mà không cần nhắc nhở.
- **-i, --interactive** : nhắc nhở trước khi xóa bỏ một file.
- **-r, -R, --recursive** : xóa bỏ nội dung của thư mục một cách đệ quy.
- **-v, --verbose** : đưa ra các thông báo về quá trình xóa file.
- **--help** : hiển thị trang trợ giúp và thoát.

Lệnh **rm** cho phép xóa nhiều file cùng một lúc bằng cách chỉ ra tên của các file cần xóa trong dòng lệnh (hoặc dùng kí kiệu mô tả nhóm).

Ví dụ, dùng lệnh **ls** để xem danh sách các file trong thư mục hiện thời:

```
# ls
ld-Linux.so.1               libnss_dns-2.1.3.so
ld-Linux.so.1.9.5           libnss_dns.so.1
ld-Linux.so.2               libnss_dns.so.2
ld.so                       libnss_files-2.1.3.so
ld.so.1.9.5                 libnss_files.so.1
libBrokenLocale-2.1.3.so    libnss_files.so.2
libBrokenLocale.so.1        libnss_hesiod-1.3.so
libNoVersion-2.1.3.so       telex.o
vd2.txt
```

Sử dụng lệnh xóa file **vd2.txt** sau đây:

```
# rm vd2.txt telex.o
```

và sau đó dùng lệnh **ls** để xem lại danh sách file:

```
# ls
ld-Linux.so.1               Libnss_dns-2.1.3.so
ld-Linux.so.1.9.5           Libnss_dns.so.1
ld-Linux.so.2               Libnss_dns.so.2
ld.so                       Libnss_files-2.1.3.so
```

```
ld.so.1.9.5                 Libnss_files.so.1
libBrokenLocale-2.1.3.so    Libnss_files.so.2
libBrokenLocale.so.1        Libnss_hesiod-1.3.so
libNoVersion-2.1.3.so       telex.o
```

Dùng lệnh

```
# rm bak/*.h
```

xóa mọi file với tên có hai kí hiệu cuối cùng là "**.h**" trong thư mục con **bak.**

*** Lệnh đếm từ và dòng trong file *wc***

Linux có lệnh **wc** dùng để đếm số ký tự, số từ, hay số dòng trong một file. Cú pháp lệnh:

```
wc [tùy-chọn] [file]...
```

Lệnh hiện ra số lượng dòng, số lượng từ, số lượng ký tự có trong mỗi file, và một dòng tính tổng nếu có nhiều hơn một file được chỉ ra. Nếu không có tùy chọn nào thì mặc định đưa ra cả số dòng, số từ và số ký tự. Ngầm định khi không có tên file trong lệnh thì sẽ đọc và đếm trên thiết bị vào chuẩn.

Các tuỳ chọn:

- **-c, --byte, --chars** : đưa ra số ký tự trong file.
- **-l, --lines** : đưa ra số dòng trong file.
- **-L, --max-line-length** : đưa ra chiều dài của dòng dài nhất trong file.
- **-w, --words** : đưa ra số từ trong file.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ, sau khi gõ lệnh:

```
# wc /home/lan/mau/mau1
```

xuất hiện dòng thông báo:

```
11 64 293 /home/lan/mau/mau1
```

Dòng thông báo trên cho biết file **mau1** có 293 ký tự, số 64 từ và có 11 dòng.

Ví dụ sau khi gõ lệnh:

```
# wc
```

người dùng gõ tiếp các dòng như sau:

```
This is a example of wc command without
[namefile]
```

sau đó người dùng gõ cặp phím **Ctrl-d** để kết thúc thì thấy dòng thông báo hiện ra:

```
2       9       49
```

Khi gõ lệnh **wc** mà không có một tham số nào, mặc định sẽ soạn thảo trực tiếp nội dung trên thiết bị vào chuẩn. Dùng **CTRL+d** để kết thúc việc soạn thảo, kết quả sẽ hiển thị lên màn hình như ví dụ trên.

```
# wc /home/lan/vd/vdcalj /home/lan/vd/vdwc
8       41      192     /home/lan/vd/vdcalj
24      209     1473    /home/lan/vd/vdwc
```

```
32      250    1665   total
```

Lệnh trên đếm số ký tự, số từ, số dòng trên mỗi file được chỉ ra, và dòng cuối cùng hiển thị tổng số dòng, số từ, số ký tự đếm được.

Bằng cách kết hợp lệnh **wc** với một số lệnh khác, có thể có nhiều cách để biết được những thông tin cần thiết. Chẳng hạn:

- kết hợp với lệnh **ls** để xác định số file có trong một thư mục:

```
# ls | wc -l
37
```

dòng lệnh trên cho biết trong thư mục chủ của có 36 file (do dòng đầu tiên kết quả thông báo của lệnh **ls** không xác định một file).

- kết hợp với lệnh **cat** để biết số tài khoản cá nhân có trên máy của người dùng:

```
# cat /etc/passwd | wc -l
 324
```

*** Lệnh loại bỏ những dòng không quan trọng *uniq***

Trong một số trường hợp khi xem nội dung một file, chúng ta thấy có một số các thông tin bị trùng lặp, ví dụ các dòng trống hoặc các dòng chứa nội dung giống nhau. Để đồng thời làm gọn và thu nhỏ kích thước của file, có thể sử dụng lệnh **uniq** để liệt kê ra nội dung file sau khi đã loại bỏ các dòng trùng lặp. Cú pháp lệnh:

```
uniq [tùy-chọn] [input] [output]
```

Lệnh **uniq** sẽ loại bỏ các dòng trùng lặp kề nhau từ **input** (thiết bị vào chuẩn) và chỉ giữ lại một dòng duy nhất trong số các dòng trùng lặp rồi đưa ra **output** (thiết bị ra chuẩn). Các tuỳ chọn:

- **-c, --count** : đếm và hiển thị số lần xuất hiện của các dòng trong file.
- **-d** : hiển thị lên màn hình dòng bị trùng lặp.
- **-u** : hiển thị nội dung file sau khi xóa bỏ toàn bộ các dòng bị trùng lặp không giữ lại một dòng nào.
- **-i** : hiển thị nội dung file sau khi xóa bỏ các dòng trùng lặp và chỉ giữ lại duy nhất một dòng có nội dung bị trùng lặp.
- **-D** : hiển thị tất cả các dòng trùng lặp trên màn hình.

Nếu sử dụng lệnh **uniq** trên một file không có các dòng trùng lặp thì lệnh không có tác dụng.

Ví dụ, người dùng sử dụng lệnh **cat** để xem nội dung file **vduniq**

```
# cat vduniq
Gnome có hai phương pháp để thoát ra ngoài.
Gnome có hai phương pháp để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

```
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
trước hết phải thêm nút này vào Panel.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Trong file **vduniq** có hai dòng bị trùng lặp và kề nhau là dòng thứ 1 và 2.

```
Gnome có hai phương pháp để thoát ra ngoài.
Gnome có hai phương pháp để thoát ra ngoài.
```

và dòng thứ 5 và 6

```
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Dùng lệnh **uniq** để loại bỏ dòng trùng lặp:

```
# uniq vduniq
Gnome có hai phương pháp để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
trước hết phải thêm nút này vào Panel.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Dòng cuối cùng trong file **vduniq** có nội dung trùng với dòng thứ 5, nhưng sau lệnh **uniq**, nó không bị xóa vì không kề với dòng có nội dung trùng lặp.

*** Sắp xếp nội dung file với lệnh *sort***

sort là lệnh đọc các thông tin và sắp xếp chúng theo thứ tự trong bảng chữ cái hoặc theo thứ tự được quy định theo các tùy chọn của lệnh. Cú pháp lệnh:

```
sort [tùy-chọn] [file] ...
```

Hiển thị nội dung sau khi sắp xếp của một hoặc nhiều file ra thiết bị ra chuẩn là tác dụng của lệnh **sort**. Ngầm định sắp xếp theo thứ tự từ điển của các dòng có trong các file (từng chữ cái theo bảng chữ hệ thống (chẳng hạn ASCII) và kể từ vị trí đầu tiên trong các dòng).

Các tùy chọn:

- **+<số1> [-<số2>]** : Hai giá trị **số1** và **số2** xác định "khóa" sắp xếp của các dòng, thực chất lấy xâu con từ vị trí **số1** tới vị trí **số2** của các dòng để so sánh lấy thứ tự sắp xếp các dòng. Nếu **số2** không có thì coi là hết các dòng; nếu **số2** nhỏ hơn **số1** thì bỏ qua lựa chọn này. Chú ý, nếu có **số2** thì phải cách **số1** ít nhất một dấu cách.
- **-b** : bỏ qua các dấu cách đứng trước trong phạm vi sắp xếp.
- **-c** : kiểm tra nếu file đã sắp xếp thì thôi không sắp xếp nữa.
- **-d** : xem như chỉ có các ký tự [a-zA-Z0-9] trong khóa sắp xếp, các dòng có các kí tự đặc biệt (dấu cách, ? ...) được đưa lên đầu.
- **-f** : sắp xếp không phân biệt chữ hoa chữ thường.
- **-n** : sắp xếp theo kích thước của file.
- **-r** : chuyển đổi thứ tự sắp xếp hiện thời.

Ví dụ, muốn sắp xếp file **vdsort**

```
# cat vdsort
trước hết phải thêm nút này vào Panel.
21434
bạn xác nhận là có thực sự muốn thoát hay không.
menu chính, chọn mục Logout ở đáy menu.
Bạn có thể sử dụng mục Logout từ menu chính
Gnome có hai phương pháp để thoát ra ngoài.
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
57879
Lựa chọn YES hoặc NO để kết thúc phiên làm việc với Gnome.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
# sort -f vdsort
21434
57879
Bạn có thể sử dụng mục Logout từ menu chính
bạn xác nhận là có thực sự muốn thoát hay không.
Gnome có hai phương pháp để thoát ra ngoài.
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
Lựa chọn YES hoặc NO để kết thúc phiên làm việc với Gnome.
menu chính, chọn mục Logout ở đáy menu.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức năng này.
trước hết phải thêm nút này vào Panel.
```

Có thể kết hợp lệnh **sort** với các lệnh khác, ví dụ:

```
# ls -s | sort -n
total 127
1       Archive/
1       infoWorld/
```

13 keylime.pie
46 drop.text.hqx
64 bitnet.mailing-lists.Z

Lệnh trên cho thứ tự sắp xếp của các file theo kích thước trong thư mục hiện thời.

3.4.4 Các lệnh thao tác theo nội dung file

*** Sử dụng lệnh *file* để xác định kiểu file**

Cú pháp lệnh **file**:

```
file [tùy-chọn] [-f file] [-m <file-ảnh>...] <file>...
```

Lệnh **file** cho phép xác định và in ra kiểu thông tin chứa trong **file**. Lệnh **file** sẽ lần lượt kiểm tra từ kiểu file hệ thống, kiểu file **magic** (ví dụ file mô tả thiết bị) rồi đến kiểu file văn bản thông thường. Nếu file được kiểm tra thỏa mãn một trong ba kiểu file trên thì kiểu file sẽ được in ra theo các dạng cơ bản sau:

- **text**: dạng file văn bản thông thường, chỉ chứa các mã ký tự ASCII.
- **executable**: dạng file nhị phân khả thi.
- **data**: thường là dạng file chứa mã nhị phân và không thể in ra được.

Một số tuỳ chọn sau đây:

- **-b** : cho phép chỉ đưa ra kiểu file mà không đưa kèm theo tên file.
- **-f tên-file** : cho phép hiển thị kiểu của các file có tên trùng với nội dung trên mỗi dòng trong file **tên-file**. Để kiểm tra trên thiết bị vào chuẩn, sử dụng dấu "-".
- **-z** : xem kiểu của file nén.

Ví dụ:

```
# file file.c file /dev/hda
file.c: C program text
file:   ELF   32-bit   LSB   executable,   Intel   80386,version
1,dynamically linked, not stripped
/dev/hda:   block special
```

Lệnh trên cho xem kiểu của hai file **file.c**, **file** và thư mục **/dev/hda**.
Nhớ rằng kết quả của lệnh **file** không phải lúc nào cũng chính xác tuyệt đối.

*** Xem nội dung file với lệnh *cat***

Ở đoạn trước, chúng ta đã có dịp làm quen với lệnh **cat** thông qua tác dụng tạo file của lệnh. Phần này giới thiệu tác dụng chủ yếu của lệnh **cat**: đó là tác dụng xem nội dung của một file. Cú pháp lệnh:

```
cat [tùy-chọn] <tên file>
```

Các tùy chọn:

- **-A, --show-all** : giống như tùy chọn **-vET**.
- **-b, --number-nonblank** : hiển thị thêm số thứ tự trên mỗi dòng (bỏ qua dòng trống).
- **-e** : giống như tùy chọn **-vE**.
- **-E, --show-ends** : hiển thị dấu "$" tại cuối mỗi dòng.

- **-n, --number** : hiển thị số thứ tự của mỗi dòng (kể cả dòng trống).
- **-s** : nếu trong nội dung file có nhiều dòng trống thì sẽ loại bỏ bớt để chỉ hiển thị một dòng trống.
- **-t** : giống như **-vT**.
- **-T, --show-tabs** : hiển thị dấu TAB dưới dạng ^I.
- **-v, --show-nonprinting** : hiển thị các ký tự không in ra được ngoại trừ LFD và TAB.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# cat vdcat
```

chúng ta thấy xuất hiện các dòng sau đây:

```
Gnome có hai phương pháp để thoát ra ngoài.
có thể sử dụng mục Logout từ menu chính
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.

Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
xác nhận là có thực sự muốn thoát hay không.

hoặc nút Logout trên Panel chính để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
# cat -bEs vdcat
```

thì nội dung file hiện ra như sau:

```
1 Gnome có hai phương pháp để thoát ra ngoài. $
2 có thể sử dụng mục Logout từ menu chính $
3 hoặc nút Logout trên Panel chính để thoát ra ngoài.$
4 Để thoát bằng cách sử dụng menu chính, hãy mở $
5 menu chính, chọn mục Logout ở đáy menu. $
$
6 Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu $
7 xác nhận là có thực sự muốn thoát hay không.$
$
8 hoặc nút Logout trên Panel chính để thoát ra ngoài.$
9 Để thoát bằng cách sử dụng menu chính, hãy mở $
10 menu chính, chọn mục Logout ở đáy menu.
```

*** Xem nội dung các file lớn với lệnh *more***

Lệnh **cat** cho phép xem nội dung của một file, nhưng nếu file quá lớn, nội dung file sẽ trôi trên màn hình và chỉ có thể nhìn thấy phần cuối của file. Linux có một lệnh cho phép có thể xem nội dung của một file lớn, đó là lệnh **more**. Cú pháp lệnh:

more [-dlfpcsu] [-số] [+/xâumẫu] [+dòng-số] [file ...]

Lệnh **more** hiển thị nội dung của file theo từng trang màn hình.

Các lựa chọn:

- **-số :** xác định số dòng nội dung của file được hiển thị (**số**).
- **-d** : trên màn hình sẽ hiển thị các thông báo giúp người dùng cách sử dụng đối với lệnh more, ví như [**Press space to continue**, **"q" to quit .]**, hay hiển thị [**Press "h" for instructions .]** thay thế cho tiếng chuông cảnh báo khi bấm sai một phím.
- **-l** : **more** thường xem ^L là một ký tự đặc biệt, nếu không có tùy chọn này, lệnh sẽ dừng tại dòng đầu tiên có chứa ^L và hiển thị % nội dung đã xem được (^L không bị mất), nhấn phím **space** (hoặc **enter**) để tiếp tục. Nếu có tùy chọn **-l**, nội dung của file sẽ được hiển thị như bình thường nhưng ở một khuôn dạng khác, tức là dấu ^L sẽ mất và trước dòng có chứa ^L sẽ có thêm một dòng trống.
- **-p** : không cuộn màn hình, thay vào đó là xóa những gì có trên màn hình và hiển thị tiếp nội dung file.
- **-c** : không cuộn màn hình, thay vào đó xóa màn hình và hiển thị nội dung file bắt đầu từ đỉnh màn hình.
- **-s** : xóa bớt các dòng trống liền nhau trong nội dung file chỉ giữ lại một dòng.
- **-u** : bỏ qua dấu gạch chân.
- **+/xâumẫu** : tùy chọn **+/xâumẫu** chỉ ra một chuỗi sẽ được tìm kiếm trước khi hiển thị mỗi file.
- **+dòng-số** : bắt đầu hiển thị từ dòng thứ **dòng-số.**

Ví dụ:

```
# more -d vdmore
total 1424
drwxr-xr-x 6 root root 4096 Oct 31 2000 AfterStep-1.8.0
drwxr-xr-x 2 root root 4096 Oct 31 2000 AnotherLevel
drwxr-xr-x 2 root root 4096 Oct 31 2000 ElectricFence
drwxr-xr-x 2 root root 4096 Oct 31 2000 GXedit-1.23
drwxr-xr-x 3 root root 4096 Oct 31 2000 HTML
drwxr-xr-x 3 root root 4096 Oct 31 2000 ImageMagick
drwxr-xr-x 6 root root 4096 Oct 31 2000 LDP
drwxr-xr-x 3 root root 4096 Oct 31 2000 ORBit-0.5.0
drwxr-xr-x 2 root root 4096 Oct 31 2000 SVGATextMode
drwxr-xr-x 2 root root 4096 Oct 31 2000 SysVinit-2.78
drwxr-xr-x 2 root root 4096 Oct 31 2000 WindowMaker
--More--(9%) [ Press space to continue, "q" to quit .]
```

Đối với lệnh **more**, có thể sử dụng một số các phím tắt để thực hiện một số các thao tác đơn giản trong khi đang thực hiện lệnh. Bảng dưới đây liệt kê các phím tắt đó:

Phím tắt	Chức năng
[Space]	Nhấn phím space để hiển thị màn hình tiếp theo

n	Hiển thị n dòng tiếp theo
[Enter]	Hiển thị dòng tiếp theo
h	Hiển thị danh sách các phím tắt
d hoặc CTRL+D	Cuộn màn hình (mặc định là 11 dòng)
q hoặc CTRL+Q	Thoát khỏi lệnh **more**
s	Bỏ qua n dòng (mặc định là 1)
f	Bỏ qua k màn hình tiếp theo (mặc định là 1)
b hoặc CTRL+B	Trở lại k màn hình trước (mặc định là 1)
=	Hiển thị số dòng hiện thời
:n	xem k file tiếp theo
:p	Trở lại k file trước
v	Chạy chương trình soạn thảo vi tại dòng hiện thời
CTRL+L	Vẽ lại màn hình
:f	Hiển thị tên file hiện thời và số dòng
.	Lặp lại lệnh trước

*** Thêm số thứ tự của các dòng trong file với lệnh *nl***

Như đã biết lệnh **cat** với tham số **-n** sẽ đánh số thứ tự của các dòng trong file, tuy nhiên Linux còn cho phép dùng lệnh **nl** để thực hiện công việc như vậy. Cú pháp lệnh:

```
nl [tùy-chọn] <file>
```

Lệnh này sẽ đưa nội dung file ra thiết bị ra chuẩn, với số thứ tự của dòng được thêm vào. Nếu không có **file** (**tên file)**, hoặc khi **file** là dấu "**-**", thì đọc nội dung từ thiết bị vào chuẩn.

Các tuỳ chọn:

- **-b, --body-numbering=STYLE** : sử dụng kiểu STYLE cho việc đánh thứ tự các dòng trong nội dung file. Có các kiểu STYLE sau:
  - **a** : đánh số tất cả các dòng kể cả dòng trống;
  - **t** : chỉ đánh số các dòng không trống;
  - **n** : không đánh số dòng.
- **-d, --section-delimiter=CC** : sử dụng CC để đánh số trang logic (CC là hai ký tự xác định phạm vi cho việc phân trang logic).
- **-f, --footer-numbering=STYLE** : sử dụng kiểu STYLE để đánh số các dòng trong nội dung file (một câu có thể có hai dòng ...).
- **-h, --header-numbering=STYLE** : sử dụng kiểu STYLE để đánh số các dòng trong nội dung file.
- **-i, --page-increment=số** : đánh số thứ tự của dòng theo cấp số cộng có công sai là **số**.
- **-l, --join-blank-lines=số** :nhóm **số** dòng trống vào thành một dòng trống.
- **-n, --number-format=khuôn** : chèn số dòng theo **khuôn** (khuôn: **ln** - căn trái, không có số 0 ở đầu; **rn** - căn phải, không có số 0 ở đầu; **rz** - căn phải và có số 0 ở đầu)
- **-p, --no-renumber** : không thiết lập lại số dòng tại mỗi trang logic.
- **-s, --number-separator=xâu** : thêm chuỗi **xâu** vào sau số thứ tự của dòng.
- **-v, --first-page=số** : số dòng đầu tiên trên mỗi trang logic.

- **-w, --number-width=số** : hiển thị số thứ tự của dòng trên cột thứ **số**.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
  # nl --body-numbering=a --number-format=rz vdnl
000001 1) New configuration mode
000002
000003
000004 1-1) Directories
000005
000006 Now, everything goes to ~/GNUstep/Library/AfterStep or
000007 /usr/local/share/afterstep !
000008
000009 You can use your old .steprc config file with afterstep -
f myoldsteprc,
000010 however, this isn't recommended at all.
000011
000012 New versions of asapps will also put their config. file
here in a near
000013 future, like modules currently do.
000014
```

Lệnh trong ví dụ trên cho thêm số thứ tự của các câu trong file **vdnl** theo dạng: đánh số thứ tự tất cả các dòng, kể cả dòng trống, các số thứ tự được căn phải và có số 0 ở đầu (lưu ý rằng có dòng trong file được hiện ra thành hai dòng trên giấy).

* Xem qua nội dung file với lệnh **head**

Các đoạn trước cho biết cách thức xem nội dung của một file nhờ lệnh **cat** hay **more**. Trong Linux cũng có các lệnh khác cho nhiều cách thức để xem nội dung của một file. Trước hết, chúng ta hãy làm quen với lệnh **head**. Cú pháp lệnh

```
head [tùy-chọn] [file]...
```

Lệnh này mặc định sẽ đưa ra màn hình 10 dòng đầu tiên của mỗi file. Nếu có nhiều hơn một file, thì lần lượt tên của file và 10 dòng nội dung đầu tiên sẽ được hiển thị. Nếu không có tham số **file**, hoặc **file** là dấu "**-**", thì ngầm định sẽ đọc từ thiết bị vào chuẩn.

Các tuỳ chọn:

- **-c, --bytes=cỡ** : hiển thị **cỡ** (số nguyên) ký tự đầu tiên trong nội dung file (**cỡ** có thể nhận giá trị là **b** cho 512, **k** cho 1K, **m** cho 1 Meg)
- **-n, --lines=n** : hiển thị **n** (số nguyên) dòng thay cho 10 dòng ngầm định.
- **-q, --quiet, --silent** : không đưa ra tên file ở dòng đầu.
- **-v, --verbose** : luôn đưa ra tên file ở dòng đầu.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
  # head -6 vdhead1 vdhead2
```

```
==> vdhead1 <==
1) New configuration mode
1-1) Directories

Now, everything goes to Ú/GNUstep/Library/AfterStep or

==> vdhead2 <==

1.7.164 patch 3

$HOME/GNUstep/Library/AfterStep/start/Desktop/Theme/.include
changed from shell script call to perl script call
```

Lệnh này cho xem qua 6 dòng đầu tiên trong nội dung hai file **vdhead1** và **vdhead2**.

*** Xem qua nội dung file với lệnh *tail***

Lệnh thứ hai cho phép xem qua nội dung của file là lệnh **tail** với cú pháp:

```
tail [tùy-chọn] [file]...
```

Lệnh **tail** ngầm định đưa ra màn hình 10 dòng cuối trong nội dung của các file. Nếu có nhiều hơn một file, thì lần lượt tên của file và 10 dòng cuối sẽ được hiển thị. Nếu không có tham số **file,** hoặc **file** là dấu "-" thì ngầm định sẽ đọc từ thiết bị vào chuẩn.

Các tùy chọn:

- **--retry** : cố gắng mở một file khó truy nhập khi bắt đầu thực hiện lệnh **tail.**
- **-c, --bytes=n** : hiển thị **n** (số) ký tự sau cùng.
- **-f, --follow[={name | descritptor}]** : sau khi hiện nội dung file sẽ hiện thông tin về file: **-f**, **--follow**, và **--follow=descriptor** là như nhau.
- **-n, --lines=n** : hiển thị **n** (số) dòng cuối cùng của file thay cho 10 dòng ngầm định.
- **--max-unchanged-stats=n** : hiển thị tài liệu về file (ngầm định **n** là 5).
- **--max-consecutive-size-changes=n** : hiển thị tài liệu về file (ngầm định **n** là 200).
- **--pid=PID** : kết hợp với tùy chọn **-f**, chấm dứt sau khi quá trình có chỉ số = **PID** lỗi.
- **-q, --quiet, --silent** : không đưa ra tên file ở dòng đầu trong nội dung được hiển thị.
- **-s, --sleep-interval=k** : kết hợp với tùy chọn -f, dừng **k** giây giữa các hoạt động.
- **-v, --verbose** : luôn hiển thị tên của file.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# tail -2 vdtail1 vdtail2

==> vdtail1 <==
Now, everything goes to ~/GNUstep/Library/AfterStep or
/usr/local/share/afterstep !

==> vdtail2 <==
```

```
changed from shell script call to perl script call
```

Lệnh trên cho xem hai dòng cuối của hai file **vdtail1** và **vdtail2**.

*** Tìm sự khác nhau giữa hai file (lệnh *diff*)**

Việc tìm ra sự khác nhau giữa hai file đôi khi là rất cần thiết. Linux có một lệnh có tác dụng như vậy, đó là lệnh **diff** với cú pháp:

```
diff [tuỳ-chọn] <file1> <file2>
```

Trong trường hợp đơn giản, lệnh **diff** sẽ so sánh nội dung của hai file. Nếu **file1** là một thư mục còn **file2** là một file bình thường, **diff** sẽ so sánh file có tên trùng với **file2** trong thư mục **file1** với **file2**.

Nếu cả **file1** và **file2** đều là thư mục, **diff** sẽ thực hiện sự so sánh lần lượt các file trong cả hai thư mục theo thứ tự từ a-z (sự so sánh này sẽ không đệ qui nếu tuỳ chọn **-r** hoặc **--recursive** không được đưa ra). Tất nhiên so sánh giữa hai thư mục không thể chính xác như khi so sánh hai file.

Các tuỳ chọn:

- **-a**: xem tất cả các file ở dạng văn bản và so sánh theo từng dòng.
- **-b**: bỏ qua sự thay đổi về số lượng của ký tự trống.
- **-B**: bỏ qua mọi sự thay đổi mà chỉ chèn hoặc xoá các dòng trống.
- **--brief**: chỉ thông báo khi có sự khác nhau mà không đưa ra chi tiết nội dung khác nhau.
- **-d**: tìm ra sự khác biệt nhỏ (tuỳ chọn này có thể làm chậm tốc độ làm việc của lệnh **diff**).
- **--exclude-from=file**: khi so sánh thư mục, bỏ qua các file và các thư mục con có tên phù hợp với mẫu có trong **file.**
- **-i**: so sánh không biệt chữ hoa chữ thường.
- **-r**: thực hiện so sánh đệ qui trên thư mục.
- **-s**: thông báo khi hai file là giống nhau.
- **-y**: hiển thị hai file cạnh nhau để dễ phân biệt sự khác nhau.

3.4.5 Các lệnh tìm file

*** Tìm theo nội dung file bằng lệnh *grep***

Lệnh **grep** cũng như lệnh **ls** là hai lệnh rất quan trọng trong Linux. Lệnh này có hai tác dụng cơ bản như sau:

- tác dụng thứ nhất là lọc đầu ra của một lệnh khác với cú pháp là

  ```
  <lệnh> | grep <mẫu lọc>
  ```

- tác dụng thứ hai, và cũng là tác dụng cơ bản được giới thiệu trong phần này, là tìm dòng chứa mẫu đã định trong file được chỉ ra.

Cú pháp lệnh **grep**:

```
grep [tùy-chọn] <mẫu-lọc> [file]
```

Lệnh **grep** hiển thị tất cả các dòng có chứa **mẫu-lọc** trong **file** được chỉ ra (hoặc từ thiết bị vào chuẩn nếu không có **file** hoặc **file** có dạng là dấu "-")

Các tùy chọn:

- **-G, --basic-regexp** : xem mẫu lọc như một biểu thức thông thường. Điều này là ngầm định.

- **-E, --extended-regexp** : xem mẫu lọc như một biểu thức mở rộng.
- **-F, --fixed-strings** : xem mẫu như một danh sách các xâu cố định, được phân ra bởi các dòng mới. Ngoài lệnh grep còn có hai lệnh là egrep và fgrep. egrep tương tự như lệnh grep -E, fgrep tương tự với lệnh grep -F .

Lệnh **grep** còn có các tùy chọn sau:

- **-A NUM, --after-context=NUM** : đưa ra NUM dòng nội dung tiếp theo sau dòng có chứa mẫu.
- **-B NUM, --before-context=NUM** : đưa ra NUM dòng nội dung trước dòng có chứa mẫu.
- **-C [NUM], --context[=NUM]** : hiển thị NUM dòng (mặc định là 2 dòng) nội dung.
- **-NUM** : giống **--context=NUM** đưa ra các dòng nội dung trước và sau dòng có chứa mẫu. Tuy nhiên, **grep** sẽ không đưa ra dòng nào nhiều hơn một lần.
- **-b, --byte-offset** : hiển thị địa chỉ tương đối trong file đầu vào trước mỗi dòng được đưa ra
- **-c, --count** : đếm số dòng tương ứng chứa mẫu trong file đầu vào thay cho việc hiển thị các dòng chứa mẫu.
- **-d ACTION, --directories=ACTION** : nếu đầu vào là một thư mục, sử dụng ACTION để xử lý nó. Mặc định, ACTION là **read**, tức là sẽ đọc nội dung thư mục như một file thông thường. Nếu ACTION là **skip**, thư mục sẽ bị bỏ qua. Nếu ACTION là **recurse**, **grep** sẽ đọc nội dung của tất cả các file bên trong thư mục (đệ quy); tùy chọn này tương đương với tùy chọn **-r**.
- **-f file, --file=file** : lấy các mẫu từ **file**, một mẫu trên một dòng. File trống chứa đựng các mẫu rỗng, và các dòng đưa ra cũng là các dòng trống.
- **-H, --with-file** : đưa ra tên file trên mỗi dòng chứa mẫu tương ứng.
- **-h, --no-filename** : không hiển thị tên file kèm theo dòng chứa mẫu trong trường hợp tìm nhiều file.
- **-i** : hiển thị các dòng chứa mẫu không phân biệt chữ hoa chữ thường.
- **-l** : đưa ra tên các file trùng với mẫu lọc.
- **-n, --line-number** : thêm số thứ tự của dòng chứa mẫu trong file.
- **-r, --recursive** : đọc tất cả các file có trong thư mục (đệ quy).
- **-s, --no-messages** : bỏ qua các thông báo lỗi file không đọc được hoặc không tồn tại.
- **-v, --invert-match** : hiển thị các dòng không chứa mẫu.
- **-w, --word-regexp** : chỉ hiển thị những dòng có chứa mẫu lọc là một từ trọn vẹn.
- **-x, --line-regexp** : chỉ hiển thị những dòng mà nội dung trùng hoàn toàn với mẫu lọc.

Ví dụ, người dùng gõ lệnh **cat** để xem nội dung file **text**:

```
# cat -n text
```

thì hiện ra nội dung file đó như sau:

```
1$ file file.c file /dev/hda
2file.c: C program text
3file:ELF 32-bit LSB executable, Intel 80386, version 1,
4dynamically linked, not stripped
5/dev/hda: block special
6
7$ file -s /dev/hdaỳ,1,2,3,4,5,6,7,8,9,10Ỳ
```

```
8/dev/hda: x86 boot sector
9/dev/hda1: Linux/i386 ext2 filesystem
10     /dev/hda2: x86 boot sector
11     /dev/hda3: x86 boot sector, extended partition table
12     /dev/hda4: Linux/i386 ext2 filesystem
13     /dev/hda5: Linux/i386 swap file
14     /dev/hda6: Linux/i386 swap file
15     /dev/hda7: Linux/i386 swap file
16     /dev/hda8: Linux/i386 swap file
17     thutest
18     toithutest
```

Sau đó, dùng lệnh **grep** để lọc các dòng có cụm **filesystem**

```
# grep -n filesystem text
9: /dev/hda1: Linux/i386 ext2 filesystem
12: /dev/hda4: Linux/i386 ext2 filesystem
```

Cũng có thể sử dụng các ký hiệu biểu diễn thông thường (regular - expression) trong mẫu lọc để đưa ra được nhiều cách tìm kiếm file khác nhau.

Bảng dưới đây liệt kê một số ký hiệu hay dùng:

Ký hiệu	Ý nghĩa
C	- thay thế cho ký tự c
\c	- hiển thị c như là một ký tự bình thường nếu c là một ký tự điều khiển
^	- bắt đầu một dòng
$	- kết thúc dòng
.	- thay cho một ký tự đơn
[xy]	- chọn một ký tự trong tập hợp các ký tự được đưa ra
[^xy]	- chọn một ký tự không thuộc tập hợp các ký tự được đưa ra
c*	- thay cho một mẫu có hoặc không chứa ký tự c

```
# grep - H thutest text
text: thutest
text: toithutest
# grep -H "^thutest" text
text: thutest
```

Ngoài các tùy chọn khác nhau, lệnh **grep** còn có hai dạng nữa trên Linux. Hai dạng đó là **egrep** - sử dụng với các mẫu lọc phức tạp, và **fgrep** - sử dụng để tìm nhiều mẫu lọc cùng một lúc.

- Thỉnh thoảng một biểu thức đơn giản không thể xác định được đối tượng cần tìm, ví dụ, như đang cần tìm các dòng có một hoặc hai mẫu lọc. Những lúc đó, lệnh **egrep** tỏ ra rất có ích. **egrep - expression grep -** có rất nhiều các ký hiệu biểu diễn mạnh hơn **grep**. Dưới đây là các ký hiệu hay dùng:

Ký hiệu	Ý nghĩa
c	- thay thế cho ký tự c
\c	- hiển thị c như là một ký tự bình thường nếu c là một ký tự điều khiển
^	- bắt đầu một dòng
$	- kết thúc dòng
.	- thay cho một ký tự đơn
[xy]	- chọn một ký tự trong tập hợp các ký tự được đưa ra
[^xy]	- chọn một ký tự không thuộc tập hợp các ký tự được đưa ra
c*	- thay cho một mẫu có hoặc không chứa ký tự c
c+	- thay cho một mẫu có chứa một hoặc nhiều hơn ký tự c
c?	- thay cho một mẫu không có hoặc chỉ có chứa duy nhất một ký tự c
a | b	- hoặc là a hoặc là b
(a)	- a một biểu thức

Ví dụ, giả sử bây giờ muốn tìm các dòng có chứa một hoặc nhiều hơn ký tự ***b*** trên file **passwk** với lệnh **egrep**.

```
# egrep 'b+' /etc/passwd | head
```

cho ra các dòng kết quả sau:

```
root : x : 0 : 0 : root : /root : /bin/bash
bin : x : 1 : 1 : bin : /bin :
daemon : x : 2 : 2 : daemon : /sbin :
sync : x : 5 : 0 : sync : /sbin : /bin/sync
shutdown : x : 6 : 0 : shutdown : /sbin : /sbin/shutdown
halt : x : 7 : 0 : halt : /sbin : /sbin/halt
gopher : x : 13 : 30 : gopher : /usr/lib/gopher-data :
nobody : x : 99 : 99 : Nobody : / :
xfs : x : 43 : 43 : X Font Server : /etc/X11/fs : /bin/false
named : x : 25 : 25 : Named : /var/named : /bin/false
```

Khi gõ lệnh:

```
# egrep '([^a-zA-Z] | :wi)' /etc/printcap | head
```

thì nhận được thông báo kết quả:

```
aglw:U\
      :wi=AG 23 : wk=multiple Apple LaserWrite IINT:
 aglw1:\
      :wi=AG 23 : wk=Apple LaserWrite IINT:
 aglw2:\
      :wi=AG 23 : wk=Apple LaserWrite IINT:
 aglw3:\
      :wi=AG 23 : wk=Apple LaserWrite IINT:
```

Lệnh trên cho phép tìm các dòng được bắt đầu bởi (^) một chữ cái không phân biệt chữ hoa chữ thường ([a-zA-Z]) hoặc (|) dòng có chứa mẫu **:wi**.

Bất kỳ lúc nào muốn tìm các dòng có chứa nhiều hơn một mẫu lọc, **egrep** là lệnh tốt nhất để sử dụng.

❖ Có những lúc cần phải tìm nhiều mẫu lọc trong một lúc. Ví dụ, có một file chứa rất nhiều mẫu lọc và muốn sử dụng một lệnh trong Linux để tìm các dòng có chứa các mẫu đó. Lệnh **fgrep** sẽ làm được điều này.

Ví dụ, file **thu** có nội dung như sau:

```
# cat thu
/dev/hda4: Linux/i386 ext2 filesystem
/dev/hda5: Linux/i386 swap file
/dev/hda8: Linux/i386 swap file
/dev/hda9: empty
/dev/hda10: empty
thutest
toithutest
```

và file **mauloc** có nội dung là:

```
# cat mauloc
empty
test
```

Bây giờ muốn sử dụng nội dung file **mauloc** làm mẫu lọc để tìm các câu trong file **thu**, hãy gõ lệnh:

```
# fgrep -i -f mauloc thu
/dev/hda9: empty
/dev/hda10: empty
thutest
toithutest
```

*** <u>Tìm theo các đặc tính của file với lệnh *find*</u>**

Các đoạn trên đây đã giới thiệu cách thức tìm file theo nội dung với các lệnh **grep**, **egrep** và **fgrep**. Linux còn cho phép người dùng sử dụng một cách thức khác đầy năng lực, đó là sử dụng lệnh **find**, lệnh tìm file theo các thuộc tính của file. Lệnh này có một sự khác biệt so với các lệnh khác, đó là các tùy chọn của lệnh là một từ chứ không phải một ký tự. Điều kiện cần đối với lệnh này là chỉ ra được điểm bắt đầu của việc tìm kiếm trong hệ thống file và những quy tắc cần tuân theo của việc tìm kiếm. Cú pháp của lệnh **find**:

```
find [đường-dẫn] [biểu-thức]
```

Lệnh **find** thực hiện việc tìm kiếm file trên cây thư mục theo **biểu thức** được đưa ra. Mặc định **đường dẫn** là thư mục hiện thời, **biểu thức** là **-print**.

Biểu thức có thể có những dạng sau:

- Các toán tử:

(EXPR); ! EXPR hoặc -not EXPR; EXPR1 -a EXPR2 hoặc EXPR1 -and EXPR2; EXPR1 -o EXPR2 hoặc EXPR1 -or EXPR2; và EXPR1, EXPR2

- Các tùy chọn lệnh: tất cả các tùy chọn này luôn trả về giá trị **true** và được đặt ở đầu biểu thức

- **-daystart** : đo thời gian (-amin, -atime, -cmin, -ctime, -mmin, -mtime).
- **-depth** : thực hiện tìm kiếm từ nội dung bên trong thư mục trước (mặc định việc tìm kiếm được thực hiện bắt đầu tại gốc cây thư mục có chứa file cần tìm).
- **-follow** : (tùy chọn này chỉ áp dụng cho thư mục) nếu có tùy chọn này thì các liên kết tượng trưng có trong một thư mục liên kết sẽ được chỉ ra.
- **-help**, **--help** : hiển thị kết quả của lệnh find và thoát.
  - các **test**
- **-amin n** : tìm file được truy nhập n phút trước.
- **-atime n** : tìm file được truy nhập n*24 giờ trước.
- **-cmin n** : trạng thái của file được thay đổi n phút trước đây.
- **-ctime n** : trạng thái của file được thay đổi n*24 giờ trước đây.
- **-empty** : file rỗng và hoặc là thư mục hoặc là file bình thường.
- **-fstype kiểu** : file thuộc hệ thống file với kiểu.
- **-gid n** : chỉ số nhóm của file là n.
- **-group nhóm** : file thuộc quyền sở hữu của nhóm.
- **-links n** : file có n liên kết.
- **-mmin n** : dữ liệu của file được sửa lần cuối vào n phút trước đây.
- **-mtime n** : dữ liệu của file được sửa vào n*24 giờ trước đây.
- **-name mẫu** : tìm kiếm file có tên là mẫu. Trong tên file có thể chứa cả các ký tự đại diện như dấu "*", "?"...
- **-type kiểu** : tìm các file thuộc kiểu với kiểu nhận các giá trị:
  - b: đặc biệt theo khối
  - c: đặc biệt theo ký tự
  - d: thư mục
  - p: pipe
  - f: file bình thường
  - l: liên kết tượng trưng
  - s: socket
- **-uid n**: chỉ số người sở hữu file là n.
- **-user tên-người**: file được sở hữu bởi người dùng tên-người.

- Các hành động
  - **-exec lệnh :** tùy chọn này cho phép kết hợp lệnh **find** với một lệnh khác để có được thông tin nhiều hơn về các thư mục có chứa file cần tìm. Tùy chọn **exec** phải sử dụng dấu {} - nó sẽ thay thế cho tên file tương ứng, và dấu '\' tại cuối dòng lệnh, (phải có khoảng trống giữa {} và '\'). Kết thúc lệnh là dấu '**;**'
  - **-fprint file :** hiển thị đầy đủ tên file vào trong **file**. Nếu **file** không tồn tại thì sẽ được tạo ra, nếu đã tồn tại thì sẽ bị thay thế nội dung.
  - **-print :** hiển thị đầy đủ tên file trên thiết bị ra chuẩn.

- **-ls :** hiển thị file hiện thời theo khuôn dạng: liệt kê danh sách đầy đủ kèm cả số thư mục, chỉ số của mỗi file, với kích thước file được tính theo khối (block).

Ví dụ:

```
# find -name 'what*'
./usr/bin/whatis
./usr/bin/whatnow
./usr/doc/AfterStep-1.8.0/TODO/1.0ỵtoỵ1.5/whatsnew
./usr/doc/gnome-libs-devel-1.0.55/devel-docs/gnome-dev-
info/gnome-dev-info/what.html
./usr/doc/gnome-libs-devel-1.0.55/devel-docs/gnome-dev-
info/gnome-dev-info/whatis.html

# find . -type f -exec grep -l -i mapping {} \ ;
./OWL/WordMap/msw-to-txt.c
./.elm/aliases.text
./Mail/mark
./News/usenet.alt
./bin/my.new.cmd: Permission denied
./src/fixit.c
./temp/attach.msg
```

3.5 Nén và sao lưu các file

3.5.1 Sao lưu các file (lệnh tar)

Dữ liệu rất có giá trị, sẽ mất nhiều thời gian và công sức nếu phải tạo lại, thậm chí có lúc cũng không thể nào tạo lại được. Vì vậy, Linux đưa ra các cách thức để người dùng bảo vệ dữ liệu của mình.

Có bốn nguyên nhân cơ bản khiến dữ liệu có thể bị mất: lỗi phần cứng, lỗi phần mềm, lỗi do con người hoặc do thiên tai.

Sao lưu là cách để bảo vệ dữ liệu một cách kinh tế nhất. Bằng cách sao lưu dữ liệu, sẽ không có vấn đề gì xảy ra nếu dữ liệu trên hệ thống bị mất.

Một vấn đề rất quan trọng trong việc sao lưu đó là lựa chọn phương tiện sao lưu. cần phải quan tâm đến giá cả, độ tin cậy, tốc độ, ích lợi cũng như tính khả dụng của các phương tiện sao lưu.

Có rất nhiều các công cụ có thể được sử dụng để sao lưu. Các công cụ truyền thống là **tar**, **cpio** và **dump** (công cụ trong tài liệu này là **tar**). Ngoài ra còn rất nhiều các công cụ khác có thể lựa chọn tùy theo phương tiện sao lưu có trong hệ thống.

Có hai kiểu sao lưu là sao lưu theo kiểu toàn bộ (***full backup***) và sao lưu theo kiểu tăng dần (***incremental backup***). Sao lưu toàn bộ thực hiện việc sao mọi thứ trên hệ thống file, bao gồm tất cả các file. Sao lưu tăng dần chỉ sao lưu những file được thay đổi hoặc được tạo ra kể từ đợt sao lưu cuối cùng.

Việc sao lưu toàn bộ có thể được thực hiện dễ dàng với lệnh **tar** với cú pháp:

```
tar [tùy-chọn] [<file>, ...] [<thư-mục>, ...]
```

Lệnh (chương trình) **tar** được thiết kế để tạo lập một file lưu trữ duy nhất. Với **tar**, có thể kết hợp nhiều file thành một file duy nhất có kích thước lớn hơn, điều này sẽ giúp cho việc di chuyển file hoặc sao lưu băng từ trở nên dễ dàng hơn nhiều.

Lệnh **tar** có các lựa chọn:

- **-c, --create** : tạo file lưu trữ mới.
- **-d, --diff, --compare** : tìm ra sự khác nhau giữa file lưu trữ và file hệ thống được lưu trữ.
- **--delete** : xóa từ file lưu trữ (không sử dụng cho băng từ).
- **-r, --append** : chèn thêm file vào cuối file lưu trữ.
- **-t, --list** : liệt kê nội dung của một file lưu trữ.
- **-u, --update** : chỉ thêm vào file lưu trữ các file mới hơn các file đã có.
- **-x, --extract, --get** : tách các file ra khỏi file lưu trữ.
- **-C, --directory tên-thư-mục** : thay đổi đến thư mục có tên là **tên-thư-mục.**
- **--checkpoint** : đưa ra tên thư mục khi đọc file lưu trữ.
- **-f, --file [HOSTNAME:]file** : tùy chọn này xác định tên file lưu trữ hoặc thiết bị lưu trữ là **file** (nếu không có tùy chọn này, mặc định nơi lưu trữ là **/dev/rmt0**).
- **-h, --dereference :** không hiện các file liên kết mà hiện các file mà chúng trỏ tới.
- **-k, --keep-old-files** : giữ nguyên các file lưu trữ đang tồn tại mà không ghi đè file lưu trữ mới lên chúng.
- **-K, --starting-file file** : bắt đầu tại **file** trong file lưu trữ.
- **-l, --one-file-system** : tạo file lưu trữ trên hệ thống file cục bộ.
- **-M, --multi-volume** : tùy chọn này được sử dụng khi dung lượng của file cần sao lưu là lớn và không chứa hết trong một đơn vị lưu trữ vật lý.
- **-N, --after-date DATE, --newer DATE** : chỉ lưu trữ các file mới hơn các file được lưu trữ trong ngày DATE.
- **--remove-files** : xóa file gốc sau khi đã sao lưu chúng vào trong file lưu trữ.
- **--totals** : đưa ra tổng số byte được tạo bởi tùy chọn **--create**.
- **-v, --verbose** : hiển thị danh sách các file đã được xử lý.

Ví dụ:

```
# tar --create --file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

Lệnh trên tạo một file sao lưu của thư mục **/usr/src** trong thư mục **/dev/ftape**, (dòng thông báo ở trên cho biết rằng **tar** sẽ chuyển cả dấu **/** vào trong file sao lưu).

Nếu việc sao lưu không thể thực hiện gọn vào trong một băng từ, lúc đó hãy sử dụng tùy chọn **-M**:

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

Chú ý rằng phải định dạng đĩa mềm trước khi thực hiện việc sao lưu, có thể sử dụng một thiết bị đầu cuối khác để thực hiện việc định dạng đĩa khi **tar** yêu cầu một đĩa mềm mới.

Sau khi thực hiện việc sao lưu, có thể kiểm tra kết quả của công việc bằng tùy chọn **--compare**:

```
# tar --compare --verbose -f /dev/ftape
usr/src/
usr/src/Linux
usr/src/Linux-1.2.10-includes/
...
#
```

Để sử dụng kiểu sao lưu tăng dần, hãy sử dụng tùy chọn **-N**:

```
# tar --create --newer '8 Sep 1995' --file /dev/ftape /usr/src
--verbose
tar: Removing leading / from absolute path names in the
archive
usr/src/
usr/src/Linux-1.2.10-includes/
usr/src/Linux-1.2.10-includes/include/
usr/src/Linux-1.2.10-includes/include/Linux/
usr/src/Linux-1.2.10-includes/include/Linux/modules/
usr/src/Linux-1.2.10-includes/include/asm-generic/
usr/src/Linux-1.2.10-includes/include/asm-i386/
usr/src/Linux-1.2.10-includes/include/asm-mips/
usr/src/Linux-1.2.10-includes/include/asm-alpha/
usr/src/Linux-1.2.10-includes/include/asm-m68k/
usr/src/Linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

Lưu ý rằng, **tar** không thể thông báo được khi các thông tin trong ***inode*** của một file bị thay đổi, ví dụ như thay đổi quyền truy nhập của file, hay thay đổi tên file chẳng hạn. Để biết được những thông tin thay đổi sẽ cần dùng đến lệnh **find** và so sánh với trạng thái hiện thời của file hệ thống với danh sách các file được sao lưu từ trước.

3.5.2 Nén dữ liệu

Việc sao lưu rất có ích nhưng đồng thời nó cũng chiếm rất nhiều không gian cần thiết để sao lưu. Để giảm không gian lưu trữ cần thiết, có thể thực hiện việc nén dữ liệu trước khi sao lưu, sau đó thực hiện việc giải nén (dãn) để nhận lại nội dung trước khi nén.

Trong Linux có khá nhiều cách để nén dữ liệu, tài liệu này giới thiệu hai phương cách phổ biến là **gzip** và **compress**.

*** Nén, giải nén và xem nội dung các file với lệnh *gzip*, *gunzip* và *zcat***

Cú pháp các lệnh này như sau:

```
gzip [tùy-chọn] [ -S suffix ] [ < file> ]
gunzip [tùy-chọn] [ -S suffix ] [ <file> ]
zcat [tùy-chọn] [ <file> ]
```

Lệnh **gzip** sẽ làm giảm kích thước của file và khi sử dụng lệnh này, file gốc sẽ bị thay thế bởi file nén với phần mở rộng là .gz, các thông tin khác liên quan đến file không thay đổi. Nếu không có tên file nào được chỉ ra thì thông tin từ thiết bị vào chuẩn sẽ được nén và gửi ra thiết bị ra chuẩn. Trong một vài trường hợp, lệnh này sẽ bỏ qua liên kết tượng trưng.

Nếu tên file nén quá dài so với tên file gốc, **gzip** sẽ cắt bỏ bớt. **gzip** sẽ chỉ cắt phần tên file vượt quá 3 ký tự (các phần được ngăn cách với nhau bởi dấu chấm). Nếu tên file gồm nhiều phần nhỏ thì phần dài nhất sẽ bị cắt bỏ. Ví dụ, tên file là **gzip.msdos.exe**, khi được nén sẽ có tên là **gzip.msd.exe.gz**.

File được nén có thể được khôi phục trở lại dạng nguyên thể với lệnh **gzip -d** hoặc **gunzip**.

Với lệnh **gzip** có thể giải nén một hoặc nhiều file có phần mở rộng là **.gz**, **-gz**, **.z**, **-z**, **_z** hoặc **.Z** ... **gunzip** dùng để giải nén các file nén bằng lệnh **gzip**, **zip**, **compress**, **compress -H**.

Lệnh **zcat** được sử dụng khi muốn xem nội dung một file nén trên thiết bị ra chuẩn.

Các tùy chọn:

- **-c, --stdout --to-stdout** : đưa ra trên thiết bị ra chuẩn; giữ nguyên file gốc không có sự thay đổi. Nếu có nhiều hơn một file đầu vào, đầu ra sẽ tuần tự là các file được nén một cách độc lập.
- **-d, --decompress --uncompress** : giải nén.
- **-f, --force** : thực hiện nén hoặc giải nén thậm chí file có nhiều liên kết hoặc file tương ứng thực sự đã tồn tại, hay dữ liệu nén được đọc hoặc ghi trên thiết bị đầu cuối.
- **-h, --help** : hiển thị màn hình trợ giúp và thoát.
- **-l, --list** : hiển thị những thông tin sau đối với một file được nén:
  - **compressed size:** kích thước của file nén
  - **uncompressed size**: kích thước của file được giải nén
  - **ratio**: tỷ lệ nén (0.0% nếu không biết)
  - **uncompressed_name**: tên của file được giải nén

Nếu kết hợp với tùy chọn **--verbose**, các thông tin sau sẽ được hiển thị:

- **method**: phương thức nén
- **crc**: CRC 32-bit cho dữ liệu được giải nén
- **date & time**: thời gian các file được giải nén

Nếu kết hợp với tùy chọn **--name**, tên file được giải nén, thời gian giải nén được lưu trữ trong file nén

Nếu kết hợp với tùy chọn **--verbose**, tổng kích thước và tỷ lệ nén của tất cả các file sẽ được hiển thị

Nếu kết hợp với tùy chọn **--quiet**, tiêu đề và tổng số dòng của các file nén không được hiển thị.

- **-n, --no-name** : khi nén, tùy chọn này sẽ không lưu trữ tên file gốc và thời gian nén, (tên file gốc sẽ luôn được lưu nếu khi nén tên của nó bị cắt bỏ). Khi giải nén, tùy chọn này sẽ không khôi phục lại tên file gốc cũng như thời gian thực hiện việc nén. Tùy chọn này được ngầm định.

- **-N, --name** : tùy chọn này ngược với tùy chọn trên (**-n**), nó hữu ích trên hệ thống có sự giới hạn về độ dài tên file hay khi thời điểm nén bị mất sau khi chuyển đổi file.
- **-q, --quiet** : bỏ qua mọi cảnh báo.
- **-r, --recursive** : nén thư mục.
- **-S .suf, --suffix .suf** : sử dụng phần mở rộng **.suf** thay cho **.gz**. Bất kỳ phần mở rộng nào cũng có thể được đưa ra, nhưng các phần mở rộng khác **.z** và **.gz** sẽ bị ngăn chặn để tránh sự lộn xộn khi các file được chuyển đến hệ thống khác.
- **-t, --test** : tùy chọn này được sử dụng để kiểm tra tính toàn vẹn của file được nén
- **-v, --verbose** : hiển thị phần trăm thu gọn đối với mỗi file được nén hoặc giải nén
- **-#, --fast, --best** : điều chỉnh tốc độ của việc nén bằng cách sử dụng dấu #,
    - ❖ nếu -# là **-1** hoặc **--fast** thì sử dụng phương thức nén nhanh nhất (less compression),
    - ❖ nếu là **-9** hoặc **--best** thì sẽ dùng phương thức nén chậm nhất (best compression).
    - ❖ Ngầm định mức nén là **-6** (đây là phương thức nén theo tốc độ nén cao).

Ví dụ:

```
# ls /home/test
Desktop data dictionary newt-0.50.8 rpm save vd1
# gzip /home/test/vd1
# ls /home/test
Desktop data dictionary newt-0.50.8 rpm save vd1.gz
# zcat /home/test/vd1
PID TTY TIME CMD
973 pts/0 00:00:00 bash
996 pts/0 00:00:00 man
1008 pts/0 00:00:00 sh
1010 pts/0 00:00:00 less
1142 pts/0 00:00:00 cat
1152 pts/0 00:00:00 cat
1181 pts/0 00:00:00 man
1183 pts/0 00:00:00 sh
1185 pts/0 00:00:00 less
#
```

*** Nén, giải nén và xem file với các lệnh *compress, uncompress, zcat***

Cú pháp các lệnh như sau:

```
compress [tùy-chọn] [<file>]
uncompress [tùy-chọn] [<file>]
zcat [tùy-chọn] [<file>]
```

Lệnh **compress** làm giảm kích thước của file và khi sử dụng lệnh này, file gốc sẽ bị thay thế bởi file nén với phần mở rộng là **.Z**, các thông tin khác liên quan đến file không

thay đổi. Nếu không có tên file nào được chỉ ra, thông tin từ thiết bị vào chuẩn sẽ được nén và gửi ra thiết bị ra chuẩn. Lệnh **compress** chỉ sử dụng cho các file thông thường. Trong một vài trường hợp, nó sẽ bỏ qua liên kết tượng trưng. Nếu một file có nhiều liên kết cứng, **compress** bỏ qua việc nén file đó trừ khi có tùy chọn **-f**.
Các tùy chọn:

- **-f** : nếu tùy chọn này không được đưa ra và **compress** chạy trong chế độ nền trước, người dùng sẽ được nhắc khi các file đã thực sự tồn tại và có thể bị ghi đè. Các file được nén có thể được khôi phục lại nhờ việc sử dụng lệnh **uncompress**.
- **-c** : tùy chọn này sẽ thực hiện việc nén hoặc giải nén rồi đưa ra thiết bị ra chuẩn, không có file nào bị thay đổi.

Lệnh **zcat** tương đương với **uncompress -c**. **zcat** thực hiện việc giải nén hoặc là các file được liệt kê trong dòng lệnh hoặc từ thiết bị vào chuẩn để đưa ra dữ liệu được giải nén trên thiết bị ra chuẩn.

- **-r** : nếu tùy chọn này được đưa ra, **compress** sẽ thực hiện việc nén các thư mục.
- **-v** : hiển thị tỷ lệ giảm kích thước cho mỗi file được nén.

CHƯƠNG 4. QUẢN TRỊ QUÁ TRÌNH

4.1 Quá trình trong UNIX

4.1.1. Sơ bộ về quá trình

Quá trình là đối tượng trong hệ thống tương ứng với một phiên thực hiện của một chương trình. Quá trình bao gồm ba thành phần là text, data, stack. Text là thành phần câu lệnh thực hiện, data là thành phần dữ liệu còn stack là thành phần thông tin tạm thời hoạt động theo cơ chế LIFO. Các câu lệnh trong text chỉ thao tác tới vùng data, stack tương ứng của quá trình, không truy nhập được tới data và stack của các quá trình khác, ngoại trừ các vùng dữ liệu dùng chung.

Các quá trình được hệ thống phân biệt bằng số hiệu của quá trình, viết tắt là PID (Proccess Index). Quá trình được tạo khi khởi động hệ điều hành là quá trình 0. Mọi quá trình khác đều được tạo ra từ một quá trình khác thông qua lời gọi hệ thống ***fork***: quá trình thực hiện lời gọi hệ thống ***fork*** được gọi là quá trình cha, còn quá trình được tạo ra theo lời gọi ***fork*** được gọi là quá trình con. Trừ quá trình 0 không có cha, mọi quá trình có trong hệ thống đều có một cha và một cha có thể có nhiều con.

Kết quả dịch chương trình nguồn sẽ tạo ra file chương trình đích gồm một số phần như sau (lưu trữ trên vật dẫn ngoài):

- Phần đầu file mô tả một số đặc tính của file chương trình (tương tự File header của file chương trình trong MS-DOS),
- Phần text của chương trình,
- Các giá trị mở đầu về việc phân phối bộ nhớ đối với vùng data của chương trình,
- Một số bảng thông tin liên quan đến đặt file.
- Khi có lời gọi ***fork***, thông qua lời gọi hệ thống ***exec***, nhân sẽ tải nội dung của file chương trình vào bộ nhớ trong theo các vùng text, data và stack:
- Vùng text của quá trình tương ứng với file chương trình,
- Vùng data của quá trình tương ứng với các giá trị được quy định trong file chương trình,
- Vùng stack được nhân tự động tạo với kích thước theo sự linh hoạt của nhân.

Phần stach bao gồm các stack frame (khung) lôgic: mỗi satck frame được đặt vào khi ***gọi một hàm*** và lấy ra khi quay về. Mỗi stack frame chứa tham sô của hàm, các biến địa phương v.v. Tương ứng trong stack có một stack pointer liên quan đến chiều sâu của stack. Trong mã chương trình có các dòng lệnh quản lý hình trạng của stack, và nhân sẽ định vị không gian đối với stack theo yêu cầu.

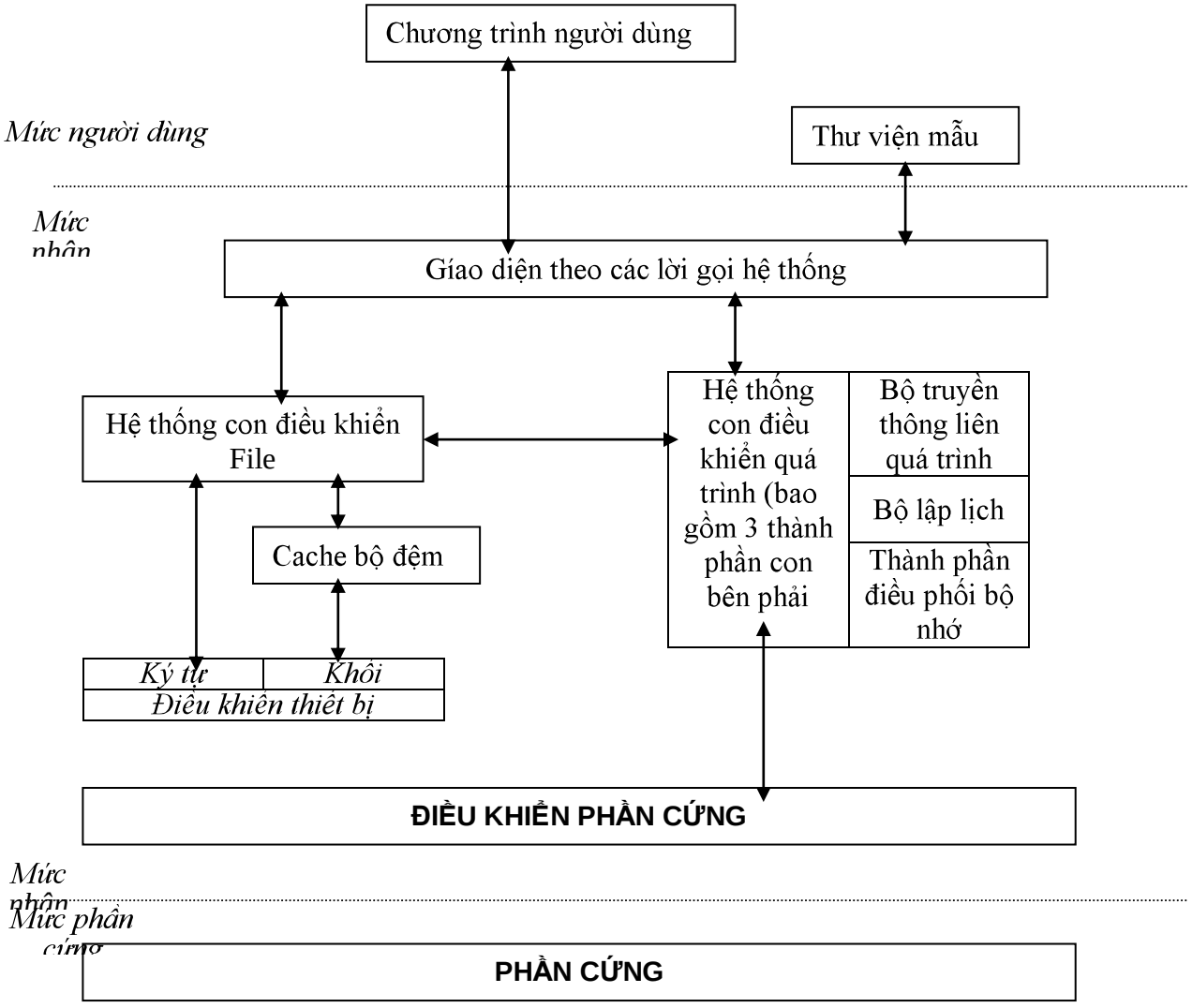
4.1.2. Sơ bộ cấu trúc điều khiển của UNIX

Theo phân cấp, hệ thống thực hiện theo ba mức: mức người dùng, mức nhân và mức phần cứng.

- Mức người dùng (user level): gồm có chương trình người dùng và chương trình trong các thư viện. Các chương trình này chạy (phần lệnh của chúng thực hiện) trong trạng thái người dùng của quá trình. Chương trình người dùng thao tác với nhân hoặc trực tiếp hoặc gián tiếp nhờ gọi thư viện nhờ các ***lời gọi hệ thống***.

Mức nhân là mức trọng tâm nhất của hệ điều hành Linux-UNIX. Chạy ở mức nhân là những chương trình của hệ điều hành thuộc hệ thống con điều khiển File (hệ thống con làm việc với File - File Subsystem), hệ thống con điều khiển quá trình (Proccess Control System), các lời gọi hệ thống (system calls), các chương trình điều khiển thiết bị (Device Drivers), Cache bộ đệm (Buffer cache) và các chương trình điều khiển phần cứng (Hard Control). Hai thành phần cơ bản nhất là Hệ thống điều khiển File và Hệ thống con điều khiển quá trình.

Hình vẽ dưới đây cho sơ bộ cấu trúc điều khiển trong UNIX:



Cấu trúc của Nhân và các mức quá trình

4.1.3. Các hệ thống con trong nhân

★ Hệ thống con điều khiển File có nhiệm vụ quản lý hệ thống File, cung cấp vùng nhớ rỗi ở đĩa cho File, điều khiển truy cập File và tìm kiếm dữ liệu v.v. Đa số các thuật toán về các

lời gọi hệ thống liên quan đến File và các hàm chương trình con mức thấp đã được trình bày trong chương 2. Các quá trình tương tác với Hệ thống con điều khiển File nhờ các lời gọi hệ thống (các lời gọi hệ thống File mức cao). Việc truy nhập tới File nhờ hai cách thức: truy nhập trực tiếp với File hoặc thông qua buffer cache.

- Các buffer cache lưu trữ dữ liệu tạm thời theo từng khối. Nhân vào-ra dữ liệu thông qua các khối trung gian và nhờ thiết bị nhớ thứ cấp: truy nhập dữ liệu theo khối,
- Nhân thao tác trực tiếp với khối điều khiển thiết bị để truy nhập trực tiếp dữ liệu trong File không qua thiết bị phụ: truy nhập theo ký tự.

★ Hệ thống con điều khiển quá trình chịu trách nhiệm đồng bộ hóa sự tương tác liên quá trình, quản lý bộ nhớ và lập lịch thực hiện đối với các quá trình đang tồn tại. Hệ thống con điều khiển File và Hệ thống con điều khiển quá trình tương tác với nhau khi file được tải vào bộ nhớ trong và cho thực hiện. Một số lời gọi hệ thống cho khối điều khiển quá trình:

- fork: Tạo quá trình mới. Lời gọi hàm này có dạng pid=fork()
- exec: Cho thực hiện quá trình đang tồn tại; exec(pid)
- exit: Cho kết thúc quá trình đang tồn tại,
- brk: điều khiển kích thước bộ nhớ cấp phát cho quá trình,
- signal: Điều khiển các hiện tượng bất thường trong quá trình

Hệ thống con điều khiển quá trình bao gồm 3 thành phần sau đây:

- Thành phần điều phối bộ nhớ có nhiệm vụ quản lý, điều khiển cấp phát bộ nhớ. Một số trang bị loại bỏ khi cấp phát bộ nhớ cho quá trình.
- Bộ lập lịch (schelduler) có nhiệm vụ điều phối CPU cho các quá trình. Các quá trình có độ ưu tiên và bộ lập lịch chọn quá trình có độ ưu tiên cao nhất.
- Bộ truyền thông liên quá trình thực hiện việc đồng bộ hóa các quá trình liên quan nhau.

★ Bộ điều khiển phần cứng (hardware control) có chức năng cho phép ngắt và tương tác thông tin với máy. Các thiết bị như đĩa, thiết bị đầu cuối có thể ngắt CPU khi đang thực hiện quá trình. Các chương trình xử lý ngắt là hàm riêng biệt trong nhân mà không phải là một quá trình.

Stack trong quá trình

Mỗi quá trình thực hiện được mode nhân và mode người dùng vì vậy phân chia hai loại stack nhân và stack người dùng.

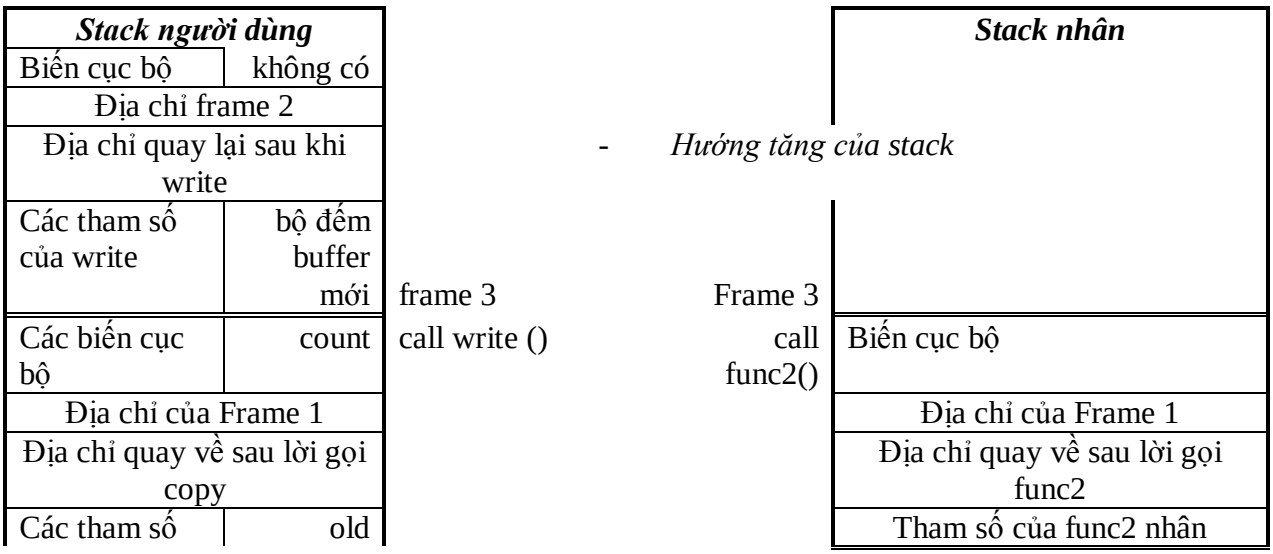
Chúng ta xem xét ví dụ sau:

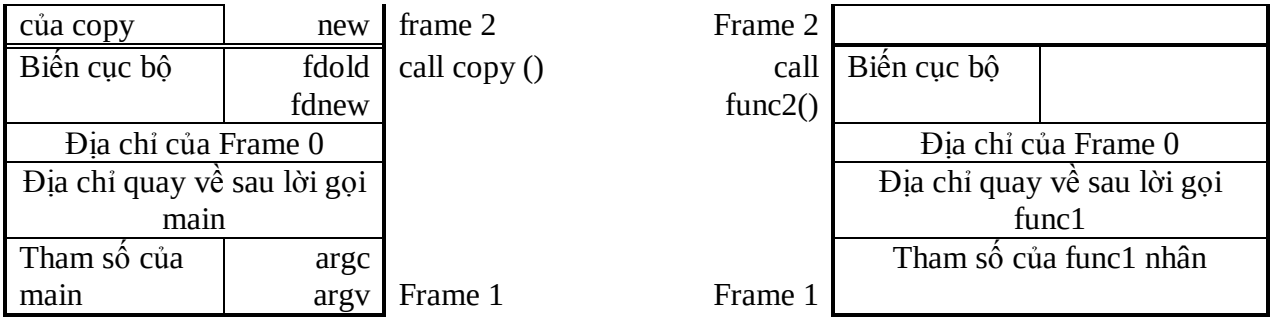
```
#include <fenth.h>
char buffer[2048];
int version;
main (argc, argv);
        int argc;
        char *argv[];
|
        int fdold, fdnew;
        if (argc != 3)
        |
                printf(' cần 2 đối số đối với chương trình sao file!');
```

```
            exit(1)
        |
        fdold = open (argv[1], O_RDONLY);          /* mở file nguồn chỉ đọc */
        if (fdold == -1)
        |
            printf (' Không thể mở file &cs\n',argv[1]);
            exit(1);
        |
        fdnew =creat (argv[2],0666); /*mở File đích rw cho mọi người */
        if (fdnew ==-1)
        |
            printf('Không thể khởi tạo file &cs\n',argv[2];
            exit(1);
        |
        copy(fdold,fdnew);
        exit(0);
|
copy (old, new)
        int old, new;
|
        int count;
        while (count = read(old,buffer,sizeof(buffer))>0)
            write(buffer,count);
|
```

Trong chương trình trên, mã lệnh (gọi là phần text) của file được sinh ra từ các hàm main và copy. Khởi tạo giá trị ban đầu cho biến version và dành vùng nhớ cho biến mảng buffer.

Trong ví dụ trên, các tham số argc, argv và các biến fdold, fdnew trong chương trình main trong stack khi main được gọi (một lần đối với mọi chương trình), còn các tham số old và new và biến count trong hàm copy xuất hiện mỗi khi copy được gọi.





Các stack cho một quá trình

Quá trình trong UNIX được thực hiện theo một trong hai mode: mode nhân hay mode người dùng và tương ứng với 2 mode này, quá trình sử dụng stack riêng biệt đối với mỗi mode.

Stack người dùng chứa các đối số, biến cục bộ, và các dữ liệu khác đối với việc thực hiện hàm trong mode người dùng. Stach nhân chứa các đối số, biến cục bộ, các tham số, các địa chỉ liên kết v.v. liên quan đến thực hiện các hàm theo mode nhân.

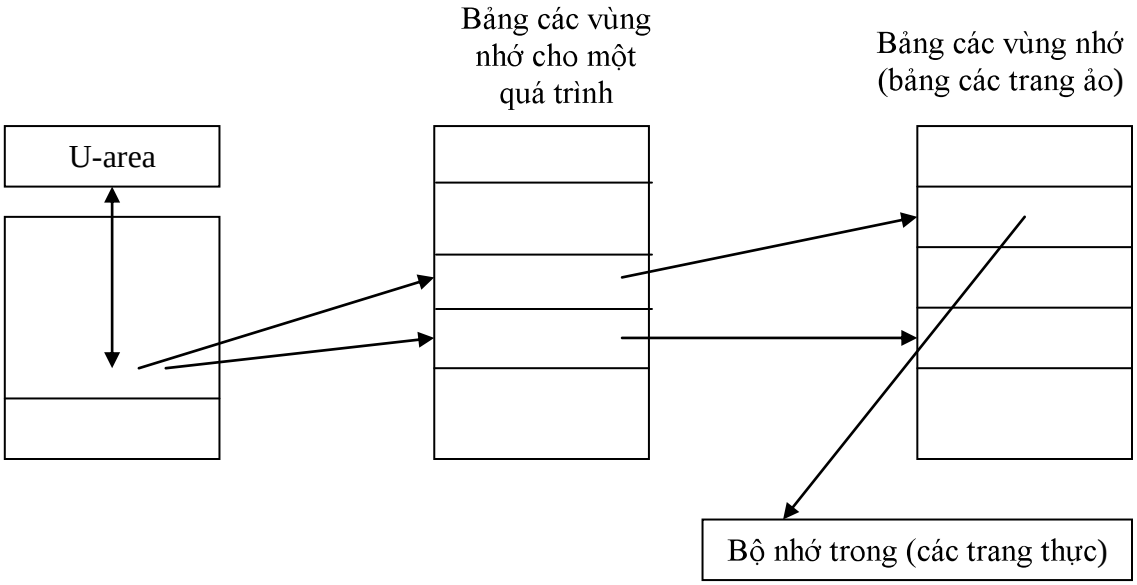
4.1.4. Sơ bộ về điều khiển quá trình

Nhân sử dụng 4 cấu trúc dữ liệu sau đây để truy nhập đến quá trình:

- Bảng các quá trình, tương ứng với mỗi quá trình đang tồn tại trong hệ thống là một thành phần. Mỗi thành phần bao gồm một số trường sau đây (mỗi thành phần ở đây chính là một PCB):
  - Trạng thái của quá trình,
  - Chủ sở hữu của quá trình,
  - Trường liên quan đến trạng thái ngưng của quá trình (theo lời gọi hàm sleep)
  - Địa chỉ của vùng sử dụng tương ứng với quá trình,
  - Các thông tin tương ứng được trình bày trong PCB.
- Vùng sử dụng (U-area) chứa các thông tin riêng, có tác dụng khi quá trình đang thực hiện:
  - Chỉ số thành phần tương ứng với quá trình trong bảng các quá trình: địa chỉ của khối PCB tương ứng,
  - Bộ đếm thời gian chạy mức nhân và mức người dùng,
  - Các giá trị trả về và mã lỗi (nếu có) đối với lời gọi hệ thống hiện tại,
  - Mô tả về các file đang mở ứng với quá trình,
  - Tham số lưu trữ dung lượng dữ liệu di chuyển trong vào - ra.
  - Thư mục hiện tại và thư mục gốc hiện tại: môi trường của quá trình,
  - Các giới hạn kích thước file và quá trình,
  - Các mức cho phép thực hiện đối với quá trình,
  - Một số thông tin khác
- Các bảng định vị địa chỉ bộ nhớ đối với mỗi quá trình,
- Bảng chứa vùng bộ nhớ chung: phân hoạch bộ nhớ, đặc tính mỗi vùng theo phân hoạch: chứa text, data hoặc vùng bộ nhớ dùng chung v.v.

Sơ bộ về mối liên kết của các cấu trúc dữ liệu trên được mô tả như hình vẽ phía sau. Nhân xử lý với các lời gọi hệ thống như sau:

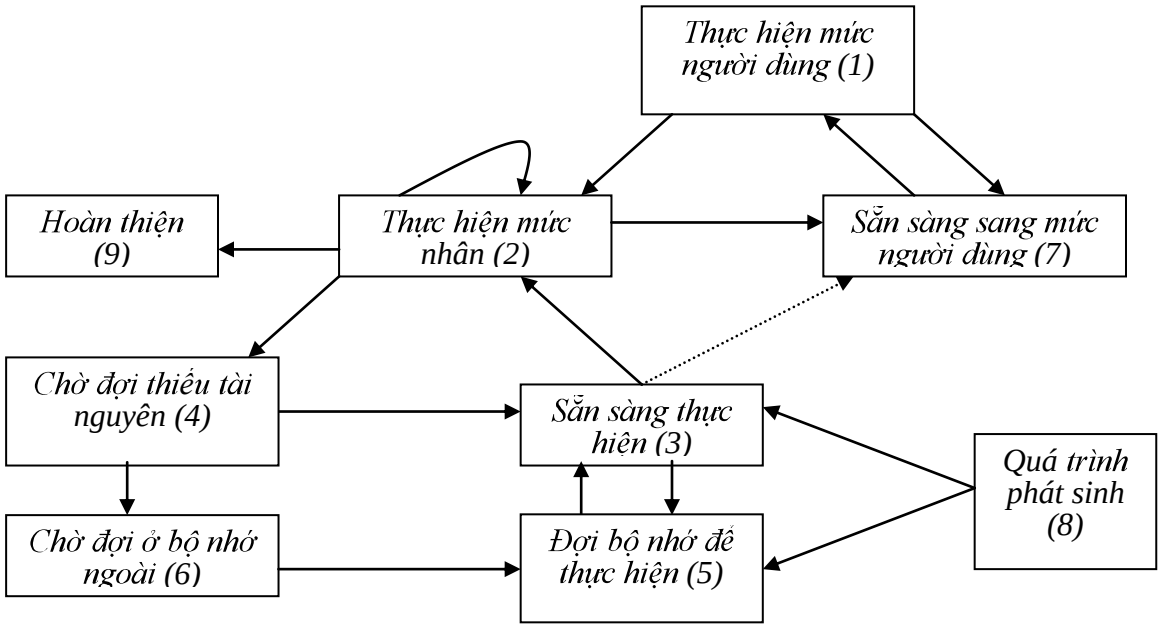
- Với lời gọi fork: Nhân sao vùng địa chỉ của quá trình cũ, cho phép các quá trình chia xẻ vùng bộ nhớ,
- Với lời gọi exec: Nhân cấp phát các vùng bộ nhớ thực cho các vùng text, data và stack,
- Với lời gọi exit: Nhân sẽ giải phóng các vùng bộ nhớ liên quan đến quá trình.



Các cấu trúc dữ liệu điều khiển quá trình

4.1.5. Trạng thái và chuyển dịch trạng thái

Sơ đồ biểu diễn các trạng thái và việc chuyển trạng thái trong UNIX được trình bày trong hình dưới đây (Số hiệu trạng thái quá trình xem trong hình vẽ).



Sơ đồ chuyển trạng thái quá trình

Khi quá trình được phát sinh nó ở trạng thái (8), tùy thuộc vào tình trạng bộ nhớ quá trình được phân phối bộ nhớ trong (3) hay bộ nhớ ngoài (5). Trạng thái (3) thể hiện quá trình đã sẵn sáng thực hiện, các thành phần của nó đã ở bộ nhớ trong chờ đợi CPU để thực hiện. Việc thực hiện tiếp theo tùy thuộc vào trạng thái trước đó của nó. Nếu lần đầu phát sinh, nó cần đi tới thực hiện mức nhân để hoàn thiện công việc lời gọi fork sẽ từ trạng thái (3) sang trạng thái (1), trong trường hợp khác, từ trạng thái (3) nó đi tới trạng thái chờ dợi CPU ở mức người dùng (7).

Trong trạng thái thực hiện ở mức người dùng (1), quá trình đi tới trạng thái (2) khi gặp lời gọi hệ thống hoặc hiện tượng ngắt xảy ra. Từ trạng thái (1) tới trạng thái (7) khi hết lượng tử thời gian.

Trạng thái (4) là trạng thái chờ đợi trong bộ nhớ còn trạng thái (6) thể hiện việc chờ đợi trong bộ nhớ ngoài.

Cung chuyển từ trạng thái (2) vào ngay trạng thái (2) xảy ra khi ở quá trình ở trạng thái thực hiện mức nhân, nhân hệ thống gọi các hàm xử lý ngắt tương ứng.

4.1.6. Sự ngưng hoạt động và hoạt động trở lại của quá trình

Một quá trình trong trạng thái thực hiện mức nhân có khả năng chuyển sang trạng thái *ngưng* theo lời gọi hàm *sleep*. Trạng thái ngưng xảy ra trong một số tình huống chờ đợi một sự kiện: hoàn thành việc vào-ra, quá trình khác thực hiện lời gọi exit v.v.

Sau khi sự kiện xảy ra, quá trình từ trạng thái ngưng chuyển sang trạng thái sẵn sàng để có thể được cấp phát CPU chạy.

4.1.7. Sơ bộ về lệnh đối với quá trình

Khi mở một trang **man**, liệt kê các file với lệnh **ls**, chạy trình soạn thảo **vi** hay chạy bất kỳ một lệnh nào trong Linux thì điều đó có nghĩa là đang khởi tạo một hoặc nhiều quá trình. Trong Linux, bất cứ chương trình nào đang chạy đều được coi là một quá trình. Có thể có nhiều quá trình cùng chạy một lúc. Ví dụ dòng lệnh **ls -l | sort | more** sẽ khởi tạo ba quá trình: **ls**, **sort** và **more**.

Quá trình có thể trải qua nhiều trạng thái khác nhau và tại một thời điểm một quá trình rơi vào một trong các trạng thái đó. Bảng dưới đây giới thiệu các trạng thái cơ bản của quá trình trong Linux.

Ký hiệu	Ý nghĩa
D	(**uninterruptible sleep**) ở trạng thái này quá trình bị treo và không thể chạy lại nó bằng một tín hiệu.
R	(**runnable**) trạng thái sẵn sàng thực hiện, tức là quá trình có thể thực hiện được nhưng chờ đến lượt thực hiện vì một quá trình khác đang có CPU.
S	(**sleeping**) trạng thái tạm dừng, tức là quá trình tạm dừng không hoạt động (20 giây hoặc ít hơn)
T	(**traced or stopped**) trạng thái dừng, quá trình có thể bị treo bởi một quá trình ngoài
Z	(**zombie process**) quá trình đã kết thúc thực hiện, nhưng nó vẫn được tham chiếu trong hệ thống
W	không có các trang thường trú

<	quá trình có mức ưu tiên cao hơn
N	quá trình có mức ưu tiên thấp hơn
L	có các trang khóa bên trong bộ nhớ

4.2. Các lệnh cơ bản

4.2.1. Lệnh fg và lệnh bg

Linux cho phép người dùng sử dụng tổ hợp phím **CTRL+z** để dừng một quá trình và khởi động lại quá trình đó bằng cách gõ lệnh **fg**. Lệnh **fg** (**foreground**) tham chiếu đến các chương trình mà màn hình cũng như bàn phím đang làm việc với chúng.

Ví dụ, người dùng đang xem trang **man** của lệnh **sort**, nhìn xuống cuối thấy có tùy chọn **-b**, muốn thử tùy chọn này đồng thời vẫn muốn xem trang **man**. Thay cho việc đánh **q** để thoát và sau đó chạy lại lệnh **man**, cho phép người dùng gõ **CTRL+z** để tạm dừng lệnh **man** và gõ lệnh thử tùy chọn **-b**. Sau khi thử xong, hãy gõ **fg** để tiếp tục xem trang **man** của lệnh **sort**. Kết quả của quá trình trên hiển thị như sau:

```
# man sort | more
SORT(1) FSF SORT(1)
NAME
sort - sort lines of text Files
SYNOPSIS
 ../src/sort [OPTION] ... [Files]...
DESCRIPTION
Write sorted concatenation of all FILE(s) to standard out-put.
 +POS1 [-POS2]
start a key at POS1,end it *before* POS2 obsoles-cent)field numbers and character offsets are num-bered starting with zero(contrast with the -k option)
-b ignore leading blanks in sort fields or keys
          --More--
(CTRL+z)
[1]+ Stopped     man sort | more
# ls -s | sort -b | head -4
  1 Archives/
  1 InfoWorld/
  1 Mail/
  1 News/
  1 OWL/
# fg
 man sort | more
 --More--
```

Trong phần trước, cách thức gõ phím **CTRL+z** để tạm dừng một quá trình đã được giới thiệu. Linux còn người dùng cách thức để chạy một chương trình dưới chế độ nền

(***background***) - sử dụng lệnh **bg** - trong khi các chương trình khác đang chạy, và để chuyển một chương trình vào trong chế độ nền - dùng ký hiệu **&**.

Nếu một quá trình hoạt động mà không đưa ra thông tin nào trên màn hình và không cần nhận bất kỳ thông tin đầu vào nào, thì có thể sử dụng lệnh **bg** để đưa nó vào trong chế độ nền (ở chế độ này nó sẽ tiếp tục chạy cho đến khi kết thúc). Khi chương trình cần đưa thông tin ra màn hình hoặc nhận thông tin từ bàn phím, hệ thống sẽ tự động dừng chương trình và thông báo cho người dùng. Cũng có thể sử dụng chỉ số điều khiển công việc (***job control***) để làm việc với chương trình nào muốn. Khi chạy một chương trình trong chế độ nền, chương trình đó được đánh số thứ tự (được bao bởi dấu ngoặc vuông []), theo sau là chỉ số của quá trình.

Sau đó có thể sử dụng lệnh **fg + số thứ tự của chương trình** để đưa chương trình trở lại chế độ nổi và tiếp tục chạy.

Để có một chương trình (hoặc một lệnh ống) tự động chạy trong chế độ nền, chỉ cần thêm ký hiệu '**&**' vào cuối lệnh.

Trong một số hệ thống, khi quá trình nền kết thúc thì hệ thống sẽ gửi thông báo tới người dùng, nhưng trên hầu hết các hệ thống, khi quá trình trên nền hoàn thành thì hệ thống sẽ chờ cho đến khi người dùng gõ phím **Enter** thì mới hiển thị dấu nhắc lệnh mới kèm theo thông báo hoàn thành quá trình (thường thì một quá trình hoàn thành sau khoảng 20 giây).

Nếu cố để chuyển một chương trình vào chế độ nền mặc dù nó có các thông tin cần xuất hoặc nhập từ các thiết bị vào ra chuẩn thì hệ thống sẽ đưa ra thông báo lỗi dưới dạng sau: **Stopped (tty input/output) tên chương trình**.

Ví dụ, lệnh sau đây thực hiện việc tìm kiếm file **thu1** trong chế độ nền:

```
# find -name thu1 &
[5] 918
```

trong chế độ này, số thứ tự của chương trình là **[5]**, chỉ số quá trình tương ứng với lệnh **find** là **918**. Vì gõ Enter khi quá trình chưa thực hiện xong nên trên màn hình chỉ hiển thị số thứ tự của chương trình và chỉ số quá trình, nếu chờ khoảng 30 hoặc 40 giây sau rồi gõ Enter lần nữa, màn hình hiển thị thông báo hoàn thành chương trình như sau:

```
#
[5] Done            find -name thu1
#
```

Giả sử chương trình chưa hoàn thành và muốn chuyển nó lên chế độ nổi, hãy gõ lệnh sau:

```
# fg 5
find -name thu1
./thu1
```

chương trình đã hoàn thành và hiển thị thông báo rằng file **thu1** nằm ở thư mục gốc.

Thông thường sẽ đưa ra một thông báo lỗi nếu người dùng cố chuyển một chương trình vào chế độ nền khi mà chương trình đó cần phải xuất hoặc nhập thông tin từ thiết bị vào ra chuẩn. Ví dụ, lệnh:

```
# vi &
[6] 920
#
```

nhấn Enter

```
#
[6] + Stopped (tty output) vi
#
```

Lệnh trên chạy chương trình **vi** trong chế độ nền, tuy nhiên lệnh gặp phải lỗi vì đây là chương trình đòi hỏi hiển thị các thông tin ra màn hình (**output**). Dòng thông báo lỗi **Stopped (tty intput) vi** cũng xảy ra khi chương trình **vi** cần nhận thông tin.

4.2.2. Hiển thị các quá trình đang chạy với lệnh ps

Linux cung cấp cho người dùng hai cách thức nhận biết có những chương trình nào đang chạy trong hệ thống. Cách dễ hơn, đó là lệnh **jobs** sẽ cho biết các quá trình nào đã dừng hoặc là được chạy trong chế độ nền.

Cách phức tạp hơn là sử dụng lệnh **ps**. Lệnh này cho biết thông tin đầy đủ nhất về các quá trình đang chạy trên hệ thống.

Ví dụ:

```
# ps
PID    TTY   TIME   CMD
7813   pts/0      00:00:00 bash
7908   pts/0      00:00:00 ps
#
```

(PID - chỉ số của quá trình, TTY - tên thiết bị đầu cuối trên đó quá trình được thực hiện, TIME - thời gian để chạy quá trình, CMD - lệnh khởi tạo quá trình).

Cú pháp lệnh **ps**:

```
ps [tùy-chọn]
```

Lệnh **ps** có một lượng quá phong phú các tùy chọn được chia ra làm nhiều loại. Dưới đây là một số các tùy chọn hay dùng. Các tùy chọn đơn giản:

- **-A, -e** : chọn để hiển thị tất cả các quá trình.
- **-T** : chọn để hiển thị các quá trình trên trạm cuối đang chạy.
- **-a** : chọn để hiển thị tất cả các quá trình trên một trạm cuối, bao gồm cả các quá trình của những người dùng khác.
- **-r** : chỉ hiển thị quá trình đang được chạy.
  - ***Chọn theo danh sách***
- **-C** : chọn hiển thị các quá trình theo tên lệnh.
- **-G** : hiển thị các quá trình theo chỉ số nhóm người dùng.
- **-U** : hiển thị các quá trình theo tên hoặc chỉ số của người dùng thực sự (người dùng khởi động quá trình).
- **-p** : hiển thị các quá trình theo chỉ số của quá trình.
- **-s** : hiển thị các quá trình thuộc về một phiên làm việc.
- **-t** : hiển thị các quá trình thuộc một trạm cuối.
- **-u** : hiển thị các quá trình theo tên và chỉ số của người dùng hiệu quả.
  - ***Thiết đặt khuôn dạng được đưa ra của các quá trình***
- **-f** : hiển thị thông tin về quá trình với các trường sau UID - chỉ số người dùng, PID - chỉ số quá trình, PPID - chỉ số quá trình khởi tạo ra quá trình, C - , STIME - thời gian khởi tạo quá

trình, TTY - tên thiết bị đầu cuối trên đó quá trình được chạy, TIME - thời gian để thực hiện quá trình, CMD - lệnh khởi tạo quá trình

- **-l** : hiển thị đầy đủ các thông tin về quá trình với các trường F, S, UID, PID, PPID, C, PRI, NI, ADDR, SZ, WCHAN, TTY, TIME, CMD
- **-o xâu-chọn** : hiển thị các thông tin về quá trình theo dạng do người dùng tự chọn thông qua xâu-chọn các kí hiệu điều khiển hiển thị có các dạng như sau:

%C, %cpu	% CPU được sử dụng cho quá trình
%mem	% bộ nhớ được sử dụng để chạy quá trình
%G	tên nhóm người dùng
%P	chỉ số của quá trình cha khởi động ra quá trình con
%U	định danh người dùng
%c	lệnh tạo ra quá trình
%p	chỉ số của quá trình
%x	thời gian để chạy quá trình
%y	thiết bị đầu cuối trên đó quá trình được thực hiện

Ví dụ, muốn xem các thông tin như tên người dùng, tên nhóm, chỉ số quá trình, chỉ số quá trình khởi tạo ra quá trình, tên thiết bị đầu cuối, thời gian chạy quá trình, lệnh khởi tạo quá trình, hãy gõ lệnh:

```
# ps -o '%U %G %p %P %y %x %c'

USER GROUP PID PPID TTY TIME COMMAND
root root 1929 1927 pts/1 00:00:00 bash
root root 2279 1929 pts/1 00:00:00 ps
```

4.2.3. Hủy quá trình với lệnh kill

Trong một số trường hợp, sử dụng lệnh **kill** để hủy bỏ một quá trình. Điều quan trọng nhất khi sử dụng lệnh **kill** là phải xác định được chỉ số của quá trình mà chúng ta muốn hủy. Cú pháp lệnh:

```
kill [tùy-chọn] <chỉ-số-của-tiến-trình>
kill -l [tín hiệu]
```

Lệnh **kill** sẽ gửi một **tín hiệu** đến quá trình được chỉ ra. Nếu không chỉ ra một tín hiệu nào thì ngầm định là tín hiệu **TERM** sẽ được gửi.

- **-s** : xác định tín hiệu được gửi. Tín hiệu có thể là số hoặc tên của tín hiệu. Dưới đây là một số tín hiệu hay dùng:

Số	Tên	Ý nghĩa
1	SIGHUP	(**hang up**) đây là tín hiệu được gửi đến tất cả các quá trình đang chạy trước khi **logout** khỏi hệ thống
2	SIGINT	(**interrupt**) đây là tín hiệu được gửi khi nhấn **CTRL+c**
9	SIGKILL	(**kill**) tín hiệu này sẽ dừng quá trình ngay lập tức
15	SIGTERM	tín hiệu này yêu cầu dừng quá trình ngay lập tức,

nhưng cho phép chương trình xóa các file tạm.

- **-p** : lệnh **kill** sẽ chỉ đưa ra chỉ số của quá trình mà không gửi một tín hiệu nào.
- **-l** : hiển thị danh sách các tín hiệu mà lệnh **kill** có thể gửi đến các quá trình (các tín hiệu này có trong file **/usr/include/Linux/signal.h**)

Ví dụ,

```
# ps
PID TTY TIME CMD

 2240 pts/2 00:00:00 bash
 2276 pts/2 00:00:00 man
 2277 pts/2 00:00:00 more
 2280 pts/2 00:00:00 sh
 2281 pts/2 00:00:00 sh
 2285 pts/2 00:00:00 less
 2289 pts/2 00:00:00 man
 2291 pts/2 00:00:00 sh
 2292 pts/2 00:00:00 gunzip
 2293 pts/2 00:00:00 less
 2298 pts/2 00:00:00 ps
# kill 2277

PID TTY TIME CMD
2240 pts/2 00:00:00 bash
2276 pts/2 00:00:00 man
2280 pts/2 00:00:00 sh
2281 pts/2 00:00:00 sh
2285 pts/2 00:00:00 less
2289 pts/2 00:00:00 man
2291 pts/2 00:00:00 sh
2292 pts/2 00:00:00 gunzip
2293 pts/2 00:00:00 less
2298 pts/2 00:00:00 ps
```

4.2.4. Cho máy ngừng hoạt động một thời gian với lệnh sleep

Nếu muốn cho máy nghỉ một thời gian mà không muốn tắt vì ngại khởi động lại thì cần dùng lệnh **sleep**. Cú pháp:

```
sleep [tùy-chọn] NUMBER[SUFFIXƯ]
```

- NUMBER: số giây(s) ngừng hoạt động.
- SUFFIX : có thể là giây(s) hoặc phút(m) hoặc giờ hoặc ngày(d)

Các tùy chọn:

- **--help** : hiện thị trợ giúp và thoát

- **--version** : hiển thị thông tin về phiên bản và thoát

4.2.5. Xem cây quá trình với lệnh pstree

Đã biết lệnh để xem các quá trình đang chạy trên hệ thống, tuy nhiên trong Linux còn có một lệnh cho phép có thể nhìn thấy mức độ phân cấp của các quá trình, đó là lệnh **pstree**. Cú pháp lệnh:

```
pstree [tùy-chọn] [pid | người-dùng]
```

Lệnh **pstree** sẽ hiển thị các quá trình đang chạy dưới dạng cây quá trình. Gốc của cây quá trình thường là init. Nếu đưa ra tên của một người dùng thì cây của các quá trình do người dùng đó sở hữu sẽ được đưa ra.

pstree thường gộp các nhánh quá trình trùng nhau vào trong dấu ngoặc vuông, ví dụ:

```
init -+-getty
      |-getty
      |-getty
      |-getty
```

thành

```
init ---4*[getty]
```

- **-a** : chỉ ra tham số dòng lệnh. Nếu dòng lệnh của một quá trình được tráo đổi ra bên ngoài, nó được đưa vào trong dấu ngoặc đơn.
- **-c** : không thể thu gọn các cây con đồng nhất. Mặc định, các cây con sẽ được thu gọn khi có thể
- **-h** : hiển thị quá trình hiện thời và "tổ tiên" của nó với màu sáng trắng
- **-H** : giống như tùy chọn -h, nhưng quá trình con của quá trình hiện thời không có màu sáng trắng
- **-l** : hiển thị dòng dài.
- **-n** : sắp xếp các quá trình cùng một tổ tiên theo chỉ số quá trình thay cho sắp xếp theo tên

Ví dụ,

```
# pstree
init-+-apmd
 |-atd
 |-automount
 |-crond
 |-enlightenment
 |-gdm-+-X
 |  `-gdm---gnome-session
 |-gen_util_applet
 |-gmc
 |-gnome-name-serv
 |-gnome-smproxy
 |-gnomepager_appl
 |-gpm
 |-identd---identd---3*[identd]
 |-inetd
```

```
|-kflushd
|-klogd
|-kpiod
|-kswapd
|-kupdate
|-lockd---rpciod
|-login---bash---mc-+-bash-+-cat
| | |-passwd
| | `-pstree
| `-cons.saver
|-lpd
|-mdrecoveryd
|-5*[mingetty]
|-panel
|-portmap
|-rpc.statd
|-sendmail
|-syslogd
`-xfs
```

4.2.6. Lệnh thiết đặt lại độ ưu tiên của quá trình nice và lệnh renice

Ngoài các lệnh xem và hủy bỏ quá trình, trong Linux còn có hai lệnh liên quan đến độ ưu tiên của quá trình, đó là lệnh **nice** và lệnh **renice**.

Để chạy một chương trình với độ ưu tiên định trước, hãy sử dụng lệnh **nice**.

Cú pháp lệnh:

```
nice [tùy-chọn] [lệnh [tham-số ]... ]
```

Lệnh **nice** sẽ chạy một chương trình (lệnh) theo độ ưu tiên đã sắp xếp. Nếu không có **lệnh**, mức độ ưu tiên hiện tại sẽ hiển thị. Độ ưu tiên được sắp xếp từ -20 (mức ưu tiên cao nhất) đến 19 (mức ưu tiên thấp nhất).

- **-ADJUST** : tăng độ ưu tiên theo ADJUST đầu tiên
- **--help** : hiển thị trang trợ giúp và thoát

Để thay đổi độ ưu tiên của một quá trình đang chạy, hãy sử dụng lệnh **renice**.

Cú pháp lệnh:

```
renice <độ-uu-tiên> [tùy-chọn]
```

Lệnh **renice** sẽ thay đổi mức độ ưu tiên của một hoặc nhiều quá trình đang chạy.

- **-g :** thay đổi quyền ưu tiên theo nhóm người dùng
- **-p :** thay đổi quyền ưu tiên theo chỉ số của quá trình
- **-u :** thay đổi quyền ưu tiên theo tên người dùng

Ví dụ:

```
# renice +1 987 -u daemon root -p 32
```

lệnh trên sẽ thay đổi mức độ ưu tiên của quá trình có chỉ số là **987** và **32**, và tất cả các quá trình do người dùng **daemon** và **root** sở hữu.

CHƯƠNG 5. QUẢN LÝ TÀI KHOẢN NGƯỜI DÙNG

Chương này cung cấp một số công cụ hữu ích trong Linux để quản lý các tài khoản người dùng trên hệ thống.

5.1 Tài khoản người dùng

Như đã biết, trong hệ điều hành đa người dùng, cần phân biệt người dùng khác nhau do quyền sở hữu các tài nguyên trong hệ thống, chẳng hạn như, mỗi người dùng có quyền hạn với file, quá trình của riêng họ. Điều này vẫn rất quan trọng thậm chí cả khi máy tính chỉ có một người sử dụng tại một thời điểm. Mọi truy cập hệ thống Linux đều thông qua tài khoản người dùng. Vì thế, mỗi người sử dụng được gắn với tên duy nhất (đã được đăng ký) và tên đó được sử dụng để đăng nhập. Tuy nhiên một người dùng thực sự có thể có nhiều tên đăng nhập khác nhau. Tài khoản người dùng có thể hiểu là tất cả các file, các tài nguyên, và các thông tin thuộc về người dùng đó.

Khi cài đặt hệ điều hành Linux, đăng nhập ***root*** sẽ được tự động tạo ra. Đăng nhập này được xem là thuộc về siêu người dùng (người dùng cấp cao, người quản trị), vì khi đăng nhập với tư cách người dùng root, có thể làm bất cứ điều gì muốn trên hệ thống. Tốt nhất chỉ nên đăng nhập **root** khi thực sự cần thiết, và hãy đăng nhập vào hệ thống với tư cách là một người dùng bình thường.

Nội dung chương này giới thiệu các lệnh để tạo một người dùng mới, thay đổi thuộc tính của một người dùng cũng như xóa bỏ một người dùng. Lưu ý, chỉ có thể thực hiện được các lệnh trên nếu có quyền của một siêu người dùng.

5.2 Các lệnh cơ bản quản lý người dùng

Người dùng được quản lý thông qua tên người dùng (thực ra là chỉ số người dùng). Nhân hệ thống quản lý người dùng theo chỉ số, vì việc quản lý theo chỉ số sẽ dễ dàng và nhanh thông qua một cơ sở dữ liệu lưu trữ các thông tin về người dùng. Việc thêm một người dùng mới chỉ có thể thực hiện được nếu đăng nhập với tư cách là siêu người dùng.

Để tạo một người dùng mới, cần phải thêm thông tin về người dùng đó vào trong cơ sở dữ liệu người dùng, và tạo một thư mục cá nhân cho riêng người dùng đó. Điều này rất cần thiết để thiết lập các biến môi trường phù hợp cho người dùng.

Lệnh chính để thêm người dùng trong hệ thống Linux là **useradd** (hoặc **adduser**).

5.2.1 File /etc/passwd

Danh sách người dùng cũng như các thông tin tương ứng được lưu trữ trong file **/etc/passwd**.

Ví dụ dưới đây là nội dung của file **/etc/passwd**:

```
mail:x:8:12:mail:/var/spool/mail:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
bien:x:500:0:Nguyen Thanh Bien:/home/bien:/bin/bash
sangnm:x:17:100:Nguyen Minh Sang:/home/sangnm:/bin/bash
lan:x:501:0:Lan GNU:/home/lan:/bin/bash
```

Mỗi dòng trong file tương ứng với bảy trường thông tin của một người dùng, và các trường này được ngăn cách nhau bởi dấu '**:**'. ý nghĩa của các trường thông tin đó lần lượt như sau:

- Tên người dùng (**username**)
- Mật khẩu người dùng (**passwd** - được mã hóa)
- Chỉ số người dùng (**user id**)
- Các chỉ số nhóm người dùng (**group id**)
- Tên đầy đủ hoặc các thông tin khác về tài khoản người dùng (**comment**)
- Thư mục để người dùng đăng nhập
- Shell đăng nhập (chương trình chạy lúc đăng nhập)

Bất kỳ người dùng nào trên hệ thống đều có thể đọc được nội dung file **/etc/passwd**, và có thể đăng nhập với tư cách người dùng khác nếu họ biết được mật khẩu, đây chính là lý do vì sao mật khẩu đăng nhập của người dùng không hiển thị trong nội dung file.

5.2.2 Thêm người dùng với lệnh useradd

Siêu người dùng sử dụng lệnh **useradd** để tạo một người dùng mới hoặc cập nhật ngầm định các thông tin về người dùng.

Cú pháp lệnh:

```
useradd [tùy-chọn] <tên-người-dùng>
useradd -D [tùy-chọn]
```

Nếu không có tùy chọn **-D**, lệnh **useradd** sẽ tạo một tài khoản người dùng mới sử dụng các giá trị được chỉ ra trên dòng lệnh và các giá trị mặc định của hệ thống. Tài khoản người dùng mới sẽ được nhập vào trong các file hệ thống, thư mục cá nhân sẽ được tạo, hay các file khởi tạo được sao chép, điều này tùy thuộc vào các tùy chọn được đưa ra.

Các tùy chọn như sau:

- **-c, comment** : soạn thảo trường thông tin về người dùng.
- **-d, home_dir** : tạo thư mục đăng nhập cho người dùng.
- **-e, expire_date** : thiết đặt thời gian (YYYY-MM-DD) tài khoản người dùng sẽ bị hủy bỏ.
- **-f, inactive_days** : tùy chọn này xác định số ngày trước khi mật khẩu của người dùng hết hiệu lực khi tài khoản bị hủy bỏ. Nếu =0 thì hủy bỏ tài khoản người dùng ngay sau khi mật khẩu hết hiệu lực, =-1 thì ngược lại (mặc định là -1).
- **-g, initial_group** : tùy chọn này xác định tên hoặc số khởi tạo đăng nhập nhóm người dùng. Tên nhóm phải tồn tại, và số của nhóm phải tham chiếu đến một nhóm đã tồn tại. Số nhóm ngầm định là 1.
- **-G, group** : danh sách các nhóm phụ mà người dùng cũng là thành viên thuộc các nhóm đó. Mỗi nhóm sẽ được ngăn cách với nhóm khác bởi dấu ',', mặc định người dùng sẽ thuộc vào nhóm khởi tạo.
- **-m** : với tùy chọn này, thư mục cá nhân của người dùng sẽ được tạo nếu nó chưa tồn tại.
- **-M** : không tạo thư mục người dùng.
- **-n** : ngầm định khi thêm người dùng, một nhóm cùng tên với người dùng sẽ được tạo. Tùy chọn này sẽ loại bỏ sự ngầm định trên.
- **-p, passwd** : tạo mật khẩu đăng nhập cho người dùng.
- **-s, shell** : thiết lập shell đăng nhập cho người dùng.
- **-u, uid** : thiết đặt chỉ số người dùng, giá trị này phải là duy nhất.
  - Thay đổi các giá trị ngầm định

Khi tùy chọn **-D** được sử dụng, lệnh **useradd** sẽ bỏ qua các giá trị ngầm định và cập nhật các giá trị mới.

- **-b, default_home** : thêm tên người dùng vào cuối thư mục cá nhân để tạo tên thư mục cá nhân mới.
- **-e, default_expire_date** : thay đổi thời hạn hết giá trị của tài khoản người dùng.
- **-f, default_inactive** : xác định thời điểm hết hiệu lực của mật khẩu đăng nhập khi tài khoản người dùng bị xóa bỏ.
- **-g, default_group** : thay đổi chỉ số nhóm người dùng.
- **-s, default_shell** : thay đổi shell đăng nhập.

Ngoài lệnh **useradd**, có thể tạo người dùng mới bằng cách sau:

Soạn thảo file **/etc/passwd** bằng **vipw**. Lệnh **vipw** mở trình soạn thảo trên hệ thống và hiệu chỉnh bản sao tạm của file **/etc/passwd**. Việc sử dụng file tạm và khóa file sẽ có tác dụng như một cơ chế khóa để ngăn việc hai người dùng cùng soạn thảo file một lúc. Lúc đó sẽ thêm dòng thông tin mới về người dùng cần tạo. Hãy cẩn thận trong việc soạn thảo tránh nhầm lẫn. Riêng trường mật khẩu nên để trống và tạo mật khẩu sau. Khi file này được lưu, **vipw** sẽ kiểm tra sự đồng nhất trên file bị thay đổi. Nếu tất cả mọi thứ dường như thích hợp thì có nghĩa là file **/etc/passwd** đã được cập nhật.

Ví dụ: thêm người dùng có tên là **new**, chỉ số người dùng **503**, chỉ số nhóm là **100**, thư mục cá nhân là **/home/new** và shell đăng nhập là shell bash:

```
# vipw
mail:x:8:12:mail:/var/spool/mail:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
bien:x:500:0:Nguyen Thanh Bien:/home/bien:/bin/bash
sang:x:17:100:Nguyen Minh Sang:/home/sangnm:/bin/bash
lan:x:501:0:Lan GNU:/home/lan:/bin/bash
new::503:100:them mot nguoi moi:/home/new:/bin/bash
```

Tạo thư mục cá nhân của người dùng mới với lệnh *mkdir*

```
# mkdir /home/new
```

- Sao chép các file từ thư mục **/etc/skel/** (đây là thư mục lưu trữ các file cần thiết cho người dùng) vào file cá nhân vừa tạo
- Thay đổi quyền sở hữu và các quyền truy nhập file **/home/new** với các lệnh **chown** và **chmod**

```
# chown new /home/new
# chmod go=u,go-w /home/new
```

Thiết lập mật khẩu của người dùng với lệnh *passwd*

```
# passwd new
passwd:
```

Sau khi thiết lập mật khẩu cho người dùng ở bước cuối cùng, tài khoản người dùng sẽ làm việc. Nên thiết lập mật khẩu người dùng ở bước cuối cùng, nếu không họ có thể vô tình đăng nhập trong khi đang sao chép các file.

5.2.3 Thay đổi thuộc tính người dùng

Trong Linux có rất nhiều lệnh cho phép thay đổi một số các thuộc tính của tài khoản người dùng như:

- **chfn**: thay đổi thông tin cá nhân của người dùng.
- **chsh**: thay đổi shell đăng nhập.
- **passwd**: thay đổi mật khẩu.

Một số các thuộc tính khác sẽ phải thay đổi bằng tay. Ví dụ, để thay đổi tên người dùng, cần soạn thảo lại trực tiếp trên file **/etc/passwd** (với lệnh **vipw**).

Nhưng có một lệnh tổng quát cho phép có thể thay đổi bất kỳ thông tin nào về tài khoản người dùng, đó là lệnh **usermod**.

Cú pháp lệnh:

```
usermod [tùy-chọn] <tên-đăng-nhập>
```

Lệnh **usermod** sửa đổi các file tài khoản hệ thống theo các thuộc tính được xác định trên dòng lệnh.

Các tùy chọn của lệnh:

- **-c, comment** : thay đổi thông tin cá nhân của tài khoản người dùng.
- **-d, home_dir** : thay đổi thư mục cá nhân của tài khoản người dùng.
- **-e, expire_date** : thay đổi thời điểm hết hạn của tài khoản người dùng (YYYY-MM-DD).
- **-f, inactive_days** : thiết đặt số ngày hết hiệu lực của mật khẩu trước khi tài khoản người dùng hết hạn sử dụng.
- **-g, initial_group** : tùy chọn này thay đổi tên hoặc số khởi tạo đăng nhập nhóm người dùng. Tên nhóm phải tồn tại, và số của nhóm phải tham chiếu đến một nhóm đã tồn tại. Số nhóm ngầm định là 1.
- **-G, group** : thay đổi danh sách các nhóm phụ mà người dùng cũng là thành viên thuộc các nhóm đó. Mỗi nhóm sẽ được ngăn cách với nhóm khác bởi dấu ',' mặc định người dùng sẽ thuộc vào nhóm khởi tạo.
- **-l, login_name** : thay đổi tên đăng nhập của người dùng. Trong một số trường hợp, tên thư mục riêng của người dùng có thể sẽ thay đổi để tham chiếu đến tên đăng nhập mới.
- **-p, passwd** : thay đổi mật khẩu đăng nhập của tài khoản người dùng.
- **-s, shell** : thay đổi shell đăng nhập.
- **-u, uid** : thay đổi chỉ số người dùng.

Lệnh **usermod** không cho phép thay đổi tên của người dùng đang đăng nhập. Phải đảm bảo rằng người dùng đó không thực hiện bất kỳ quá trình nào trong khi lệnh **usermod** đang thực hiện thay đổi các thuộc tính của người dùng đó.

Ví dụ muốn thay đổi tên người dùng **new** thành tên mới là **newuser**, hãy gõ lệnh sau:

```
# usermod -l new newuser
```

5.2.4 Xóa bỏ một người dùng (lệnh userdel)

Để xóa bỏ một người dùng, trước hết phải xóa bỏ mọi thứ có liên quan đến người dùng đó.

Lệnh hay được dùng để xóa bỏ một tài khoản người dùng là lệnh **userdel** với cú pháp:

```
userdel [-r] <tên-người-dùng>
```

Lệnh này sẽ thay đổi nội dung của các file tài khoản hệ thống bằng cách xóa bỏ các thông tin về người dùng được đưa ra trên dòng lệnh. Người dùng này phải thực sự tồn tại. Tuỳ chọn **-r** có ý nghĩa:

- **-r** : các file tồn tại trong thư mục riêng của người dùng cũng như các file nằm trong các thư mục khác có liên quan đến người dùng bị xóa bỏ cùng lúc với thư mục người dùng.

Lệnh **userdel** sẽ không cho phép xóa bỏ người dùng khi họ đang đăng nhập vào hệ thống. Phải hủy bỏ mọi quá trình có liên quan đến người dùng trước khi xoá bỏ người dùng đó.

Ngoài ra cũng có thể xóa bỏ tài khoản của một người dùng bằng cách hiệu chỉnh lại file **/etc/passwd**.

5.3 Các lệnh cơ bản liên quan đến nhóm người dùng

Mỗi người dùng trong hệ thống Linux đều thuộc vào một nhóm người dùng cụ thể. Tất cả những người dùng trong cùng một nhóm có thể cùng truy nhập một trình tiện ích, hoặc đều cần truy cập một thiết bị nào đó như máy in chẳng hạn.

Một người dùng cùng lúc có thể là thành viên của nhiều nhóm khác nhau, tuy nhiên tại một thời điểm, người dùng chỉ thuộc vào một nhóm cụ thể.

Nhóm có thể thiết lập các quyền truy nhập để các thành viên của nhóm đó có thể truy cập thiết bị, file, hệ thống file hoặc toàn bộ máy tính mà những người dùng khác không thuộc nhóm đó không thể truy cập được.

5.3.1 Nhóm người dùng và file /etc/group

Thông tin về nhóm người dùng được lưu trong file **/etc/group**, file này có cách bố trí tương tự như file **/etc/passwd**. Ví dụ nội dung của file **/etc/group** có thể như sau:

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
disk:x:6:root
lp:x:7:daemon,lp
mail:x:12:mail
huyen:x:500:
langnu:x:501:
```

Mỗi dòng trong file có bốn trường được phân cách bởi dấu '**:**'. ý nghĩa của các trường theo thứ tự xuất hiện như sau:

- Tên nhóm người dùng (groupname)
- Mật khẩu nhóm người dùng (**passwd** - được mã hóa), nếu trường này rỗng, tức là nhóm không yêu cầu mật khẩu
- Chỉ số nhóm người dùng (group id)
- Danh sách các người dùng thuộc nhóm đó (users)

5.3.2 Thêm nhóm người dùng

Cho phép hiệu chỉnh thông tin trong file **/etc/group** bằng bất kỳ trình soạn thảo văn bản nào có trên hệ thống của để thêm nhóm người dùng, nhưng cách nhanh nhất là sử dụng lệnh **groupadd**.

Cú pháp lệnh :

```
groupadd [tùy-chọn] <tên-nhóm>
```

Các tuỳ chọn là:

- **-g, gid** : tùy chọn này xác định chỉ số nhóm người dùng, chỉ số này phải là duy nhất. Chỉ số mới phải có giá trị lớn hơn 500 và lớn hơn các chỉ số nhóm đã có trên hệ thống. Giá trị từ 0 đến 499 chỉ dùng cho các nhóm hệ thống.
- **-r** : tùy chọn này được dùng khi muốn thêm một tài khoản hệ thống.
- **-f** : tùy chọn này sẽ bỏ qua việc nhắc nhở, nếu nhóm người dùng đó đã tồn tại, nó sẽ bị ghi đè.

Ví dụ:

- Thêm nhóm người dùng bằng cách soạn thảo file **/etc/group**:

```
installer:x:102:hieu, huy, sang
tiengviet:x:103:minh, long, dung
```

Hai dòng trên sẽ bổ sung hai nhóm người dùng mới cùng danh sách các thành viên trong nhóm: nhóm **installer** với chỉ số nhóm là **102** và các thành viên là các người dùng có tên **hieu**, **huy**, **sang**. Tương tự là nhóm **tiengviet** với chỉ số nhóm là **103** và danh sách các thành viên là **minh**, **long**, **dung**. Đây là hai nhóm (**102**, **103**) người dùng hệ thống.

- Thêm nhóm người dùng mới với lệnh **groupadd**:

```
# groupadd -r installer
```

Lệnh trên sẽ cho phép tạo một nhóm người dùng mới có tên là **installer**, tuy nhiên các thành viên trong nhóm sẽ phải bổ sung bằng cách soạn thảo file **/etc/group**.

5.3.3 Sửa đổi các thuộc tính của một nhóm người dùng (lệnh groupmod)

Trong một số trường hợp cần phải thay đổi một số thông tin về nhóm người dùng bằng lệnh **groupmod** với cú pháp như sau:

```
groupmod [tùy-chọn] <tên-nhóm>
```

Thông tin về các nhóm xác định qua tham số **tên-nhóm** được điều chỉnh.

Các tùy chọn của lệnh:

- **-g, gid** : thay đổi giá trị chỉ số của nhóm người dùng.
- **-n, group_name** : thay đổi tên nhóm người dùng.

5.3.4 Xóa một nhóm người dùng (lệnh groupdel)

Nếu không muốn một nhóm nào đó tồn tại nữa thì chỉ việc xóa tên nhóm đó trong file **/etc/group**. Nhưng phải lưu ý rằng, chỉ xóa được một nhóm khi không có người dùng nào thuộc nhóm đó nữa.

Ngoài ra có thể sử dụng lệnh **groupdel** để xóa một nhóm người dùng.

Cú pháp lệnh:

```
groupdel <tên-nhóm>
```

Lệnh này sẽ sửa đổi các file tài khoản hệ thống, xóa tất cả các thực thể liên quan đến nhóm. Tên nhóm phải thực sự tồn tại.

5.4 Các lệnh cơ bản khác có liên quan đến người dùng

Ngoài các lệnh như thêm người dùng, xóa người dùng ..., còn có một số lệnh khác có thể giúp ích rất nhiều nếu đang làm việc trên một hệ thống đa người dùng.

5.4.1 Đăng nhập với tư cách một người dùng khác khi dùng lệnh su

Đôi lúc muốn thực hiện lệnh như một người dùng khác và sử dụng các file hay thiết bị thuộc quyền sở hữu của người dùng đó. Lệnh **su** cho phép thay đổi tên người dùng một cách hiệu quả và cấp cho các quyền truy nhập của người dùng đó.

Cú pháp lệnh:

```
su <người-dùng>
```

Nếu đăng nhập với tư cách người dùng bình thường và muốn trở thành siêu người dùng (root) dùng lệnh sau:

```
# su root
```

Khi đó hệ thống sẽ yêu cầu nhập mật khẩu của siêu người dùng. Nếu cung cấp đúng mật mã, thì sẽ là người dùng **root** cho tới khi dùng lệnh **exit** hoặc **CTRL+d** để đăng xuất ra khỏi tài khoản này và trở về đăng nhập ban đầu. Tương tự, nếu đăng nhập với tư cách **root** và muốn trở thành người dùng bình thường có tên là **newer** thì hãy gõ lệnh sau:

```
# su newer
```

sẽ không bị hỏi về mật khẩu khi thay đổi từ siêu người dùng sang một người dùng khác. Tuy nhiên nếu đăng nhập với tư cách người dùng bình thường và muốn chuyển đổi sang một đăng nhập người dùng khác thì phải cung cấp mật khẩu của người dùng đó.

5.4.2 Xác định người dùng đang đăng nhập (lệnh who)

* Lệnh **who** là một lệnh đơn giản, cho biết được hiện tại có những ai đang đăng nhập trên hệ thống với cú pháp như sau:

```
who [tùy-chọn]
```

Các tuỳ chọn là:

- **-H, --heading** : hiển thị tiêu đề của các cột trong nội dung lệnh.
- **-m** : hiển thị tên máy và tên người dùng với thiết bị vào chuẩn.
- **-q, --count** : hiển thị tên các người dùng đăng nhập và số người dùng đăng nhập.

Ví dụ:

```
# who
root tty1 Nov 15 03:54
lan pts/0 Nov 15 06:07
#
```

Lệnh **who** hiển thị ba cột thông tin cho từng người dùng trên hệ thống. Cột đầu là tên của người dùng, cột thứ hai là tên thiết bị đầu cuối mà người dùng đó đang sử dụng, cột thứ ba hiển thị ngày giờ người dùng đăng nhập.

Ngoài **who**, có thể sử dụng thêm lệnh **users** để xác định được những người đăng nhập trên hệ thống.

Ví dụ:

```
# users
lan root
#
```

* Trong trường hợp người dùng không nhớ nổi tên đăng nhập trong một phiên làm việc (điều này nghe có vẻ như hơi vô lý nhưng là tình huống đôi lúc gặp phải), hãy sử dụng lệnh **whoami** và **who am i**.

Cú pháp lệnh:

```
whoami
```

hoặc

```
who am i
```

Ví dụ:

```
# whoami
lan
#
# who am i
may9!lan pts/0 Nov 15 06:07
#
```

Lệnh **who am i** sẽ hiện kết quả đầy đủ hơn với tên máy đăng nhập, tên người dùng đang đăng nhập, tên thiết bị và ngày giờ đăng nhập.

*** <u>Có một cách khác để xác định thông tin người dùng với lệnh *id*</u>**

Cú pháp lệnh:

```
id [tùy-chọn] [người-dùng]
```

Lệnh này sẽ đưa ra thông tin về người dùng được xác định trên dòng lệnh hoặc thông tin về người dùng hiện thời.

Các tuỳ chọn là:

- **-g, --group** : chỉ hiển thị chỉ số nhóm người dùng.
- **-u, --user** : chỉ hiển thị chỉ số của người dùng.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# id
uid=506(lan) gid=503(lan) groups=503(lan)
#
# id -g
503
#
# id -u
506
#
# id root
```

```
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),
3(sys),4(adm),6(disk),10(wheel)
#
```

5.4.3 Xác định các quá trình đang được tiến hành (lệnh w)

Lệnh **w** cho phép xác định được thông tin về các quá trình đang được thực hiện trên hệ thống và những người dùng tiến hành quá trình đó.

Cú pháp lệnh:

```
w [người-dùng]
```

Lệnh **w** đưa ra thông tin về người dùng hiện thời trên hệ thống và quá trình họ đang thực hiện. Nếu chỉ ra người dùng trong lệnh thì chỉ hiện ra các quá trình liên quan đến người dùng đó.

Ví dụ:

```
# w
root tty2 - 2:14pm 13:03 9.30s 9.10s /usr/bin/mc -P
lan pts/1 192.168.2.213 3:20pm 0.00s 0.69s 0.10s w
root pts/2 :0 3:33pm 9:32 0.41s 0.29s /usr/bin/mc –P
```

CHƯƠNG 6. TRUYỀN THÔNG VÀ MẠNG UNIX-LINUX

6.1. Lệnh truyền thông

6.1.1. Lệnh write

Lệnh write được dùng để trao đổi giữa những người hiện đang cùng làm việc trong hệ thống. Thông thường, một người dùng muốn liên hệ với người dùng khác, cần sử dụng lệnh who:

```
$who
```

hiện thông tin như sau:

```
user1     tty17          Oct 15 10:20
user2     tty43          Oct 15 8:25
user4     tty52          Oct 15 12:20
```

trong đó có tên người dùng, số hiệu terminal, ngày giờ vào hệ thống.

Sau đó sử dụng lệnh write để chuyển thông báo cho nhau.

```
$write <tên người dùng>  [<tên trạm cuối>]
```

cần gửi thông báo đến người dùng user1 có tên user2 sẽ gõ:

```
$write user2 tty43
```

- Nếu người dùng user2 hiện không làm việc thì trên màn hình người dùng user1 sẽ hiện ra: "user2 is not logged in" và hiện lại dấu mời shell.
- Nếu người dùng user2 đang làm việc, máy người dùng user2 sẽ phát ra tiếng chuông và trên màn hình hiện ra:

```
Message from user1 on tty17 at <giờ, phút>
```

Cùng lúc đó, tại máy của user1 màn hình trắng để hiện những thông tin gửi tới người dùng user2. Người gửi gõ thông báo của mình theo quy tắc:

- Kết thúc một dòng bằng cụm -o,
- Kết thúc dòng cuối cùng (hết thông báo) bằng cụm -oo.

Để kết thúc kết nối với người dùng user2, người dùng user1 gõ ctrl-d.

Để từ chối mọi việc nhận thông báo từ người khác, sử dụng lệnh không nhận thông báo:

```
$mesg n    (n - no)
```

Một người khác gửi thông báo đến người này sẽ nhận được việc truy nhập không cho phép permission denied.

Để tiếp tục cho phép người khác gửi thông báo đến, sử dụng lệnh:

```
$mesg y    (y - yes)
```

6.1.2. Lệnh mail

Lệnh mail cho phép gửi thư điện tử giữa các người dùng, song hoạt động theo chế độ off-line (gián tiếp). Khi dùng lệnh write để truyền thông cho nhau thì đòi hỏi hai người gửi và nhận đồng thời đang làm việc và cùng chấp nhận cuộc trao đổi đó. Cách thức sử dụng mail là khác hẳn: một trong hai người gửi hoặc nhận có thể không đăng nhập vào hệ thống. Để đảm bào cách thức truyền thông gián tiếp (còn gọi là off-line) như vậy, hệ thống tạo ra cho mỗi người dùng một hộp thư riêng. Khi một người dùng lệnh mail gửi thư đến một người khác thì thư được tự động cho vào hộp thư của người nhận và người nhận sau đó cũng dùng lệnh mail để xem trong hộp thư có thư mới hay không. Không những thế mail còn cho phép sử dụng trên mạng internet (địa chỉ mail thường dưới dạng ***tên-login@máy.mạng.lĩnh-vực.quốc-gia***).

- Lệnh mail chỉ yêu cầu người gửi (hoặc người nhận) login trong hệ thống. Việc nhận và gửi thư được tiến hành từ một người dùng. Thư gửi đi cho người dùng khác, được lưu tại hộp thư của hệ thống.
- Tại thời điểm login hệ thống, người dùng có thể thấy được có thư mới khi trên màn hình xuất hiện dòng thông báo "you have mail".

Lệnh mail trong UNIX gồm 2 chức năng: gửi thư và quản lý thư. Tương ứng, có hai chế độ làm việc với lệnh mail: mode lệnh (command mode) quản trị thư và mode soạn (compose mode) cho phép tạo thư.

a/ Mode soạn

Mode soạn làm việc trực tiếp với một thư và gửi ngay cho người khác. Mode soạn thực chất là sử dụng lệnh mail có tham số:

$mail tên_người_nhận> **Ví dụ, $mail user2**

Lệnh này cho phép soạn và gửi thư cho người nhận có tên được chỉ.

Sau khi gõ lệnh, màn hình bị xóa và con trỏ soạn thảo nhấp nháy ở góc trên, trái để người dùng gõ nội dung thư.

Để kết thúc soạn thư, hãy gõ ctrl-d, màn hình của mail biến mất và dấu mời của shell lại xuất hiện.

Chú ý: Dạng sau đây được dùng để gửi thư đã soạn trong nội dung một file nào đó (chú ý dấu "<" chỉ dẫn thiết bị vào chuẩn là nội dung file thay vì cho bàn phím):

$mail tên_người_nhận < tên_file_nội_dung_thư

Ví dụ, **$ mail user2 < thu1**

Nội dung thư từ File thu1 được gửi cho người nhận user2, dấu mời của shell lại hiện ra.

Cách làm trên đây hay được sử dụng trong gửi / nhận thư điện tử hoặc liên kết truyền thông vì cho phép tiết kiệm được thời gian kết nối vào hệ thống, đặc biệt chi phí phải trả khi kết nối là đáng kể.

b/ Mode lệnh

Như đã nói sử dụng mode lệnh của mail để quản lý hộp thư. Vào mail theo mode lệnh khi dùng lệnh mail không tham số:

$mail

Sau khi gõ lệnh, màn hình mail ở mode lệnh được hiện ra với dấu mời của mode lệnh. (phổ biến là dấu chấm hỏi "?") Tại đây người dùng sử dụng các lệnh của mail quản lý hệ thống thư của mình.

- Cần trợ giúp gõ dấu chấm hỏi (màn hình có hai dấu ??): ? màn hình hiện ra dạng sau:

<số>	Hiện thư số <số>
(dấu cách)	Hiện thư ngay phía trước
+	Hiện thư ngay tiếp theo
l cmd	thực hiện lệnh cmd
dq	xóa thư hiện thời và ra khỏi mail
m user	gửi thư hiện thời cho người dùng
s tên-file	ghi thư hiện thời vào file có tên
r [tên-file]	trả lời thư hiện thời (có thể từ file)
d <số>	xóa thư số
u	khôi phục thư hiện thời
u <số>	khôi phục thư số
m <user> ...	chuyển tiếp thư tới các người dùng khác
q	ra khỏi mail

- Thực hiện các lệnh theo chỉ dẫn trên đây để quản trị được hộp thư của cá nhân.

6.1.3. Lệnh talk

Trong Linux cho phép sử dụng lệnh talk thay thế cho lệnh write.

6.2 Cấu hình Card giao tiếp mạng

Để các máy có thể giao tiếp được với nhau trong mạng theo giao thức TCP/IP, thiết bị dùng làm phương tiện giao tiếp đó là Card giao tiếp mạng (network card). Để quản lý thiết bị này Linux cung cấp lệnh ifconfig. Lệnh này dùng để xem các thông tin về cấu hình mạng hiện thời của máy cũng như gán các địa chỉ cho các card giao tiếp mạng (interface). Ngoài ra ta cũng có thể dùng lệnh này để kích hoạt hoặc tắt một card mạng.

```
/sbin/ifconfig <giao diện> [ <địa chỉ> ] [ arp | -arp][
broadcast <địa chỉ>][netmask <mặt nạ mạng> ]
```

trong đó:

<giao diện>

tên của thiết bị giao tiếp mạng, chẳng hạn eth0 cho card mạng đầu tiên, eth1 cho card mạng thứ hai.

<địa chỉ>

địa chỉ mạng sẽ gán cho giao diện này.

up

tuỳ chọn này sẽ kích hoạt giao diện được chỉ ra.

down

tuỳ chọn này sẽ tắt giao diện được chỉ ra.

arp | -arp

cho phép hay cấm giao thức ARP trên giao diện này.

broadcast <địa chỉ>

xác định địa chỉ quảng bá cho giao diện này.

netmask <mặt nạ mạng>

xác định mặt nạ mạng cho giao diện này.

Để xem cấu hình của máy hiện tại ta dùng lệnh

```
# ifconfig
```

Và ví dụ về kết quả thu được là:

```
eth0 Link encap:Ethernet HWaddr 00:02:55:07:63:07
inet addr:203.113.130.201 Bcast:203.113.130.223 Mask:255.255.255.224
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3912830 errors:84463 dropped:0 overruns:0 frame:0
TX packets:2402090 errors:0 dropped:0 overruns:0 carrier:0
collisions:84463 txqueuelen:100
RX bytes:2767096664 (2638.9 Mb) TX bytes:1265930467 (1207.2 Mb)
Interrupt:29
```

eth1 Link encap:Ethernet HWaddr 00:05:1C:98:05:B1
inet addr:10.10.0.10 Bcast:10.10.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:15389731 errors:0 dropped:0 overruns:0 frame:0
TX packets:7768909 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:2578998337 (2459.5 Mb) TX bytes:1471928637 (1403.7 Mb)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:45868 errors:0 dropped:0 overruns:0 frame:0
TX packets:45868 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:5338927 (5.0 Mb) TX bytes:5338927 (5.0 Mb)

Trong trường hợp này ta thấy máy hiện tại có 2 card mạng và được gán các địa chỉ tương ứng như trên.

Muốn chỉ xem các thông tin về một card mạng nào đó thôi ta dùng lệnh:

```
# ifconfig eth0
```

Muốn kích hoạt một card mạng ta dùng lệnh

```
# ifconfig eth0 up
```

Muốn tắt một card mạng ta dùng lệnh

```
# ifconfig eth0 down
```

Muốn đặt lại địa chỉ cho một card mạng ta dùng lệnh:

```
# ifconfig eth0 203.162.9.154 netmask 255.255.255.248
```

Ngoài ra nếu máy tính có cài giao diện GNOME cùng các package quản lý mạng của GNOME thì ta có thể sử dụng lệnh có giao diện đồ hoạ giúp cho việc cấu hình các tham số card mạng dễ dàng hơn. Để có công cụ này ta phải cài đặt package redhat-config-network-xxx.rpm trong đó xxx là số hiệu phiên bản của chương trình.

Trong giao diện đồ hoạ GNOME ta đánh lệnh redhat-config-network, một hộp thoại sẽ hiện lên cho phép ta thay đổi các tham số cho từng card mạng được cài trên máy.

6.3. Các dịch vụ mạng

6.3.1 Hệ thông tin mạng NIS

Khi sử dụng hệ thống mạng nói chung, mục đích của chúng ta là làm cho môi trường mạng trở nên trong suốt đối với người dùng. Một trong những điểm quan trọng là làm cho các dữ liệu quan trọng như là thông tin về người dùng, về các trạm trong mạng là đồng nhất trên tất cả các trạm làm việc. NIS (Network Information System) là một ứng dụng cung cấp các tiện ích truy nhập cơ sở dữ liệu để phân phối thông tin, chẳng hạn như dữ liệu trong /etc/passwd và /etc/group cho tất cả các máy trạm trên mạng. Điều này làm cho mạng trở nên một hệ thống duy nhất. NIS được xây dựng trên việc sử dụng dịch vụ RPC (Remote Procedure Call). Nó bao gồm một thư viện máy chủ, thư viện máy trạm và các công cụ quản trị. Ban đầu NIS được gọi là những trang vàng (Yellow Pages –YP). Cùng với sự phát

triển của NIS mà có sự xuất hiện khác nhau trong các phiên bản. NIS truyền thống được xây dựng trên thư viện libc 4/5. NIS+ là sự mở rộng của NIS song vẫn hỗ trợ bảo mật thông tin. NYS là một phiên bản chuẩn hỗ trợ cả NIS và NIS+.

Hoạt động của NIS

NIS lưu trữ cơ sở dữ liệu về thông tin quản trị mạng trong các file maps. Các file này được đặt trên một NIS server trung tâm, từ đó các NIS client có thể truy nhập đến các thông tin thông qua dịch vụ RPC. Các file maps thường là các file theo định dạng DMB, một dạng cơ sở dữ liệu đơn giản. Các file maps được tạo ra từ các file văn bản như /etc/hosts hay /etc/passwd. Mỗi file văn bản này có thể có nhiều file maps khác nhau tùy thuộc vào khóa của nó. Ví dụ nếu khóa là tên máy trạm thì ta có file hosts.byname, nếu khóa là địa chỉ IP thì ta có file hosts.byname.

File chủ	File maps tương ứng
/etc/hosts	hosts.addr Hosts.byname
/etc/networks	network.byname network.byaddr
/etc/passwd	passwd.byname passwd.byid
/etc/groups	Groups.byname group.byid
/etc/services	service.byname service.bynumber
/etc/rpc	rpc.bynumber rpc.byname
/etc/protocol	protocol.byname protocol.bynumber
/usr/lib/aliases	mail.aliases

Mỗi một file maps có một tên ngắn hơn để đễ nhớ đối với người dùng gọi là các nickname. Để hiển thị danh sách các nickname ta dùng lệnh ***ypcat***:

```
#ypcat -x
Use "ethers" for map "ethers.byname"
Use "aliases" for map "mail.aliases"
Use "services" for map "services.byname"
Use "protocols" for map "protocols.bynumber"
Use "hosts" for map "hosts.byname"
Use "networks" for map "networks.byaddr"
....
```

Các chương trình máy chủ của NIS thường có tên là ypserv. Trong các mạng cỡ nhỏ ta chỉ cần một máy làm máy chủ NIS. Một miền (domain) NIS là một tập hợp các máy trạm được quản lý bởi một máy chủ NIS. Để hiển thị và đặt tên cho một miền ta sử dụng lệnh

```
#domainname nis-domain
```

Tên miền NIS sẽ cho biết máy chủ của miền nào các ứng dụng sẽ truy cập để nhận thông tin cần thiết. Để biết được máy chủ nào trong mạng là NIS server, các chương trình ứng dụng phải hỏi ***ypbind***, một chương trình chạy ngầm có nhiệm vụ phát hiện các NIS server trên mạng. Nó sẽ phát các gói tin quảng bá để tìm các máy chủ NIS trên mạng hoặc sử dụng các thông tin trong các file cấu hình người quản trị đã cung cấp.

Cài đặt và cấu hình cho máy chủ NIS

Với NIS ta có khái niệm máy chủ NIS chính và máy chủ NIS phụ, một miền chỉ có thể có một máy chủ NIS chính. Khi trong mạng có nhiều máy trạm làm việc, một máy chủ NIS có thể bị quá tải, hoặc khi có sự cố thì toàn bộ hệ thống mạng đó sẽ không thể hoạt động được. Các máy chủ NIS phụ sẽ giúp giải quyết vấn đề này. Việc cài đặt các máy chủ NIS phụ chỉ khác máy chủ NIS chính ở chỗ tạo ra các file map. Chúng không được tạo ra bằng *makedbm* mà được lấy về từ máy chủ chính.

Bây giờ ta tìm hiểu cách cài đặt máy chủ NIS chính. Trước tiên ta phải cài đặt phần mềm ***ypserv*** lên máy tính. Chương trình sẽ nằm trong package ypserv-xxx.rpm. Ta có thể cài đặt bằng lệnh:

```
#rpm -ivh ypserv-xxx.rpm
#mkdir /var/yp/nis-domain
```

Để tạo các file cơ sở dữ liệu ta sử dụng chương trình *makedbm*. Do đó phải đảm bảo chương trình đã được cài trên máy, việc tạo lập sẽ được tiến hành thông qua một makefile. Trong file này sẽ chứa các lệnh cần thiết để tạo ra file maps. Sau khi cài đặt phần mềm ta dùng lệnh ***make***:

```
#domainname nis-domain
#cd /var/yp
#make
```

Các file map không tự động cập nhật mỗi khi ta sửa thông tin quản trị. Do vậy mỗi khi có sự thay đổi, ta cần thực hiện lại lệnh *make* để cập nhật sự sửa đổi.

Cài đặt các máy trạm NIS

Trước tiên ta cần cài đặt phần mềm *ypbind* lên máy trạm bằng lệnh:

```
#rpm -ivh ypbind-xxx.rpm
```

Bước tiếp theo là chỉ ra tên của máy chủ và tên miền NIS mà trạm này sẽ sử dụng bằng cách thay đổi thông tin trong file */etc/yp.conf* như sau:

```
#/etc/yp.conf
domainname nis-domain
server lnserver
```

Dòng đầu tiên cho biết máy trạm này thuộc vào miền NIS có tên là ***nis-domain.*** Nếu không có dòng lệnh này thì ta có thể chỉ ra bằng cách đánh lệnh *domainname* tại dấu nhắc dòng lệnh. Dòng thứ 2 chỉ ra tên máy chủ NIS. Địa chỉ IP của tên máy chủ này phải xuất hiện trong file */etc/hosts*. Hoặc ta có thể sử dụng địa chỉ IP ngay trên dòng này.

Khi ta sử dụng máy tính thường xuyên phải thay đổi miền NIS, ta có thể chỉ ra nhiều miền NIS và các máy chủ tương ứng với nó bằng lệnh server. File cấu hình dưới đây cho phép thực hiện điều đó:

```
#yp.conf
server ln-server1 domainname1
server ln-server2 domainname2
```

Khi muốn sử dụng một miền khác thì ta chỉ cần đánh lại lệnh *domainname* để xác định miền ta tương ứng.

Sau khi đã tạo ra các file cấu hình cơ bản, ta nên kiểm tra xem chương trình ypbind đã hoạt động hay chưa. Trước hết, khởi động *ypbind,* sau đó dùng tiện ích *ypcat* để lấy thông tin quản lý bởi NIS server. Để xem thông tin về địa chỉ IP của các trạm ta dùng lệnh:

```
#ypbind
#ypcat hosts.byname
```

192.168.50.1 may1
192.168.50.1 may2
....

Nếu ta không nhận được kết quả như trên hoặc ta nhận được một thông báo lỗi "can't bind to servers domain", có nghĩa là hệ thống NIS hoạt động chưa tốt, ta có thể kiểm tra xem tên miền và tên máy chủ trong file yp.conf đã chính xác chưa và sau đó *ping* máy chủ. Nếu máy chủ đã hoạt động ta kiểm tra xem sự hoạt động của ypserv bằng lệnh rpcinfo:

```
#rpcinfo –u serverhost ypserv
```

program 10004 version 2 ready and waiting

Nếu ta nhận được thông báo như trên là ypserv đang hoạt động tốt.

Lựa chọn các file map

Khi sử dụng NIS ta cần xác định những file cấu hình nào của các máy trạm sẽ được thay thể bởi NIS. Thông thường NIS được sử dụng để tra cứu các thông tin về máy trạm và tài khoản người dùng. Mặc dù ta đã sử dụng NIS như là một hệ quản trị tập trung, hệ thống này vẫn cho phép các máy trạm làm việc được quyền tự do lựa chọn sử dụng các file cấu hình cục bộ hoặc sử dụng từ NIS server. Thứ tự được chỉ ra trong file */etc/nsswitch.conf*.

Ví dụ sau cho biết thứ tự sử dụng dịch vụ của các hàm *gethostbyname(), gethostbyaddr()* và *getservbyname()*. Các dịch vụ được liệt kê trước sẽ được sử dụng, nếu không thành công thì sử dụng dịch vụ sau đó.

```
#nsswitch.conf
hosts: nis dns files
services files nis
```

Dưới đây là danh sách các dịch vụ có thể sử dụng trong file */etc/nsswitch.conf*. Các file, chương trình cụ thể được sử dụng sẽ phụ thuộc vào từng loại dịch vụ:

nisplus hay nis+: sử dụng NIS+ server cho miền NIS hiện thời. Tên của server được chỉ ra trong file /etc/nis.conf.

nis: sử dụng NIS server cho domain hiện thời. Tên của server được chỉ ra trong file */etc/yp.conf*. Với thành phần *hosts*, các file map là *hosts.byname* và *hosts.byaddr* sẽ được sử dụng.

dns: sử dụng DNS server, dịch vụ này được sử dụng cho mình thành phần hosts. Tên của máy chủ được đặt trong file */etc/resolv.conf*.

files: sử dụng các file cấu hình cục bộ, ví dụ: /etc/passwd cho thành phần passwd.

dbm: tìm thông tin trong các file cơ sở dữ liệu /var/dbm. Tên của các file là tên của các file map tương ứng của dịch vụ NIS.

Các thành phần được hỗ trợ hiện thời của NYS là: hosts, networks, passwd, group, shadow, services, protocols, rpc, và một số file khác.

Nếu có từ khóa [NOTFOUND=return] trong các thành phần của file *nsswitch.conf,* NIS sẽ thoát ra ngay mà không sử dụng tiếp các dịch vụ sau trong trường hợp nó không tìm thấy thông tin ở dịch vụ trước đó. Chỉ khi nào dịch vụ trước bị lỗi, NIS mới dùng tiếp dịch vụ

sau. Trong ví dụ dưới NIS chỉ sử dụng các file cục bộ khi khởi động hoặc DNS, NIS server bị hỏng.

```
#/ect/nsswitch.conf
hosts: nis dns [NOTFOUND=return] files
network: nis [NOTFOUND=return] files
services: file nis
protocol: files nis
rpc: files nis
```

Sử dụng các file map passwd và group

Một trong những ứng dụng chính của NIS là đồng bộ thông tin về các tài khoản của người sẻ dụng trên tất cả các máy trạm trong miền NIS. Khi đó thông tin về người dùng trên trạm được liệt kê một phần nhỏ trong */etc/passwd*, phần còn lại được lưu trong file map *passwd.byname*. Việc chọn nis trong file */etc/nsswitch.conf* chưa đủ để NIS có thể hoạt động.

Khi sử dụng tài khoản của người dùng được cung cấp bởi NIS, trước tiên phải đảm bảo số hiệu của người dùng trong file *passwd* phải trùng với số hiệu của người dùng đó trên NIS. Nếu số hiệu người dùng, số hiệu nhóm của người dùng đó khác với thông tin trong miền NIS, ta cần sửa lại cho trùng nhau.

Trước tiên ta thay đổi số hiệu người dùng (uid) và số hiệu nhóm (gid) trong đó file */etc/passwd* và */etc/group* trên trạm cục bộ sang các giá trị mới của NIS. Sau đó đổi quyền sở hữu của tất cả các file bằng cách thay đổi số hiệu uid và gid cũ sang uid và gid mới. Giả sử người dùng anhnv có số hiệu uid là 501, thuộc nhóm sinhvien có gid là 423, ta sửa đổi quyền sở hữu như sau:

```
#find / -uid 501 – print > /tmp/uid/uid.501
#find / -gid 501 – print > /tmp/uid/gid.501
#cat /tmp/uid.501 | xargs chown anhnv
#cat /tmp/gid.501 | xargs chgrp sinhvien
```

Sau khi thực hiện các công việc trên số hiệu uid và gid của một người dùng trên máy trạm sẽ đồng nhất với NIS. Bước tiếp theo là sửa đổi file */etc/nsswitch.conf* như sau:

```
#/etc/nsswitch.conf
passwd: nis files
group: nis files
```

File trên quy định lệnh login và các lệnh thuộc họ này sẽ truy vấn NIS server khi một người dùng muốn truy nhập, nếu không tìm thấy nó sẽ tìm tiếp đến các file cục bộ. Thông thường ta loại bỏ hầu hết người dùng khỏi các file cục bộ, chỉ giữ lại root hoặc các tài khoản chung như mail, news,… cho một số tác vụ cần chuyển đổi số hiệu uid sang tên và ngược lại. Ví dụ trong chương trình quản lý công việc cron sử dụng lệnh *su* để tạm trở thành news. Nếu news không có trong */etc/passwd*, chương trình trên sẽ không thực hiện được.

Khi người dùng muốn thay đổi mật khẩu, họ không thể dùng lệnh *passwd* như khi chưa có NIS. Lệnh *passwd* chỉ có tác dụng sửa đổi các file cấu hình cục bộ. NIS cung cấp một công cụ là *yppasswd*, nó không những cho phép sửa đổi mật khẩu người dùng trên NIS mà còn thay đổi các thuộc tính khác như shell… Chương trình này được thực hiện khi khởi động hệ thống bằng cách chạy thêm dịch vụ *rpc.yppasswd*. Vì thói quen người dùng có thể

gõ lệnh *passwd* khi muốn thay đổi mật khẩu. Giải pháp ở đây là thay đổi *passwd* bằng một liên kết đến *yppasswd*.

```
#cd /bin
#mv passwd passwd.old
#ln yppasswd passwd
```

6.4 Hệ thống file trên mạng

Linux có dịch vụ chia sẻ file trên mạng máy tính. Khi ta muốn có khả năng các máy Linux có thể chia sẻ tài nguyên là các file với nhau, dịch vụ NFS sẽ cung cấp khả năng này. Dịch vụ này cho phép chia sẻ file cho các người dùng trên mạng LAN, các file này có khả năng xuất hiện đối với các người dùng như là các file ở trên máy của mình.

6.4.1 Cài đặt NFS

Để cài đặt dịch vụ này ta cần chuẩn bị một package là nfs-utils-xxx.rpm trong đó xxx là số hiệu phiên bản. Đăng nhập với quyền root và sử dụng lệnh:

```
# rpm –ivh nfs-utils-xxx.rpm
```

Nếu không có lời thông báo lỗi thì việc cài đặt đã thành công.

NFS sử dụng thủ tục RPC (Remote Procedure Calls) để gửi và nhận yêu cầu giữa các máy chủ và máy trạm trên mạng, do vậy dịch vụ ánh xạ cổng portmap (dịch vụ quản lý các yêu cầu RPC) phải được khởi động trước. Trên máy chủ NFS dự định sẽ chia sẻ các file dữ liệu phải khởi động hai dịch vụ nfs và portmap bằng lệnh:

```
# service nfs start
# service portmap start
```

Để NFS hoạt động thì ta cần phải khởi động các dịch vụ sau:

Portmapper: tiến trình này không làm việc trực tiếp với NFS mà tham gia quản lý các yêu cầu RPC từ máy trạm gửi đến.

Mountd: tiến trình này sẽ ánh xạ các file trên máy chủ tới các thư mục trên máy trạm yêu cầu. Nó sẽ huỷ bỏ ánh xạ này nếu có lệnh umount từ máy trạm.

Nfs: là tiến trình chính thực hiện các nhiệm vụ của giao thức NFS. Nó có nhiệm vụ cung cấp cho các máy trạm các thư mục hoặc file được yêu cầu.

Ta có thể kiểm tra các thông tin về các dịch vụ NFS bằng lệnh:

```
#rpcinfo -p
```

Ta sẽ thu được kết quả:

```
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
…
100005 3 udp 1024 mountd
100005 3 tcp 1024 mountd
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
...
```

6.4.2 Khởi động và dừng NFS

Việc khởi động dịch vụ NFS cũng khá đơn giản và đã được giới thiệu ở trên bằng cách khởi động portmap và nfs.

```
# service nfs start
```

hoặc

```
#/etc/init.d/nfs start
```

Việc dừng (tắt) dịch vụ này cũng khá đơn giản, ta dùng lệnh sau:

```
#service nfs stop
```

hoặc

```
#/etc/init.d/nfs stop
```

Ta có thể đặt cho dịch vụ này được tự động khởi động khi ta khởi động máy tính bằng cách dùng lệnh:

```
#setup
```

Hình Đặt các ứng dụng tự khởi động khi Linux khởi động

Sau đó chọn "System services", tiếp đó ta sẽ nhận được một danh sách các dịch vụ hiện đang có trong hệ thống. Muốn cho dịch vụ nào được tự động khởi động ta chỉ cần chọn dịch vụ đó, ở đây ta chọn dịch vụ có tên nfs. Chọn OK và cuối cùng chọn Quit.

6.4.3 Cấu hình NFS server và Client

Cấu hình nfs ta chỉ cần sửa file /etc/exports, đây là file chứa danh sách các thư mục được chia sẻ cho các máy khác. Nó cũng đồng thời chứa danh sách các máy trạm có quyền được truy cập và quyền truy cập của các máy trạm này. Một chú ý về định dạng của file này như sau: các dòng trống sẽ được bỏ qua, các dòng bắt đầu bằng dấu # được coi là các dòng

chú thích và sẽ được bỏ qua. Các dòng dài quá ta có thể ngắt trên nhiều dòng bằng cách sử dụng dấu ngắt dòng (\).

Cấu trúc của mỗi dòng khai báo trong file này như sau:

Tên thư mục	Danh sách địa chỉ các máy trạm, quyền truy nhập của các máy trạm đó	Danh sách địa chỉ các máy trạm, quyền truy nhập của các máy trạm đó
/software/project /software/setup	192.168.0.172(rw) 192.168.0.0/28(ro)	192.168.0.127(ro)

Trong đó, các tham số có ý nghĩa như sau: tên thư mục là đường dẫn đến thư mục ta muốn chia sẻ cho các máy khác. Danh sách địa chỉ các máy trạm, có thể là địa chỉ IP hoặc tên máy (được liệt kê trong file /etc/hosts). Trong trường hợp muốn liệt kê danh sách các máy có địa chỉ gần kề nhau trong một khoảng nào đó ta có thể có cách viết rút gọn như sau: chẳng hạn ta muốn liệt kê các địa chỉ của máy trạm trong khoảng từ 192.168.0.0 đến các máy trạm có địa chỉ 192.168.0.15 ta chỉ cần viết 192.168.0.0/28. Quyền truy nhập được viết dưới dạng (rw) chỉ quyền đọc và ghi, còn (ro) thì các máy trạm chỉ có quyền đọc trên thư mục đó, (noaccess) cấm các máy trạm truy nhập vào các thư mục con của thư mục chia sẻ. Chú ý, giữa địa chỉ của máy và quyền truy nhập không có dấu cách.

6.4.4 Sử dụng mount

Để đọc dữ liệu trên các thư mục đã được chia sẻ này ta có thể dùng cách sau: ta sẽ dùng lệnh mount, nhưng trong trường hợp này ta phải cần quyền quản trị (root). Cú pháp của lệnh sẽ như sau:

```
#mount <tên_máy_chủ:/tên_thư_mục_chia_sẻ>
</tên_thư_mục_cần_ánh_xạ>
```

Lưu ý, trước khi ra lệnh này ta phải tạo ra thư mục cần ánh xạ (trong trường hợp nó chưa tồn tại).

Ví dụ, để ánh xạ thư mục /software/project trên một máy chủ 192.168.0.33 vào thư mục /mnt/project trên máy hiện tại ta dùng lệnh sau:

```
#mount 192.168.0.33:/software/project /mnt/project
```

Bây giờ thư mục /mnt/project trên máy hiện tại sẽ bình đẳng như các thư mục khác trên máy. Ta có thể sao chép, đọc các file trên thư mục này.

6.4.5 Unmount

Sau khi thực hiện xong các thao tác cần thiết, ta có thể hủy bỏ ánh xạ này bằng lệnh umount như sau:

```
#umount /mnt/project
```

Sau lệnh này thì ta không còn có khả năng thao tác với thư mục trên máy chủ được nữa, nếu muốn ta lại phải ánh xạ lại.

Ngoài ra muốn xem trạng thái hoạt động của dịch vụ nfs ta có thể dùng lệnh:

```
#/etc/init.d/nfs status
```

Nó sẽ hiển thị thông tin về trạng thái hiện tại của dịch vụ này đang chạy hay đã dừng lại.

```
rpc.mountd (pid 936) is running...
nfsd (pid 948 947 946 945 944 943 942 941) is running...
rpc.rquotad (pid 931) is running...
```

6.4.6 Mount tự động qua tệp cấu hình

Bây giờ nếu ta muốn hệ thống sẽ tự động ánh xạ thư mục này khi máy khởi động để cho những người dùng không có quyền quản trị có thể dùng được thì ta có thể sử dụng cách sửa đổi nội dung của file /etc/fstab.

Cũng tương tự như lệnh mount ở trên, trong file /etc/fstab cũng có các trường giống như đã nói ở trên. Mỗi một dòng trong file này sẽ có cấu trúc như sau:

```
<tên_máy_chủ:/đường_dẫn_đến_thư_mục_chia_sẻ>
</đường_dẫn_đến_thư_mục_cục_bộ> nfs
```

tham số nfs chỉ cho hệ điều hành biết kiểu file là nfs. Ví dụ ta có thể thêm dòng

```
192.168.0.33:/software/project /mnt/project nfs
```

vào cuối file /etc/fstab.

CHƯƠNG 7. LẬP TRÌNH SHELL VÀ LẬP TRÌNH C TRÊN LINUX

7.1. Cách thức pipes và các yếu tố cơ bản lập trình trên shell

7.1.1. Cách thức pipes

Trong Linux có một số loại shell, shell ngầm định là bash. Shell cho phép người dùng chạy từng lệnh shell (thực hiện trực tiếp) hoặc dãy lệnh shell (file script) và đặc biệt hơn là theo dạng thông qua ống dẫn (pipe).

- Trong một dòng lệnh của shell có thể thực hiện một danh sách các lệnh tuần tự nhau dạng:

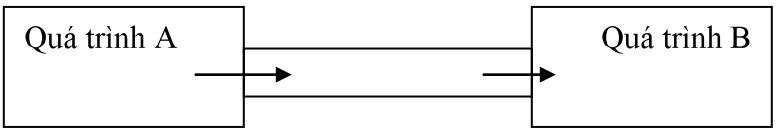
  <lệnh> [; <lệnh>]...

  Như vậy danh sách lệnh là dãy các lệnh liên tiếp nhau, cái sau cách cái trước bởi dấu chấm phảy ";"

  Ví dụ, **$ cal 10 1999; cal 11 1999 ; cal 12 1999**

  Shell cho người dùng cách thức đặc biệt thực hiện các lệnh tuần tự nhau, cái ra của lệnh trước là cái vào của lệnh sau và không phải thông qua nơi lưu trữ trung gian.

- Sử dụng ống dẫn là cách thức đặc biệt trong UNIX và Linux, được thể hiện là một cách thức của shell để truyền thông liên quá trình. ống dẫn được tổ chức theo kiểu cấu trúc dữ liệu dòng xếp hàng "vào trước ra trước" FIFO "First In First Out". Trong cấu trúc dòng xếp hàng, một đầu của dòng nhận phần tử vào và còn đầu kia lại xuất phần tử ra. Trong ngữ cảnh của shell, với hai quá trình A và B được kết nối một ống dẫn được thể hiện như sau:



Như vậy đầu ra của A thông thường hoặc là thiết bị ra chuẩn (màn hình) hoặc là một File (là một tham số của lệnh) được thay bằng "đầu nhập của ống dẫn". Tương tự, đầu vào của B thông thường hoặc là thiết bị vào chuẩn (bàn phím) hoặc là một File (là một tham số của lệnh) được thay bằng "đầu xuất của ống dẫn". Dòng byte lần lượt "chảy" từ quá trình A sang quá trình B.

Mô tả cách thức sử dụng đường ống trong shell như sau:

<lệnh phức hợp> là hoặc **<lệnh>** hoặc **(<lệnh>[;<lệnh>]...)**

Vậy đường ống có dạng

<lệnh phức hợp> | <lệnh phức hợp>

Lệnh phức hợp phía sau có thể không có đối số. Trong trường hợp đó, thông tin kết quả từ lệnh phía trước trở thành thông tin input của lệnh ngay phía sau mà không chịu tác động theo cách thông thường của lệnh trước nữa.

Ví dụ, **$ cal 1999 | more**

Nội dung lịch năm 1999 (lệnh ***cal*** đóng vai trò quá trình A) không được in ngay ra màn hình như thông thường theo tác động của lệnh ***cal*** nữa mà được lưu lên một "file" tạm thời kiểu "ống dẫn" của hệ thống và sau đó trở thành đối số của lệnh ***more*** (lệnh ***more*** đóng vai trò quá trình B).

Trong chương trình, có thể dùng ống dẫn làm file vào chuẩn cho các lệnh đọc tiếp theo. Ví dụ,

```
ls -L | \
```

thì ký hiệu "\" chỉ ra rằng ống dẫn được dùng như file vào chuẩn.

7.1.2. Các yếu tố cơ bản để lập trình trong shell

Shell có công cụ cho phép có thể lập trình trên shell làm tăng thêm độ thân thiện khi giao tiếp với người dùng. Các đối tượng tham gia công cụ như thế có thể được liệt kê:

- Các biến (trong đó chú ý tới các biến chuẩn),
- Các hàm vào - ra
- Các phép toán số học,
- Biểu thức điều kiện,
- Cấu trúc rẽ nhánh,
- Cấu trúc lặp.

a. Một số nội dung trong chương trình shell

- Chương trình là dãy các dòng lệnh shell song được đặt trong một file văn bản (được soạn thảo theo soạn thảo văn bản),
- Các dòng lệnh bắt đầu bằng dấu # chính là dòng chú thích, bị bỏ qua khi shell thực hiện chương trình,
- Thông thường các bộ dịch lệnh shell là sh (/bin/sh) hoặc ksh (/bin/ksh)

Để thực hiện một chương trình shell ta có các cách sau đây:

$sh <<tên chương trình>

hoặc **$sh <tên chương trình>**

hoặc nhờ đổi mod của chương trình:

$chmod u+x <tên chương trình>

và chạy chương trình **$<tên chương trình>**

- Phần lớn các yếu tố ngôn ngữ trong lập trình shell là tương đồng với lập trình C. Trong tài liệu này sử dụng chúng một cách tự nhiên.

b. Các biến trong file script

Trong shell có thể kể tới 3 loại biến:

- Biến môi trường (biến shell đặc biệt, biến từ khóa, biến shell xác định trước hoặc biến shell chuẩn) được liệt kê như sau (các biến này thường gồm các chữ cái hoa):
  - HOME : đường dẫn thư mục riêng của người dùng,
  - MAIL: đường dẫn thư mục chứa hộp thư người dùng,
  - PATH: thư mục dùng để tìm các file thể hiện nội dung lệnh,
  - PS1: dấu mời ban đầu của shell (ngầm định là $),
  - PS2: dấu mời thứ 2 của shell (ngầm định là >),
  - PWD: Thư mục hiện tại người dùng đang làm,
  - SHELL: Đường dẫn của shell (/bin/sh hoặc /bin/ksh)
  - TERM: Số hiệu gán cho trạm cuối,
  - USER: Tên người dùng đã vào hệ thống,

  Trong ***.profile*** ở thư mục riêng của mỗi người dùng thường có các câu lệnh dạng:

  <biến môi trường> = <giá trị>

- Biến người dùng: Các biến này do người dùng đặt tên và có các cánh thức nhận giá trị các biến người dùng từ bàn phím (lệnh read).

Biến được đặt tên gồm một xâu ký tự, quy tắc đặt tên như sau: ký tự đầu tiên phải là một chữ cỏi hoặc dấu gạch chân (_), sau tên là một hay nhiều ký tự khác. Để tạo ra một biến ta chỉ cần gán biến đó một giá trị nào đó. Phép gán là một dấu bằng (=). Ví dụ:

myname=”TriThanh”

Chú ý: không được có dấu cách (space) đằng trước hay đằng sau dấu bằng. Tên biến là phân biệt chữ hoa chữ thường. Để truy xuất đến một biến ta dùng cú pháp sau; $tên_biến. Chẳng hạn ta muốn in ra giá trị của biến myname ở trên ta chỉ cần ra lệnh: ***echo $myname.***

$myname.

$myname.

Một số ví dụ về cách đặt tên biến:

$ no=10 #đây là một cách khai báo hợp lệ

Nhưng cách khai báo dưới đây là không hợp lệ

$ no =10 #có dấu cách sau tên biến

$ no= 10 # có dấu cách sau dấu =

$ no = 10 # có dấu cách cả đằng trước lẫn đằng sau dấu =

Ta có thể khai báo một biến nhưng nó có giá trị NULL như trong những cách sau:

$ vech=

$ vech=""

Nếu ta ra lệnh in giá trị của biến này thì ta sẽ thu được một giá trị NULL ra màn hình (một dòng trống).

- Biến tự động (hay biến-chỉ đọc, tham số vị trí) là các biến do shell đã có sẵn; tên các biến này cho trước. Có 10 biến tự động:

  $0, $1, $2, ..., $9

Tham biến "$0" chứa tên của lệnh, các tham biến thực bắt đầu bằng "$1" (nếu tham số cú vị trí lớn hơn 9, ta phải sử dụng cú pháp ${} – ví dụ, ${10} để thu được các giá trị của chúng). Shell ***bash*** có ba tham biến vị trí đặc biệt, "$#", "$@", và "$#". "$#" là số lượng tham biến vị trí (không tính "$0"). "$*" là một danh sách tất cả các tham biến vị trí loại trừ "$0", đã được định dạng như là một xâu đơn với mỗi tham biến được phân cách bởi kớ tự $IFS. "$@" trả về tất cả các tham biến vị trí được đưa ra dưới dạng N xâu được bao trong dấu ngoặc kép.

Sự khác nhau giữa "$*" và "$@" là gì và tại sao lại có sự phân biệt? Sự khác nhau cho phép ta xử lý các đối số dòng lệnh bằng hai cách. Cách thứ nhất, "$*", do nó là một xâu đơn, nên có thể được biểu diễn linh hoạt hơn không cần yêu cầu nhiều mã shell. "$@" cho phép ta xử lý mỗi đối số riêng biệt bởi vì giá trị của chúng là N đối số độc lập.

Dòng ra (hay dòng vào) tương ứng với các tham số vị trí là các "từ" có trong các dòng đó. Ví dụ,

```
$chay vao chuong trinh roi
```

Nếu chay là một lệnh thì dòng vào này thì:

$0 có giá trị chay $1 có giá trị vao $2 có giá trị chuong

$3 có giá trị trinh $4 có giá trị roi

Một ví dụ khác về biến vị trí giúp ta phân biệt được sự khác nhau giữa biến $* và $@:

```
#!/bin/bash
#testparm.sh
function cntparm
{
     echo –e "inside cntparm $# parms: $*"
```

```
}
cntparm ‘$*’
cntparm ‘$@’
echo –e “outside cntparm $* parms\n”
echo –e “outside cntparm $# parms\n”
```

Khi chạy chương trình này ta sẽ thu được kết quả:

$./testparm.sh Kurt Roland Wall

inside cntparm 1 parms: Kurt Roland Wall
inside cntparm 3 parms: Kurt Roland Wall
outside cntparm: Kurt Roland Wall
outside cntparm: Kurt Roland Wall

Trong dòng thứ nhất và thứ 2 ta thấy kết quả có sự khác nhau, ở dòng thứ nhất biến “$*” trả về tham biến vị trí dưới dạng một xâu đơn, vì thế cntparm báo cáo một tham biến đơn. Dòng thứ hai gọi cntparm, trả về đối số dòng lệnh của là 3 xâu độc lập, vì thế cntparm báo cáo ba tham biến.

c. Các ký tự đặc biệt trong bash

Ký tự	Mô tả
<	Định hướng đầu vào
>	Định hướng đầu ra
(	Bắt đầu subshell
)	Kết thúc subshell
|	Ký hiệu dẫn
\	Dùng để hiện ký tự đặc biệt
&	Thi hành lệnh chạy ở chế độ ngầm
{	Bắt đầu khối lệnh
}	Kết thúc khối lệnh
~	Thư mục home của người dùng hiện tại
`	Thay thế lệnh
;	Chia cắt lệnh
#	Lời chú giải
‘	Trích dẫn mạnh
“	Trích dẫn yếu
$	Biểu thức biến
*	Ký tự đại diện cho chuỗi
?	Ký tự đại diện cho một ký tự

Các ký tự đặc biệt của bash

Dấu chia cắt lệnh, ; , cho phép thực hiện những lệnh bash phức tạp đánh trên một dòng. Nhưng quan trọng hơn, nó là kết thúc lệnh theo lý thuyết POSIX.

Ký tự chú giải, # , khiến bash bỏ qua mọi ký tự từ đó cho đến hết dòng. điểm khác nhau giữa các ký tự trích dẫn mạnh và trích dẫn yếu, ‘ và “, tương ứng là: trích dẫn mạnh bắt bash hiểu tất cả các ký tự theo nghĩa đen; trích dẫn yếu chỉ bảo hộ cho một vài ký tự đặc biệt của bash .

7.2. Một số lệnh lập trình trên shell

7.2.1. Sử dụng các toán tử bash

Các toán tử string

Các toán tử string, cũng được gọi là các toán tử thay thế trong tài liệu về bash, kiểm tra giá trị của biến là chưa gán giá trị hoặc không xác định. Bảng dưới là danh sách các toán tử này cùng với miêu tả cụ thể cho chức năng của từng toán tử.

Toán tử	Chức năng
${var:- word}	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì trả về word
${var:= word}	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì gán biến thành word, sau đó trả về giá trị của nó
${var:+ word}	Nếu biến tồn tại và xác định thì trả về word, còn không thì trả về null
${var:?message}	Nếu biến tồn tại và xác định thì trả về giá trị của nó, còn không thì hiển thị "bash: $var:$message" và thoát ra khỏi lệnh hay tập lệnh hiện thời.
${var: offset[:length]}	Trả về một xâu con của var bắt đầu tại offset của độ dài length. Nếu length bị bỏ qua, toàn bộ xâu từ offset sẽ được trả về.

Các toán tử string của bash

Để minh hoạ, hãy xem xét một biến shell có tên là status được khởi tạo với giá trị defined. Sử dụng 4 toán tử string đầu tiên cho kết quả status như sau:

$echo ${status:-undefined}
defined
$echo ${status:=undefined}
defined
$echo ${status:+undefined}
undefined
$echo ${status:?Dohhh\! undefined}
defined

Bây giờ sử dụng lệnh unset để xoá biến status, và thực hiện vẫn các lệnh đó, được output như sau:

$unset status
$echo ${status:-undefined}
undefined
$echo ${status:=undefined}
undefined
$echo ${status:+undefined}
undefined
$unset status
$echo ${status:?Dohhh\! undefined}
bash:status Dohhh! Undefined

Cần thiết unset status lần thứ hai vì ở lệnh thứ ba, echo ${status:+undefined}, khởi tạo lại status thành undefined.

Các toán tử substring đã có trong danh sách ở bảng trên đặc biệt có ích. Hãy xét biến foo có giá trị Bilbo_the_Hobbit. Biểu thức ${foo:7} trả về he_Hobbit, trong khi ${foo:7:5} lại trả về he_Ho.

Các toán tử Pattern-Matching

Các toán tử pattern-matching có ích nhất trong công việc với các bản ghi độ dài biến hay các xâu đã được định dạng tự do được định giới bởi các kí tự cố định. Biến môi trường $PATH là một ví dụ. Mặc dù nó có thể khá dài, các thư mục riêng biệt được phân định bởi dấu hai chấm. Bảng dưới là danh sách các toán tử Pattern-Matching của bash và chức năng của chúng.

Toán tử	Chức năng
${var#pattern}	Xoá bỏ phần khớp (match) ngắn nhất của pattern trước var và trả về phần còn lại
${var##pattern}	Xoá bỏ phần khớp (match) dài nhất của pattern trước var và trả về phần còn lại
${var%pattern}	Xoá bỏ phần khớp ngắn nhất của pattern ở cuối var và trả về phần còn lại
${var%%pattern}	Xoá bỏ phần khớp dài nhất của pattern ở cuối var và trả về phần còn lại
${var/pattern/string}	Thay phần khớp dài nhất của pattern trong var bằng string. Chỉ thay phần khớp đầu tiên. Toán tử này chỉ có trong bash 2.0 hay lớn hơn.
${var//pattern/string}	Thay phần khớp dài nhất của pattern trong var bằng string. Thay tất cả các phần khớp. Toán tử này có trong bash 2.0 hoặc lớn hơn.

Các toán tử bash Pattern-Matching

Thông thường quy tắc chuẩn của các toán tử bash pattern-matching là thao tác với file và tên đường dẫn. Ví dụ, giả sử ta có một tên biến shell là mylife có giá trị là /usr/src/linux/Documentation/ide.txt (tài liệu về trình điều khiển đĩa IDE của nhân). Sử dụng mẫu "/*" và "*/" ta có thể tách được tên thư mục và tên file.

```
#!/bin/bash
##############################################
myfile=/usr/src/linux/Documentation/ide.txt
echo '${myfile##*/}=' ${myfile##*/}
echo 'basename $myfile =' $(basename $myfile)
echo '${myfile%/*}=' ${myfile%/*}
echo 'dirname $myfile =' $(dirname $myfile)
```

Lệnh thứ 2 xoá xâu matching "*/" dài nhất trong tên file và trả về tên file. Lệnh thứ 4 làm khớp tất cả mọi thứ sau "/", bắt đầu từ cuối biến, bỏ tên file và trả về đường dẫn của file. Kết quả của tập lệnh này là:

$ ./pattern.sh

${myfile##/} = ide.txt*
basename $myfile   = ide.txt
${myfile%/}   = /usr/src/linux/Documentation*
dirname $myfile   = /usr/src/linux/Documentation

Để minh hoạ về các toán tử pattern-matching và thay thế, lệnh thay thế mỗi dấu hai chấm trong biến môi trường $PATH bằng một dòng mới, kết quả hiển thị đường dẫn rất dễ đọc (ví dụ này sẽ sai nếu ta không có bash phiên bản 2.0 hoặc mới hơn):

$ echo –e ${PATH//:/\\n}
/usr/local/bin
/bin
/usr/bin
/usr/X11R6/bin
/home/kwall/bin
/home/wall/wp/wpbin

Các toán tử so sánh chuỗi

kiểm tra	Điều kiện thực
str1 = str2	str1 bằng str2
str1 != str2	str1 khác str2
-n str	str có độ dài lớn hơn 0 (khác null)
-z str	str có độ dài bằng 0 (null)

Toán tử sánh chuỗi của bash

Các toán tử so sánh số học

kiểm tra	Điều kiện thực
-eq	bằng
-ge	lớn hơn hoặc bằng
-gt	lớn hơn
-le	nhỏ hơn hoặc bằng
-lt	nhỏ hơn
-ne	khác

Các cách test số nguyên của bash

7.2.2. Điều khiển luồng

Các cấu trúc điều khiển luồng của bash, nó bao gồm:

- if – Thi hành một hoặc nhiều câu lệnh nếu có điều kiện là true hoặc false.
- for – Thi hành một hoặc nhiều câu lệnh trong một số cố định lần.
- while – Thi hành một hoặc nhiều câu lệnh trong khi một điều kiện nào đó là true hoặc false.
- until – Thi hành một hoặc nhiều câu lệnh cho đến khi một điều kiện nào đó trở thành true hoặc false.
- case – Thi hành một hoặc nhiều câu lệnh phụ thuộc vào giá trị của biến.

- select – Thi hành một hoặc nhiều câu lệnh dựa trên một khoảng tuỳ chọn của người dùng.

7.2.2.1 Cấu trúc rẽ nhánh có điều kiện if

Bash cung cấp sự thực hiện có điều kiện lệnh nào đó sử dụng câu lệnh if, câu lệnh if của bash đầy đủ chức năng như của C. Cú pháp của nó được khái quát như sau:

```
if condition
then
        statements
[elif  condition
        statements]
[else
        statements]
fi
```

Đầu tiên, ta cần phải chắc chắn rằng mình hiểu if kiểm tra trạng thái thoát của câu lệnh last trong condition. Nếu nó là 0 (true), sau đó statements sẽ được thi hành, nhưng nếu nó khác 0, thì mệnh đề else sẽ được thi hành và điều khiển nhảy tới dòng đầu tiên của mã fi. Các mệnh đề elif (tuỳ chọn) (có thể nhiều tuỳ ý) sẽ chỉ thi hành khi điều kiện if là false. Tương tự, mệnh đề else (tuỳ chọn) sẽ chỉ thi hành khi tất cả else không thỏa mãn. Nhìn chung, các chương trình Linux trả về 0 nếu thành công hay hoàn toàn bình thường, và khác 0 nếu ngược lại, vì thế không có hạn chế nào cả.

Chú ý: Không phải tất cả chương trình đều tuân theo cùng một chuẩn cho giá trị trả về, vì thế cần kiểm tra tài liệu về các chương trình ta kiểm tra mã thoát với điều kiện if. Ví dụ chương trình diff, trả về 0 nếu không có gì khác nhau, 1 nếu có sự khác biệt và 2 nếu có vấn đề nào đó. Nếu một câu điều kiện hoạt động không như mong đợi thì hãy kiểm tra tài liệu về mã thoát .

Không quan tâm đến cách mà chương trình xác định mã thoát của chúng, bash lấy 0 có nghĩa là true hoặc bình thường còn khác 0 là false. Nếu ta cần cụ thể để kiểm tra một mã thoát của lệnh, sử dụng toán tử ***$?*** ngay sau khi chạy lệnh. ***$?*** trả về mã thoát của lệnh chạy ngay lúc đó.

Phức tạp hơn, bash cho phép ta phối hợp các mã thoát trong phần điều kiện sử dụng các toán tử && và || được gọi là toán tử logic AND và OR. Cú pháp đầy đủ cho toán tử AND như sau:

command1 && command2

Câu lệnh ***command2*** chỉ được chạy khi và chỉ khi ***command1*** trả về trạng thái là số 0 (true).

Cú pháp cho toán tử OR thì như sau:

command1 || command2

Câu lệnh ***command2*** chỉ được chạy khi và chỉ khi ***command1*** trả lại một giá trị khác 0 (false).

Ta có thể kết hợp lại cả 2 loại toán tử lại để có một biểu thức như sau:

command1 && comamnd2 || command3

Nếu câu lệnh ***command1*** chạy thành công thì shell sẽ chạy lệnh ***command2*** và nếu ***command1*** không chạy thành công thì ***command3*** được chạy.

Ví dụ:

$ rm myf && echo "File is removed successfully" || echo "File is not removed"

Nếu file myf được xóa thành công (giá trị trả về của lệnh là 0) thì lệnh "*echo File is removed successfully*" sẽ được thực hiện, nếu không thì lệnh "*echo File is not removed*" được chạy.

Giả sử trước khi ta vào trong một khối mã, ta phải thay đổi một thư mục và copy một file. Có một cách để thực hiện điều này là sử dụng các toán tử ***if*** lồng nhau, như là đoạn mã sau:

```
if cd /home/kwall/data
then
        if cp datafile datafile.bak
        then
                # more code here
        fi
fi
```

Tuy nhiên, bash cho phép ta viết đoạn mã này súc tích hơn nhiều như sau:

```
if cd /home/kwall/data && cp datafile datafile.bak
then
        # more code here
fi
```

Cả hai đoạn mã đều thực hiện cùng một chức năng, nhưng đoạn thứ hai ngắn hơn nhiều, gọn nhẹ và đơn giản. Mặc dù ***if*** chỉ kiểm tra các mã thoát, ta có thể sử dụng cấu trúc ***[…]*** lệnh ***test*** để kiểm tra các điều kiện phức tạp hơn. ***[condition]*** trả về giá trị biểu thị condition là true hay false. ***test*** cũng có tác dụng tương tự.

Một ví dụ khác về cách sử dụng cấu trúc ***if:***

```
#!/bin/sh
# Script to test if..elif...else
#
if [ $1 -gt 0 ]; then
  echo "$1 is positive"
elif [ $1 -lt 0 ]
then
  echo "$1 is negative"
elif [ $1 -eq 0 ]
then
  echo "$1 is zero"
else
  echo "Opps! $1 is not number, give number"
fi
```

Số lượng các phép toán điều kiện của biến hiện tại khoảng 35, khá nhiều và hoàn chỉnh. Ta có thể kiểm tra các thuộc tính file, so sánh các xâu và các biểu thức số học.

Chú ý: Các khoảng trống trước dấu mở ngoặc và sau dấu đóng ngoặc trong ***[condition]*** là cần phải có. Đây là điều kiện cần thiết trong cú pháp shell của bash.

Bảng dưới là danh sách các toán tử ***test*** file phổ biến nhất (danh sách hoàn chỉnh có thể tìm thấy trong những trang manual đầy đủ về bash).

Toán tử	Điều kiện true
-d file	file tồn tại và là một thư mục
-e file	file tồn tại
-f file	file tồn tại và là một file bình thường(không là một thư mục hay một file đặc biệt)
-r file	file cho phép đọc
-s file	file tồn tại và khác rỗng
-w file	file cho phép ghi
-x file	file khả thi hoặc nếu file là một thư mục thì cho phép tìm kiếm trên file
-O file	file của người dùng hiện tại
-G file	file thuộc một trong các nhóm người dùng hiện tại là thành viên
file1 -nt file2	file1 mới hơn file2
file1 -ot file2	file1 cũ hơn file2

Các toán tử test file của bash

Ví dụ chng trình shell cho các toán tử ***test*** file trên các thư mục trong biến $PATH. Mã cho chương trình descpath.sh như sau:

```
#!/bin/bash
##################################

IFS=:
for dir in $PATH;
do
        echo $dir
        if [ -w $dir ]; then
                echo -e "\tYou have write permission in $dir"
        else
                echo –e “\tYou don’t have write permission in $dir”
        fi
        if [ -0 $dir ]; then
                echo -e "\tYou own $dir"
        else
                echo –e “\tYou don’t own $dir”
        fi
        if [ -G $dir ]; then
                echo -e "\tYou are a member of $dir's group"
        else
                echo -e "\tYou aren't a member of $dir's group"
        fi
done
```

Chương trình descpath.sh

Vòng lặp for (giới thiệu trong phần dưới) sẽ duyệt toàn bộ các đường dẫn thư mục trong biến PATH sau đó kiểm tra các thuộc tính của thư mục đó. Kết quả như sau (kết quả có thể khác nhau trên các máy khác nhau do giá trị của biến PATH khác nhau):

/usr/local/bin
You don't have write permission in /usr/local/bin
You don't own /usr/local/bin
You aren't a member of /usr/local/bin's group
/bin
You don't have write permission in /bin
You don't own /bin
You aren't a member of /bin's group
/usr/bin
You don't have write permission in /usr/bin
You don't own /usr/bin
You aren't a member of /usr/bin's group
/usr/X11R6/bin
You don't have write permission in /usr/X11R6/bin
You don't own /usr/X11R6/bin
You aren't a member of /usr/X11R6/bin's group
/home/kwall/bin
You have write permission in /home/kwall/bin
You own /home/kwall/bin
You are a member of /home/kwall/bin's group
/home/kwall/wp/wpbin
You have write permission in /home/kwall/wp/wpbin
You own /home/kwall/wp/wpbin
You are a member of /home/kwall/wp/wpbin's group

Các biểu thức trong phần điều kiện cũng có thể kết hợp với nhau tạo thành các biểu thức phức tạp hơn bằng các phép toán logic. Dưới đây là một bảng các biểu thức logic trong shell.

Toán tử	Ý nghĩa
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR

7.2.2.2 Các vòng lặp đã quyết định: for

Như đã thấy ở chương trình trên, ***for*** cho phép ta chạy một đoạn mã một số lần nhất định. Tuy nhiên cấu trúc ***for*** của bash chỉ cho phép ta lặp đi lặp lại trong danh sách các giá trị nhất định bởi vì nó không tự động tăng hay giảm con đếm vòng lặp như là C, Pascal, hay Basic. Tuy nhiên vòng lặp ***for*** là công cụ lặp thường xuyên được sử dụng bởi vì nó điều

khiển gọn gàng trên các danh sách, như là các tham số dòng lệnh và các danh sách các file trong thư mục. Cú pháp đầy đủ của for là:

for value in list
do
statements using $value
done

list là một danh sách các giá trị, ví dụ như là tên file. Giá trị là một thành viên danh sách đơn và ***statements*** là các lệnh sử dụng value. Một cú pháp khác của lệnh ***for*** có dạng như sau:

```
for (( expr1; expr2; expr3 ))
do
      .....
                  ...
      repeat all statements between do and
      done until expr2 is TRUE
done
```

Linux không có tiện ích để đổi tên hay copy các nhóm của file. Trong MS-DOS nếu ta có 17 file có phần mở rộng a*.doc, ta có thể sử dụng lệnh COPY để copy *.doc thành file *.txt. Lệnh DOS như sau:

C:\ cp doc*.doc doc*.txt

sử dụng vòng lặp for của bash để bù đắp những thiếu sót này. Đoạn mã dưới đây có thể được chuyển thành chương trình shell thực hiện đúng như những gì ta muốn:

for docfile in doc/*.doc
do
cp $docfile ${docfile%.doc}.txt
done

Sử dụng một trong các toán tử pattern-matching của bash, đoạn mã này làm việc copy các file có phần mở rộng là *.doc bằng cách thay thế .doc ở cuối của tên file bằng .txt.

Một ví dụ khác về vòng for đơn giản như sau:

```
#!/bin/bash
for i in 1 2 3 4 5
do
      echo "Welcome $i times"
done
```

Ta cũng có một cấu trúc về ***for*** như sau, chương trình này cũng có cùng chức năng như chương trình trên nhưng ta chú ý đến sự khác biệt về cú pháp của lệnh ***for***.

```
#!/bin/bash
for ((´ i = 0 ;´ i <= 5;´ i++´ ))
do
´ echo "Welcome $i times"
done
```

$ chmod +x for2
$ ./for2
Welcome 0 times

Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times

Tiếp theo là một ví dụ về vòng ***for*** lồng nhau:

```
#!/bin/bash

for (( i = 1; i <= 5; i++ ))      ### Outer for loop ###
do

    for (( j = 1 ; j <= 5; j++ )) ### Inner for loop ###
    do
          echo -n "$i "
done
```

Ví dụ khác về cách sử dụng cấu trúc ***if*** và ***for*** như sau:

```
#!/bin/sh
#Script to test for loop
#
#
if [ $# -eq 0 ]
then
echo "Error - Number missing form command line argument"
echo "Syntax : $0 number"
echo "Use to print multiplication table for given number"
exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
done
```

Khi ta chạy chương trình với tham số:

$ chmod 755 mtable
$ ./mtable 7
Ta thu được kết quả như sau:

```
7 * 1 = 7
7 * 2 = 14
...
..
7 * 10 = 70
```

7.2.2.3 Các vòng lặp không xác định: while và until

Vòng lặp ***for*** giới hạn số lần mà một đoạn mã được thi hành, các cấu trúc ***while*** và ***until*** của bash cho phép một đoạn mã được thi hành liên tục cho đến khi một điều kiện nào đó xảy ra. Chỉ với chú ý là đoạn mã này cần viết sao cho điều kiện cuối phải xảy ra nếu không sẽ tạo ra một vòng lặp vô tận. Cú pháp của nó như sau:

while condition
do
    statements
done

Cú pháp này có nghĩa là khi nào condition còn true, thì thực hiện statements cho đến khi condition trở thành false (cho đến khi một chương trình hay một lệnh trả về khác 0):

until condition
do
    statements
done

Cú pháp ***until*** có nghĩa là trái ngược với ***while***: cho đến khi condition trở thành true thì thi hành statements (có nghĩa là cho đến khi một lệnh hay chương trình trả về mã thoát khác 0)

Cấu trúc ***while*** của bash khắc phục thiếu sót không thể tự động tăng, giảm con đếm cua vòng lặp for. Ví dụ, ta muốn copy 150 bản của một file, thì vòng lặp while là một lựa chọn để giải quyết bài toán này. Dưới đây là chương trình:

```
#!/bin/sh
#
declare  -i  idx
idx=1
while [ $idx != 150]
do
        cp somefile  somefile.$idx
        idx=$idx+1
done
```

Chương trình này giới thiệu cách sử dụng tính toán số nguyên của bash. Câu lệnh ***declare*** khởi tạo một biến, idx, định nghĩa là một số nguyên. Mỗi lần lặp idx tăng lên, nó sẽ được kiểm tra để thoát khỏi vòng lặp. Vòng lặp until tuy cũng có khả năng giống while nhưng không được dùng nhiều vì rất khó viết và chạy chậm.

Một ví dụ nữa về cách sử dụng vòng lặp while được minh họa trong chương trình in bản nhân của một số:

```
#!/bin/sh
#Script to test while statement
#
#
if [ $# -eq 0 ]
then
   echo "Error - Number missing form command line argument"
   echo "Syntax : $0 number"
   echo " Use to print multiplication table for given number"
exit 1
fi
```

```
n=$1
i=1
while [ $i -le 10 ]
do
  echo "$n * $i = `expr $i \* $n`"
  i=`expr $i + 1`
done
```

7.2.2.4 Các cấu trúc lựa chọn: case và select

Cấu trúc điều khiển luồng tiếp theo là ***case***, hoạt động cũng tương tự như lệnh switch của C. Nó cho phép ta thực hiện các khối lệnh phụ thuộc vào giá trị của biến. Cú pháp đầy đủ của ***case*** như sau:

```
case expr in
        pattern1 )
        statements ;;
        pattern2 )
        statements ;;
        …
[*)
        statements ;;]
esac
```

expr được đem đi so sánh với từng pattern, nếu nó bằng nhau thì các lệnh tương ứng sẽ được thi hành. Dấu ;; là tương đương với lệnh break của C, tạo ra điều khiển nhảy tới dòng đầu tiên của mã ***esac***. Không như từ khoá ***switch*** của C, lệnh case của bash cho phép ta kiểm tra giá trị của ***expr*** dựa vào pattern, nó có thể chứa các kí tự đại diện. Cách làm việc của cấu trúc case như sau: nó sẽ khớp (match) biểu thức expr với các mẫu pattern1, pattern2,…nếu có một mẫu nào đó khớp thì khối lệnh tương ứng với mẫu đó sẽ được thực thi, sau đó nó thoát ra khỏi lệnh case. Nếu tất cả các mẫu đều không khớp và ta có sử dụng mẫu * (trong nhánh ****)***), ta thấy đây là mẫu có thể khớp với bất kỳ giá trị nào (ký tự đại diện là *), nên các lệnh trong nhánh này sẽ được thực hiện.

Cấu trúc điều khiển ***select*** (không có trong các phiên bản bash nhỏ hơn 1.14) chỉ riêng có trong Korn và các shell bash. Thêm vào đó, nó không có sự tương tự như trong các ngôn ngữ lập trình quy ước. ***select*** cho phép ta dễ dàng trong việc xây dựng các menu đơn giản và đáp ứng các chọn lựa của người dùng. Cú pháp của nó như sau:

```
select  value [in list]
do
        statements that manipulate $value
done
```

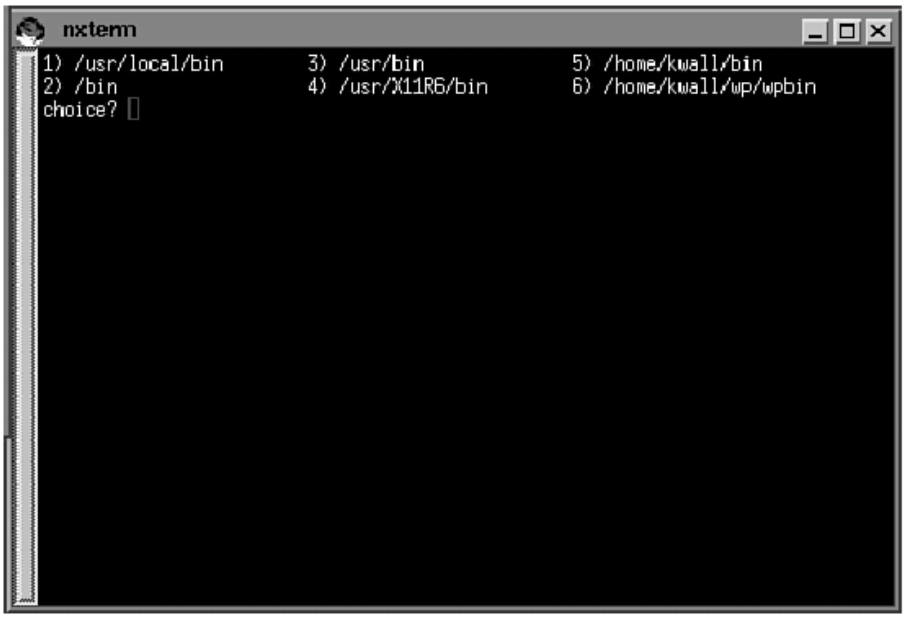
Dưới đây là một ví dự về cách sử dụng lệnh select:

```
#!/bin/bash
# menu.sh – Createing simple menus with select
##########################################

IFS=:
PS3="choice? "
```

```
# clear the screen
clear

select dir in $PATH
do
    if [ $dir ]; then
            cnt=$(ls –Al $dir | wc -l)
            echo “$cnt files in $dir”
    else
            echo “Dohhh! No such choice!”
    fi
    echo –e “\nPress ENTER to continue, CTRL –C to quit”
    read
    clear
done
```

Chương trình tạo các menu bằng select

Lệnh đầu tiên đặt kí tự IFS là : (ký tự phân cách), vì thế select có thể phân tích hoàn chỉnh biến môi trường $PATH. Sau đó nó thay đổi lời nhắc default khi ***select*** bằng biến PS3. Sau khi xoá sạch màn hình, nó bước vào một vòng lặp, đưa ra một danh sách các thư mục nằm trong $PATH và nhắc người dùng chọn lựa như là minh hoạ trong hình dưới.



Nếu người dùng chọn hợp lệ, lệnh ***ls*** được thực hiện kết quả được gửi cho lệnh đếm từ ***wc*** để đếm số file trong thư mục và hiển thị kết quả có bao nhiêu file trong thư mục đó. Do ***ls*** có thể sử dụng mà không cần đối số, script đầu tiên cần chắc chắn là $dir khác null (nếu nó là null, ls sẽ hoạt động trên thư mục hiện hành nếu người dùng chọn 1 menu không hợp lệ). Nếu người dùng chọn không hợp lệ, một thông báo lỗi sẽ được hiển thị. Câu lệnh ***read*** (được giới thiệu sau) cho phép người dùng đánh vào lựa chọn của mình và nhấn Enter để lặp lại vòng lặp hay nhấn Ctrl + C để thoát.

Chú ý: Như đã giới thiệu, các vòng lặp script không kết thuc nếu ta không nhấn Ctrl+C. Tuy nhiên ta có thể sử dụng lệnh break để thoát ra.

7.2.2.5 Các hàm shell

Các hàm chức năng của bash là một cách mở rộng các tiện ích sẵn có trong shell, nó có các điểm lợi sau:

- Thi hành nhanh hơn do các hàm shell luôn thường trực trong bộ nhớ.
- Cho phép việc lập trình trở nên dễ dàng hơn vì ta có thể tổ chức chương trình thành các module.

Ta có thể định nghĩa các hàm shell sử dụng theo hai cách:

```
function        fname
{
        commands
}
hoặc là
fname()
{
        commands

}
```

Cả hai dạng đều được chập nhận và không có gì khác giữa chúng. Để gọi một hàm đã định nghĩa đơn giản là gọi tên hàm cùng với các đối số mà nó cần.

Nếu so sánh với C hay Pascal, hàm của bash không được chặt chẽ, nó không kiểm tra lỗi và không có phương thức trả về đối số bằng giá trị. Tuy nhiên giống như C và Pascal, các biến địa phương có thể khai báo cục bộ đối với hàm, do đó tránh được sự xung đột với biến toàn cục. Để thực hiện điều này ta dùng từ khoá local như trong đoạn mã sau:

```
function        foo
{
        local  myvar
        local  yourvar=1
}
```

Trong ví dụ về các biến vị trí ở trên ta cũng thấy được cách sử dụng hàm trong bash. Các hàm shell giúp mã của ta dễ hiểu và dễ bảo dưỡng. Sử dụng các hàm và các chú thích ta sẽ đỡ rất nhiều công sức khi ta phải trở lại nâng cấp đoạn mã mà ta đã viết từ thời gian rất lâu trước đó.

7.2.3 Các toán tử định hướng vào ra

Ta đã được biết về các toán tử định hướng vào ra, > và <. Toán tử định hướng ra cho phép ta gửi kết quả ra của một lệnh vào một file. Ví dụ như lệnh sau:

```
$ cat  $HOME/.bash_profile > out
```

Nó sẽ tạo một file tên là out trong thư mục hiện tại chứa các nội dung của file bash_profile, bằng cách định hướng đầu ra của cat tới file đó.

Tương tự, ta có thể cung cấp đầu vào là một lênh từ một file hoặc là lệnh sử dụng toán tử đầu vào, <. Tacó thể viết lại lệnh cat để sử dụng toán tử định hướng đầu vào như sau:

$ cat < $HOME/.bash_profile > out

Kết quả của lệnh này vẫn như thế nhưng nó cho ta hiểu thêm về cách sử dụng định hướng đầu vào đầu ra.

Toán tử định hướng đầu ra, >, sẽ ghi đè lên bất cứ file nào đang tồn tại. Đôi khi điều này là không mong muốn, vì thế bash cung cấp toán tử nối thêm dữ liệu, >>, cho phép nối thêm dữ liệu vào cuôi file. Hay xem lệnh thêm bí danh cdlpu vào cuối của file .bashrc của tôi:

$echo "alias cdlpu='cd $HOME/kwall/projects/lpu' " >> $HOME/.bashrc

Một cách sử dụng định hướng đầu vào là đầu vào chuẩn (bàn phím). Cú pháp của lệnh này như sau:

Command << label
Input …
Label

Cú pháp này nói lên rằng ***command*** đọc các input cho đến khi nó gặp label. Dưới đây là ví dụ về cách sử dụng cấu trúc này:

```
#!/bin/bash
#####################################

USER=anonymous
PASS=kwall@xmission.com

ftp -i   -n  <<  END
open  ftp.caldera.com
user $USER $PASS
cd  /pub
ls
close
END
```

7.2.4. Hiện dòng văn bản

Lệnh **echo** hiện ra dòng văn bản được ghi ngay trong dòng lệnh có cú pháp:

`echo [tùy chọn] [xâu ký tự]`…

với các tùy chọn như sau:

- **-n** : hiện xâu ký tự và dấu nhắc trên cùng một dòng.
- **-e** : bật khả năng thông dịch được các ký tự điều khiển.
- **-E** : tắt khả năng thông dịch được các ký tự điều khiển.
- **--help** : hiện hỗ trợ và thoát. Một số bản Linux không hỗ trợ tham số này.

Ví dụ, dùng lệnh **echo** với tham số **-e**

 # echo -e 'thử dùng lệnh echo \n'

sẽ thấy hiện ra chính dòng văn bản ở lệnh:

thử dùng lệnh echo
#

ở đây ký tự điều khiển '\n' là ký tự xuống dòng.

7.2.5. Lệnh read độc dữ liệu cho biến người dùng

Lệnh read có dạng read <tên biến>

Ví dụ chương trình shell có tên thu1.arg có nội dung như sau:

```
#!/bin/sh
# Chuong trinh hoi ten nguoi va hien lai
echo "Ten anh la gi?"
read name
echo "Xin chao, $name , anh go $# doi so"
echo "$*"
```

Sau đó, ta thực hiện

```
$chmod u+x thu1.arg
```

và `$thu1.arg Hoi ten nguoi va hien lai`

sẽ thấy xuất hiện

```
Ten anh la gi? Tran Van An
Xin chao, Tran Van An, anh go 6 doi so
Hoi ten nguoi va hien lai
```

7.2.6. Lệnh set

Để gán kết quả đư ra từ lệnh shell ra các biến tự động, ta dùng lệnh set

Dạng lệnh set set `<lệnh>`

Sau lệnh này, kết quả thực hiện lệnh không hiệ ra lê màn hình mà gán kết quả đó tương ứng cho các biến tự động. Một cách tự động các từ trong kết quả thực hiện lệnh sẽ gán tương ứng cho các biến tự động (từ $1 trở đi).

Xem xét một ví dụ sau đây (chương trình thu2.arg) có nội dung:

```
#!/bin/sh
# Hien thoi diem chay chuong trinh nay
set `date`
echo "Thoi gian: $4 $5"
echo "Thu: $1"
echo "Ngay $3 thang $2 nam $6"
```

Sau khi đổi mode của File chương trình này và chạy, chúng ta nhận được:

```
Thoi gian: 7:20:15 EST
Thu: Tue
Ngay 20 thang Oct nam 1998
```

Như vậy,

$# = 6

$* = Tue Oct 20 7:20:15 EST 1998

$1 = Tue $2=Oct $3 = 20 $4 = 7:20:15

$5 = EST $6 = 1998

7.2.7. Tính toán trên các biến

Các tính toán trong shell được thực hiện với các đối số nguyên. Các phép toán gồm có: cộng (+), trừ (-), nhân (*), chia (/), mod (%).

Biểu thức thực hiện theo các phép toán đã nêu.

Tính toán trên shell có dạng:

`expr <biểu thức>`

Ví dụ, chương trình với tên ***cong.shl*** sau đây:

```
#!/bin/sh
# Tinh va in hai so
        tong = `expr $1 + $2`
        echo "Tong = $tong"
```

Sau đó, khi đổi mod và chạy

```
$cong.shl 5 6
```

sẽ hiện ra:

```
Tong = 11
```

7.2.8. Chương trình ví dụ

```
/* Program 5 */
#!/bin/sh
# Chuong trinh liet ke cac thu muc con cua 1 thu muc
# Minh hoa cach su dung if then fi, while do done
# va cac CT test, expr
if test $# -ne 1
then
        echo Cu phap: $0 \<Ten thu muc\>
        exit 1
fi

cd $1                   # Chuyen vao thu muc can list
if test $? -ne 0        # Neu thu muc khong ton tai thi ra khoi CT
then
        exit 1
fi

ls -lL |\
                        # Liet ke ca cac thong tin cua symbolic link
                        # Su dung sub-shell de tu giai phong bien
{
sum=0
# Lenh read x y de bo di dong 'total 1234..' cua lenh ls -lL
read x y ; while read mode link user group size month day hour name
do
        if [ -d $name ]
        then
                echo $name $size \($mode\)
```

```
        fi
done
        }
```

7.3. Lập trình C trên UNIX

7.3.1. Trình biên dịch gcc

Hệ điều hành UNIX luôn kèm theo bộ dịch ngôn ngữ lập trình C với tên gọi là cc (C compiler). Trong Linux, bộ dịch có tên là ***gcc*** (GNU C Compiler) với ngôn ngữ lập trình không khác nhiều với C chuẩn. Nội dung chi tiết về các ngôn ngữ lập trình trên Linux thuộc phạm vi của các tài liệu khác.

gcc cho người lập trình kiểm tra trình biên dịch. Quá trình biên dịch bao gồm bốn giai đoạn:

Tiền xử lý
Biên dịch
Tạp hợp
Liên kết

Ta có thể dừng quá trình sau một trong những giai đoạn để kiểm tra kết quả biên dịch tại giai đoạn ấy. ***gcc*** cũng có thể chấp nhận ngôn ngữ khác của C, như ANSI C hay C truyền thống. Như đã nói ở trên, ***gcc*** thích hợp biên dịch C++ hay Objective-C. Ta có thể kiểm soát lượng cũng như kiểu thông tin cần debug, tất nhiên là có thể nhúng trong quá trình nhị phân hóa kết quả và giống như hầu hết các trình biên dịch, gcc cũng thực hiện tối ưu hóa mã.

Trước khi bắt đầu đi sâu vào nghiên cứu **gcc**, ta xem một ví dụ sau:

```
#include<stdio.h>
int main (void)
{
        fprintf( stdout, “Hello, Linux programming world!\n”);
        return  0;
 }
```

*Một chương trình điển hình dùng để minh hoạ việc sử dụng **gcc***

Để biên dịch và chạy chương trình này hãy gõ:

$ gcc hello.c –o hello

$ ./hello

Hello, Linux programming world!

Dòng lệnh đầu tiên chỉ cho **gcc** phải biên dịch và liên kết file nguồn **hello.c**, tạo ra tập tin thực thi, bằng cách chỉ định sử dụng đối số **-o hello**. Dòng lệnh thứ hai thực hiện chương trình, và kết quả cho ra trên dòng thứ 3.

Có nhiều chỗ mà ta không nhìn thấy được, **gcc** trước khi chạy **hello.c** thông qua bộ tiền xử lý của **cpp**, để mở rộng bất kỳ một **macro** nào và chèn thêm vào nội dung của những file #include. Tiếp đến, nó biên dịch mã nguồn tiền xử lý sang mã **obj** . Cuối cùng, trình liên kết, tạo ra mã nhị phân cho chương trình **hello**.

Ta có thể tạo lại từng bước này bằng tay, chia thành từng bước qua tiến trình biên dịch. Để chỉ cho **gcc** biết phải dừng việc biên dịch sau khi tiền xử lý, ta sử dụng tuỳ chọn **–E** của **gcc**:

$ gcc –E hello.c –o hello.cpp

Xem xét **hello.cpp** và ta có thể thấy nội dung của **stdio.h** được chèn vào file, cùng với những mã thông báo tiền xử lý khác. Bước tiếp theo là biên dịch **hello.cpp** sang mã **obj**. Sử dụng tuỳ chọn **–c** của **gcc** để hoàn thành:

$ gcc –x cpp-output -c hello.cpp –o hello.o

Trong trường hợp này, ta không cần chỉ định tên của file output bởi vì trình biên dịch tạo một tên file **obj** bằng cách thay thế **.c** bởi **.o**. Tuỳ chọn –x chỉ cho **gcc** biết bắt đầu biên dịch ở bước được chỉ báo trong trường hợp này với mã nguồn tiền xử lý.

Làm thế nào **gcc** biết chia loại đặc biệt của file? Nó dựa vào đuôi mở rộng của file ở trên để xác định rõ phải xử lý file như thế nào cho dúng. Hầu hết những đuôi mở rộng thông thường và chú thích của chúng được liệt kê trong bảng dưới.

Phần mở rộng	Kiểu
.c	Mã nguồn ngôn ngữ C
.c, .cpp	Mã nguồn ngôn ngữ C++
.i	Mã nguồn C tiền xử lý
.ii	Mã nguồn C++ tiền xử lý
.S, .s	Mã nguồn Hợp ngữ
.o	Mã đối tượng biên dịch (obj)
.a, .so	Mã thư viện biên dịch

Các phần mở rộng của tên file đối với ***gcc***

Liên kết file đối tượng, và cuối cùng tạo ra mã nhị phân:

$ gcc hello.o –o hello

Trong trường hợp , ta chỉ muốn tạo ra các file **obj,** và như vậy thì bước liên kết là không cần thiết.

Hầu hết các chương trình C chứa nhiều file nguồn thì mỗi file nguồn đó đều phải được biên dịch sang mã **obj** trước khi tới bước liên kết cuối cùng. Giả sử có một ví dụ, ta đang làm việc trên **killerapp.c** là chương trình sử dụng phần mã của **helper.c**, như vậy để biên dịch **killerapp.c** ta phải dùng dòng lệnh sau:

$ gcc killerapp.c helper.c –o killerapp

gcc qua lần lượt các bước tiền xử lý - biên dịch – liên kết, lúc này tạo ra các file **obj** cho mỗi file nguồn trước khi tạo ra mã nhị phân cho **killerapp**.

Một số tuỳ chọn dòng lệnh của **gcc**:

-o FILE	Chỉ định tên file output; không cần thiết khi biên dịch sang mã obj. Nếu FILE không được chỉ rõ thì tên mặc định sẽ là a.out.
-c	Biên dịch không liên kết.

-DF00=BAR	Định nghĩa macro tiền xử lý đặt tên F00 với một giá trị của BAR trên dòng lệnh.
-IDIRNAME	Trước khi chưa quyết định được DIRNAME hãy tìm kiếm những file include trong danh sách các thư mục(tìm trong danh sách các đường dẫn thư mục)
-LDIRNAME	Trước khi chưa quyết định được DIRNAME hãy tìm kiếm những file thư viện trong danh sách các thư mục. Với mặc định gcc liên kết dựa trên những thư viện dùng chung
-static	Liên kết dựa trên những thư viện tĩnh
-lF00	Liên kết dựa trên libF00
-g	Bao gồm chuẩn gỡ rối thông tin mã nhị phân
-ggdb	Bao gồm tất cả thông tin mã nhị phân mà chỉ có chương trình gỡ rối GNU- gdb mới có thể hiểu được
-O	Tối ưu hoá mã biên dịch
-ON	Chỉ định một mức tối ưu hoá mã N, 0<=N<=3.
-ANSI	Hỗ trợ chuẩn ANSI/ISO của C, loại bỏ những mở rộng của GNU mà xung đột với chuẩn(tuỳ chọn này không bảo đảm mã theo ANSI).
-pedantic	Cho ra tất cả những cảnh báo quy định bởi chuẩn
-pedantic-erors	Thông báo ra tất cả các lỗi quy định bởi chuẩn ANSI/ISO của C.
-traditional	Hỗ trợ cho cú pháp ngôn ngữ C của Kernighan và Ritchie (giống như cú pháp định nghĩa hàm kiểu cũ).
-w	Chặn tất cả thông điệp cảnh báo.
-Wall	Thông báo ra tất cả những cảnh báo hữu ích thông thường mà gcc có thể cung cấp.
-werror	Chuyển đổi tất cả những cảnh báo sang lỗi mà sẽ làm ngưng tiến trình biên dịch.
-MM	Cho ra một danh sách sự phụ thuộc tương thích được tạo.
-v	Hiện ra tất cả các lệnh đã sử dụng trong mỗi bước của tiến trình biên dịch.

7.3.2. Công cụ GNU make

Trong trường hợp ta viết một chương trình rất lớn được cấu thành bởi từ nhiều file, việc biên dịch sẽ rất phức tạp vì phải viết các dòng lệnh gcc rất là dài. Để khắc phục tình trạng này, công cụ GNU make đã được đưa ra. GNU make được giải quyết bằng cách chứa tất cả các dòng lệnh phức tạp đó trong một file gọi là makefile. Nó cũng làm tối ưu hóa quá trình dịch bằng cách phát hiện ra những file nào có thay đổi thì nó mới dịch lại, còn file nào không bị thay đổi thì nó sẽ không làm gì cả, vì vậy thời gian dịch sẽ được rút ngắn.

Một makefile là một cơ sở dữ liệu văn bản chứa cách luật, các luật này sẽ báo cho chương trình make biết phải làm gì và làm như thế nào. Một luật bao gồm các thành phần như sau:

- Đích (target) – cái mà make phải làm
- Một danh sách các thành phần phụ thuộc (dependencies) cần để tạo ra đích
- Một danh sách các câu lệnh để thực thi trên các thành phần phụ thuộc

Khi được gọi, GNU make sẽ tìm các file có tên là GNUmakefile, makefile hay Makefile. Các luật sẽ có cú pháp như sau:

target: dependency1, dependency2, ….
command
command
……

Target thường là một file như file khả thi hay file object ta muốn tạo ra. Dependency là một danh sách các file cần thiết như là đầu vào để tạo ra target. Command là các bước cần thiết (chẳng hạn như gọi chương trình dịch) để tạo ra target.

Dưới đây là một ví dụ về một makefile về tạo ra một chương trình khả thi có tên là editor (số hiệu dòng chỉ đưa vào để tiện theo dõi, còn nội dung của makefile không chứa số hiệu dòng). Chương trình này được tạo ra bởi một số các file nguồn: editor.c, editor.h, keyboard.h, screen.h, screen.c, keyboard.c.

```
1.  editor : editor.o screen.o keyboard.o
2.          gcc -o  editor.o screen.o keyboard.o
3.  editor.o : editor.c editor.h keyboard.h screen.h
4.          gcc -c  editor.c
5.  screen.o : screen.c screen.h
6.          gcc -c  screen.c
7.  keyboard.o : keyboard.c keyboard.h
8.          gcc -c  keyboard.c
9.  clean:
10.          rm *.o
```

Để biên dịch chương trình này ta chỉ cần ra lệnh make trong thư mục chứa file này.

Trong makefile này chứa tất cả 5 luật, luật đầu tiên có đích là ***editor*** được gọi là đích ngầm định. Đây chính là file mà make sẽ phải tạo ra, editor có 3 dependencies editor.o, screen.o, keyboard.o. Tất cả các file này phải tồn tại thì mới tạo ra được đích trên. Dòng thứ 2 là lệnh mà make sẽ gọi thực hiện để tạo ra đích trên. Các dòng tiếp theo là các đích và các lệnh tương ứng để tạo ra các file đối tượng (object).

7.3.3. Làm việc với file

Trong Linux, để làm việc với file ta sử dụng mô tả file (file descriptor). Một trong những thuận lợi trong Linux và các hệ thống UNIX khác là giao diện file làm như nhau đối với nhiều loại thiết bị. Đĩa từ, các thiết bị vào/ra, cổng song song, giả

máy trạm (pseudoterminal), cổng máy in, bảng mạch âm thanh, và chuột được quản lý như các thiết bị đặc biệt giống như các tệp thông thường để lập trình ứng dụng. Các socket TCP/IP và miền, khi kết nối được thiết lập, sử dụng mô tả file như thể chúng là các file chuẩn. Các ống (pipe) cũng tương tự các file chuẩn.

Một mô tả file đơn giản chỉ là một số nguyên được sử dụng như chỉ mục (index) vào một bảng các file mở liên kết với từng tiến trình. Các giá trị 0, 1 và 2 liên quan đến các dòng (streams) vào ra chuẩn: *stdin, stderr* và *stdout*; ba dòng đó thường kết nối với máy của người sử dụng và có thể được chuyển tiếp (redirect).

Một số lời gọi hệ thống sử dụng mô tả file. Hầu hết các lời gọi đó trả về giá trị -1 khi có lỗi xảy ra và biến *errno* ghi mã lỗi. Mã lỗi được ghi trong trang chính tuỳ theo từng lời gọi hệ thống. Hàm *perror()* được sử dụng để hiển thị nội dung thông báo lỗi dựa trên mã lỗi.

Hàm open()

Lời gọi open() sử dụng để mở một file. Khuôn mẫu của hàm và giải thích tham số và cờ của nó được cho dưới đây:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

Đối số pathname là một xâu chỉ ra đường dẫn đến file sẽ được mở. Thông số thứ ba xác định chế độ của file Unix (các bit được phép) được sử dụng khi tạo một file và nên được sử dụng khi tạo một file . Tham số flags nhận một trong các giá trị O_RDONLY, O_WRONLY hoặc O_RDWR

Cờ	Chú giải
O_RDONLY	Mở file để đọc
O_WRONLY	Mở file để ghi
O_RDWR	Mở file để đọc và ghi
O_CREAT	Tạo file nếu chưa tồn tại file đó
O_EXCL	Thất bại nếu file đã có
O_NOCTTY	Không điều khiển *tty* nếu tty đã mở và tiến trình không điều khiển *tty*
O_TRUNC	Cắt file nếu nó tồn tại
O_APPEND	Nối thêm và con trỏ đặt ở cuối file
O_NONBLOCK	Nếu một quá trình không thể hoàn thành mà không có trễ, trả về trạng thái trước đó
O_NODELAY	Tương tự O_NONBLOCK
O_SYNC	Thao tác sẽ không trả về cho đến khi dữ liệu được ghi vào đĩa hoặc thiết bị khác

Các giá trị cờ của hàm open()

open() trả về một mô tả file nếu không có lỗi xảy ra. Khi có lỗi , nó trả về giá trị -1 và đặt giá trị cho biến errno. Hàm create() cũng tương tự như open() với các cờ O_CREATE | O_WRONLY | O_TRUNC

Hàm close()

Chúng ta nên đóng mô tả file khi đã thao tác xong với nó. Chỉ có một đối số đó là số mô tả file mà lời gọi open() trả về. Dạng của lời gọi close() là:

```
#include <unistd.h>
int close(int fd);
```

Tất cả các khoá (lock) do tiến trình xử lý trên file được giải phóng, cho dù chúng được đặt mô tả file khác. Nếu quá trình đóng file làm cho bộ đếm liên kết bằng 0 thì file sẽ bị xoá. Nếu đây là mô tả file cuối cùng liên kết đến một file được mở thì bản ghi ở bảng file mở được giải phóng. Nếu không phải là một file bình thường thì các hiệu ứng không mong muốn có thể xảy ra.

Hàm read()

Lời gọi hệ thống ***read()*** sử dụng để đọc dữ liệu từ file tương ứng với một mô tả file.

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);
```

Đối số đầu tiên là mô tả file mà được trả về từ lời gọi ***open()*** trước đó. Đối số thứ hai là một con trỏ tới bộ đệm để sao chép dữ liệu và đối số thứ ba là số byte sẽ được đọc. ***read()*** trả về số byte được đọc hoặc -1 nếu có lỗi xảy ra.

Hàm write()

Lời gọi hệ thống ***write()*** sử dụng để ghi dữ liệu vào file tương ứng với một mô tả file.

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t count);
```

Đối số đầu tiên là số mô tả file được trả về từ lời gọi ***open()*** trước đó. Đối số thứ hai là con trỏ tới bộ đệm (để sao chép dữ liệu, có dung lượng đủ lớn để chứa dữ liệu) và đối số thứ ba xác định số byte sẽ được ghi. write() trả về số byte đọc hoặc -1 nếu có lỗi xảy ra

Hàm ftruncate()

Lời gọi hệ thống ftruncate() cắt file tham chiếu bởi mô tả file fd với độ dài được xác định bởi tham số length

```
#include <unistd.h>
int ftruncate(int fd, size_t length);
```

Trả về giá trị 0 nếu thành công và -1 nếu có lỗi xảy ra.

Hàm lseek()

Hàm lseek() đặt vị trí đọc và ghi hiện tại trong file được tham chiếu bởi mô tả file files tới vị trí offset

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fildes, off_t offset, int whence);
```

Phụ thuộc vào giá trị của whence, giá trị của offset là vị trí bắt đầu (SEEK_SET), vị trí hiện tại (SEEK_CUR), hoặc cuối file (SEEK_END). Giá trị trả về là kết quả của offset: bắt đầu file, hoặc một giá trị của off_t , giá trị -1 nếu có lỗi.

Hàm fstat()

Hàm fstat () đưa ra thông tin về file thông qua việc mô tả các file, nơi kết quả của struct stat được chỉ ra ở con trỏ chỉ đến buf().Kết quả trả về giá trị 0 nếu thành công và nhận giá trị -1 nếu sai (kiểm tra lỗi).

```
#include <sys/stat.h>
#include <unistd.h>
int fstat(int filedes, struct stat *buf);
```

Sau đây là định nghĩa của struct stat:

```
struct stat
{
        dev_t           st_dev; / * thiết bị */
        int_t           st_ino  ;       /* inode */
        mode_t      st_mode;    /* chế độ bảo vệ */
        nlink_t     st_nlink;   /* số lượng các liên kết cứng  */
        uid_t           st_uid; /* số hiệu của người chủ */
        gid_t           st_gid; /* số hiệu nhóm của người chủ*/
        dev_t           st_rdev;    /* kiểu thiết bị */
        off_t           st_size;/* kích thước bytes */
        unsigned long st_blksize;   /* kích thước khối*/
        unsigned long st_blocks;    /* Số lượng các khối đã sử dụng*/
        time_t          st_atime;   /* thời gian truy cập cuối cùng*/
        time_t          st_mtime;   /* thời gian cập nhật cuối cùng */
        time_t          st_ctime;   /* thời gian thay đổi cuối cùng */
};
```

Hàm fchown()

Lời gọi hệ thống fchown() cho phép tathay đổi người chủ và nhóm người chủ kết hợp với việc mở file.

```
#include <sys/types.h>
#include <unistd.h>
int fchown(int  fd, uid_t  owner, gid_t  group);
```

Tham số đầu tiên là mô tả file, tham số thứ hai là số định danh của người chủ, và tham số thứ ba là số định danh của nhóm người chủ**.** Người dùng hoặc nhóm người dùng sẽ được

phép sử dụng khi giá trị -1 thay đổi. Giá trị trả về là 0 nếu thành công và –1 nếu gặp lỗi (kiểm tra biến errno).

Thông thường người dùng có thể thay đổi nhóm các file thuộc về họ. Chỉ root mới có quyền thay đổi người chủ sở hữu của nhiều nhóm.

Hàm fchdir( )

Lời gọi hàm fchdir( ) thay đổi thư mục bằng cách mở file được mô tả bởi biến fd. Giá trị trả về là 0 nếu thành công và –1 nếu có lỗi (kiểm tra biến errno).

```
#include <unistd.h>
int fchdir(int fd);
```

Một ví dụ về cách sử dụng các hàm thao tác với file:

```
/* filedes_io.c */
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <fcntl.h>
#include <unistd.h>

#include < assert.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>  /*for print */
char sample1[] = “This is sample data 1\n”;
char sample2[] = “This is sample data 2\n”;
char data[16];

main ( )
{
        int fd;
        int rc;
        struct   stat      statbuf;

        printf( “Creating file\n”);
        fd = open(“junk.out”, 0_WRONLY | 0_CREAT| 0_TRUNG, 0666);
        assert(fd>=0);

        rc = write(fd, sample1, strlen(sample 1) );
        assert(fd>=0);

        rc = write(fd, sample1, strlen(sample1));
        assert(rc == strlen(sample1));

        close(fd);
```

```
printf(“ Appending to file\n”);
fd = open(“junk.out”, 0_WRONLY| 0_APPEND);
assert(fd>=0);

printf( “ locking file\n”);
rc = flock(fd, LOCK_EX);
assert(rc == 0);

printf(“sleeping for 10 seconds\n”);
sleep(10);

printf(“writing data\n”);
rc = write(fd, sample2, strlen(sample2));
assert(rc == strlen(sample2));

printf(“unlocking file\n”);
rc = flock(fd, LOCK_UN);
assert(rc == 0);

close(fd);

printf(“Reading file \n”);
fd = open(“junk.out”, 0_RDONLY);
assert (fd >=0);

while (1)
{
        rc = read (fd, data, sizeof (data) );
        if( rc > 0 )
        {
                data[rc] =0;     /* kết thúc xâu */
                printf (“ Data read (rc = %d): <%s>\n”, rc, data);
        }
        else if (rc == 0)
        {
                printf (“ End of file read \ n”);
                break;
        } else
        {
                perror ( “ read error ” );
                break;
        }
}
close (fd);

printf (“ Fiddling with inode\n”);
fd = open (“ junk.out”, 0_RDONLY);
```

```
        assert (fd >= 0);

        printf (“ changing file mode\n”);
        rc = fchmod ( fd, 0600);
        assert (rc == 0);
        if ( getuid ( ) == 0 )     {
                 printf ( “ changing file owner \n ” );
                rc = fchown (fd, 99, 99);
                assert (rc == 0);
        } else
        {
                printf ( “ not changing file owner\n”);
        }

        fstat (fd, &statbuf);
        printf (“ file mode = 0% ( octal ) \ n”, statbuf.st_mode);
        printf(“Owner uid = %d \ n”, statbuf.st_uid);
        printf(“ Owner gid = %d \n”, statbuf.st_uid);

        close(fd);
}
```

7.3.4. Thư viện liên kết

Phần này sẽ giới thiệu cách tạo ra và sử dụng thư viện (các module chương trình đã được viết và được tái sử dụng nhiều lần). Thư viện gốc của C/C++ trên Linux chính là *glibc,* thư viện này cung cấp cho người dùng rất nhiều lời gọi hệ thống. Các thư viện trên Linux thường được tổ chức dưới dạng tĩnh (static library), thư viện chia sẻ (shared library) và động (dynamic library - giống như DLL trên MS Windows). Thư viện tĩnh được liên kết cố định vào trong chương trình trong quá trình liên kết. Thư viện dùng chung được nạp vào bộ nhớ trong khi chương trình bắt đầu thực hiện và cho phép các ứng dụng cùng chia sẻ loại thư viện này. Thư viện liên kết động được nạp vào bộ nhớ chỉ khi nào chương trình gọi tới.

7.3.4.1 Thư viện liên kết tĩnh

Thư viện tĩnh và các thư viện dùng chung (shared library) là các file chứa các file được gọi là các module đã được biên dịch và có thể sử dụng lại được. Chúng được lưu trữ dưới một định dạng đặc biệt cùng với một bảng (hoặc một bản đồ) phục vụ cho quá trình liên kết và biên dịch. Các thư viện liên kết tĩnh có phần mở rộng là .a. Để sử dụng các module trong thư viện ta cần thêm phần *#include* file tiêu đề (header) vào trong chương trình nguồn và khi liên kết (sau quá trình biên dịch) thì liên kết với thư viện đó. Dưới đây là một ví dụ về cách tạo và sử dụng một thư viên liên kết tĩnh. Có 2 phần trong ví dụ này, phần thứ nhất là mã nguồn cho thư viện và phần thứ 2 cho chương trình sử dụng thư viện.

```
/*
* liberr.h
*/

#ifndef  _LIBERR_H
#define  _LIBERR_H
#include <stdarg.h>
/* in ra một thông báo lỗi tới việcgọi stderr và return hàm gọi */
void err_quit(const char *fmt, … );
/*  in ra một thông điệp lỗi cho logfile và trả về hàm gọi */
void log_ret(char *logfile, const char *fmt, …);
 /* in ra một thông điệp lỗi cho logfile và thoát */
void log_quit( char *logfile, const char  *fmt , …);
/* in ra một  thông  báo lỗi và trả lại hàm gọi */
void err_prn(const char *fmt, va_list ap, char *logfile);
#endif //_LIBERR_H
```

Mã nguồn cho file liberr.h

```
#include <errno.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include  "liberr.h"
#define MAXLINELEN 500
void err_ret(const char *fmt, ...)
{
                va_list ap;
                va_start(ap, fmt);
                err_prn(fmt, ap, NULL);
                va_end(ap);
                return;
}
void err_quit(const char *fmt, ...)
{
                va_list ap;
                va_start(ap, fmt);
                err_prn(fmt, ap, NULL);
                va_end(ap);
                exit(1);
}
void log_ret(char *logfile, const char *fmt, ...)
{
                va_list ap;
                va_start(ap, fmt);
                err_prn(fmt,ap, logfile);
                va_end(ap);
                return;
}
```

```
void log_quit(char *logfile, const char *fmt,... )
{
		va_list ap;
		va_start(ap, fmt);
		err_prn(fmt, ap,logfile);
		va_end(ap);
		exit(1);
}
extern void err_prn( const char *fmt, va_list ap, char *logfile)
{
		int save_err;
		char buf[MAXLINELEN];
		FILE *plf;

		save_err = errno;
		vsprintf(buf,fmt, ap);
		sprintf( buf+strlen(buf),  ": %s", strerror(save_err));
		strcat(buf, "\n");
		fflush(stdout);
		if(logfile !=NULL){
		if((plf=fopen(logfile, "a") ) != NULL){
			fputs(buf, plf);
			fclose(plf);
		}else
			fputs("failed to open log file \n", stderr);
		}else	fputs(buf, stderr);
		fflush(NULL);
		return;
}
```

Mã nguồn file liberr.c

Để tạo một thư viện tĩnh, bước đầu tiên là dịch đoạn mã của form đối tượng:

$gcc –H –c liberr.c –o liberr.o

tiếp theo:

$ar rcs liberr.a liberr.o

```
/*
* errtest.c
*/
#include <stdio.h>
#include <stdlib.h>
#include "liberr.h"
#define ERR_QUIT_SKIP 1
#define LOG_QUIT_SKIP 1

int main(void)
{
```

```
        FILE *pf;

        fputs("Testing err_ret()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL)
        err_ret("%s %s", "err_ret()", "failed to open foo");

        fputs("Testing log_ret()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL);
        log_ret("errtest.log", "%s %s", "log_ret()",
        "failed to open foo");

#ifndef ERR_QUIT_SKIP
        fputs("Testing err_quit()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL)
        err_ret("%s %s", "err_quit()", "failed to open foo");
#endif /* ERR_QUIT_SKIP */

#ifndef LOG_QUIT_SKIP
        fputs("Testing log_quit()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL)
        log_ret("errtest.log", "%s %s", "log_quit()", "failed to open foo");
#endif /* LOG_QUIT_SKIP */

        return EXIT_SUCCESS;
}
```

Mã nguồn file testerr.c

Biên dịch chương trình kiểm tra, ta sử dụng dòng lệnh:

$ gcc -g errtest.c -o errtest -L. -lerr

Tham số -L. chỉ ra đường dẫn tới thư mục chứa file thư viện là thư mục hiện thời, tham số –lerr chỉ rõ thư viện thích hợp mà chúng ta muốn liên kết. Sau khi dịch ta có thể kiểm tra bằng cách chạy chương trình.

7.3.4.2 Thư viện dùng chung

Thư viện dùng chung có nhiều thuận lợi hơn thư viện tĩnh.Thứ nhất, thư viện dùng chung tốn ít tài nguyên hệ thống, chúng sử dụng ít không gian đĩa vì mã nguồn thư viện dùng chung không biên dịch sang mã nhị phân nhưng được liên kết và được dùng tự động mỗi lần dùng. Chúng sử dụng ít bộ nhớ hệ thống vì nhân chia sẻ bộ nhớ cho thư viện dùng chung này và tất cả các chương trình đều sử dụng chung miền bộ nhớ này. Thứ 2, thư viện dùng chung nhanh hơn vi chúng chỉ cần nạp vào một bộ nhớ. Lí do cuối cùng là mã nguồn trong thư viện dùng chung dễ bảo trì. Khi các lỗi được sửa hay thêm vào các đặc tính, người dùng cần sử dụng thư viện nâng cấp. Đối với thư viện tĩnh, mỗi chương trình khi sử dụng thư viện phải biên dịch lại.

Trình liên kết (linker)/module tải (loader) ***ld.so*** liên kết tên biểu tượng tới thư viện dùng chung mỗi lần chạy. Thư viện dùng chung có tên đặc biệt (gọi là soname), bao gồm tên thư viện và phiên bản chính. Ví dụ: tên đầy đủ của thư viện C trong hệ thống là libc.so.5.4.46, tên thư viện là libc.so, tên phiên bản chính là 5, tên phiên bản phụ là 4, 46 là mức vá (patch

level). Như vậy, soname thư viện C là libc.5. Thư viện libc6 có soname là libc.so.6, sự thay đổi phiên bản chính là sự thay đổi đáng kể thư viện. Phiên bản phụ và patch level thay đổi khi lỗi được sửa nhưng soname không thay đổi và bản mới có sự thay khác biệt đáng kể so với bản cũ.

Các chương trình ứng dụng liên kết dựa vào soname. Tiện ích ***idconfig*** tạo một biểu tượng liên kết từ thư viện chuẩn libc.so.5.4.46 tới soname libc.5 và lưu trữ thông tin này trong /etc/ld.so.cache. Trong lúc chạy, ld.so đọc phần lưu trữ, tìm soname thích hợp và nạp thư viện hiện tai vào bộ nhớ, kết nối hàm ứng dụng gọi tới đối tượng thích hợp trong thư viện.

Các phiên bản thư viện khác nhau nếu:

- Các giao diện hàm đầu ra thay đổi.
- Các giao diện hàm mới được thêm.
- Chức năng hoạt động thay đổi so với đặc tả ban đầu
- Cấu trúc dữ liệu đầu ra thay đổi
- Cấu trúc dữ liệu đầu ra được thêm

Để duy trì tính tương thích của thư viện, cần đảm bảo các yêu cầu:

- Không thêm vào những tên hàm đã có hoặc thay đổi hoạt động của nó
- Chỉ thêm vào cuối cấu trúc dữ liệu đã có hoặc làm cho chúng có tính tuỳ chọn hay được khởi tạo trong thư viện
- Không mở rộng cấu trúc dữ liệu sử dụng trong các mảng

Xây dựng thư viện dùng chung hơi khác so với thư viện tĩnh, quá trình xây dựng thư viện dùng chung được minh hoạ dưới đây:

- Khi biên dịch file đối tượng, sử dụng tùy chọn -fpic của gcc nó sẽ tạo ra mã độc lập vị trí (position independence code) từ đó có thể liên kết hay sử dụng ở bất cứ chỗ nào
- Không loại bỏ file đối tượng và không sử dụng các tùy chọn ***–fomit –frame -pointer*** của ***gcc***, vì nếu không sẽ ảnh hưởng đến quá trình gỡ rối (debug)
- Sử dụng tuỳ chọn ***-shared*** and ***–soname*** của ***gcc***
- Sử dụng tuỳ chọn ***–Wl*** của ***gcc*** để truyền tham số tới trình liên kết ***ld***.
- Thực hiện quá trình liên kết dựa vào thư viện C, sử dụng tuỳ chọn ***–l*** của ***gcc***

Trở lại thư viện xử li lỗi , để tạo thư viện dùng chung trước hết xây dựng file đối tượng:

$ gcc -fPiC -g -c liberr.c -o liberr.o

Tiếp theo liên kết thư viện:

$ gcc -g -shared -Wl,-soname,liberr.so -o liberr.so.1.0.0 liberr.o -lc

Vì không thể cài đặt thư viện này như thư viện hệ thống trong /usr hay /usr/lib chúng ta cần tạo 2 kiên kết, một cho soname:

Và cho trình liên kết khi kết nối dựa vào liberr, sử dụng ***–lerr***:

$ ln -s liberr.so.1.0.0 liberr.so

Bây giờ, để dử dụng thư viện dùng chung mới chúng ta quay lại chương trình kiểm tra, chúng ta cần hướng trình liên kết tới thư viện nào để sử dụng và tìm nó ở đâu, vì vậy chúng ta sẽ sử dụng tuỳ chọn –l và –L:

$ gcc -g errtest.c -o errtest -L. -lerr

Cuối cùng để chạy chương trình, chúng ta cần chỉ cho ld.so nơi để tìm thư viện dùng chung :

$ LD_LIBRARY_PATH=$(pwd) ./errtest

7.3.4.3 Sử dụng đối tượng dùng chung theo cách động

Một cách để sử dụng thư viện dùng chung là nạp chúng tự động mỗi khi chạy không giống như những thư viện liên kết và nạp một cách tự động. Ta có thể sử dụng giao diện dl (dynamic loading) vì nó tạo sự linh hoạt cho lập trình viên hay người dùng.

Giả sử ta đang tạo một ứng dụng sử lý đồ hoạ. Trong ứng dụng, ta biểu diễn dữ liệu ở một dạng không theo chuẩn nhưng lại thuận tiện cho ta xử lý, và ta cần có nhu cầu chuyển dữ liệu đó ra các định dạng thông dụng đã có (số lượng các định dạng này có thể có hàng trăm loại) hoặc đọc dữ liệu từ các định dạng mới này vào để xử lý. Để giải quyết vấn đề này ta có thể sử dụng giải pháp là thư viện. Nhưng khi có thêm một định dạng mới thì ta lại phải biên dịch lại chương trình. Đây lại là một điều không thích hợp lắm. Khả năng sử dụng thư viện động sẽ giúp ta giải quyết vấn đề vừa gặp phải. Giao diện ***dl*** cho phép tạo ra giao diện (các hàm) đọc và viết chung không phụ thuộc vào định dạng của file ảnh. Để thêm hoặc sửa các định dạng của file ảnh ta chỉ cần viết thêm một module để đảm nhận chức năng đó và báo cho chương trình ứng dụng biết là có thêm một module mới bằng cách chỉ cần thay đổi một file cấu hình trong một thư mục xác định nào đó.

Giao diện ***dl*** (cũng đơn thuần được xây dựng như một thư viện - thư viện ***libdl***) chứa các hàm để tải (load), tìm kiếm và giải phóng (unload) các đối tượng chia sẻ. Để sử dụng các hàm này ta thêm file *<dlfcn.h>* vào phần *#include* vào trong mã nguồn, và khi dịch thì liên kết nó với thư viện ***libdl*** bằng cách sử dụng tham số và tên ***–ldl*** trong dòng lệnh dịch.

dl cung cấp 4 hàm xử lí các công việc cần thiết để tải, sử dụng và giải phóng đối tượng dùng chung.

Truy cập đối tượng chia sẻ

Để truy cập một đối tượng chia sẻ, dùng hàm dlopen() có đặc tả như sau:

```
void *dlopen(const char *filename, int flag);
```

dlopen() truy cập đối tượng chia sẻ bằng filename và bằng cờ. Filename có thể là đường dẫn đầy đủ, tên file rút gọn hay NULL. Nếu là NULL ***dlopen()*** mở chương trình đang chạy, đó là chương trình của bạn, nếu filename là đường dẫn ***dlopen()*** mở file đó, nếu là tên rút gọn ***dlopen()*** sẽ tìm trong vị trí sau để tìm file:

$LD_ELF_LiBRARY_PATH,
$LD_LIBRARY_PATH, /etc/ld.so.cache, /usr/lib, và /lib.

Cờ có thể là RTLD_LAZY, có nghĩa là các kí hiệu (symbol) hay tên hàm từ đối tượng truy cập sẽ được tìm mỗi khi chúng được gọi, hoặc cờ có thể là RTLD_NOW, có nghĩa tất cả kí hiệu từ đối tượng truy cập sẽ được tìm trước khi hàm ***dlopen()*** trả về. ***dlopen()*** trả điều khiển tới đối tượng truy nhập nếu nó tìm thấy từ filename hay trả về giá trị NULL nếu không tìm thấy.

Sử dụng đối tượng chia sẻ

Trước khi có thể sử dụng mã nguồn trong thư viện ta phải biết đang tìm cái gì và tìm ở đâu. Hàm dlsym() sẽ giúp điều đó:

```
void *dlsym(void *handle, char *symbol);
```

dlsym() tìm kí hiệu hay tên hàm trong truy cập và trả lại con trỏ kiểu void tới đối tượng hay NULL nếu không thành công.

Kiểm tra lỗi

Hàm ***dlerror()*** sẽ giúp ta kiểm tra lỗi khi sử dụng đối tượng truy cập động:

```
const char *dlerror(void);
```

Nếu một trong các hàm lỗi, ***dlerror()*** trả về thông báo chi tiết lỗi và gán giá trị NULL cho phần bị lỗi.

Giải phóng đối tượng chia sẻ

Để bảo vệ tài nguyên hệ thống đặc biệt bộ nhớ, khi ta sử dụng xong module trong một đối tượng chia sẻ, thì giải phóng chúng. Hàm ***dlclose()*** sẽ đóng đối tượng chia sẻ:

```
int dlclose(void *handle);
```

Sử dụng giao diện dl

Để minh hoạ cách sử dụng ***dl***,chúng ta quay lại thư viện xử lí lỗi, sử dụng một chương trình khác như sau:

```
/*
* dltest.c
* Dynamically load liberr.so and call err_ret()
*/
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
 int main(void)
 {
        void *handle;
        void (*errfcn)();
        const char *errmsg;
        FILE *pf;

        handle = dlopen("liberr.so", RTLD_NOW);
        if(handle == NULL) {
                fprintf(stderr, "Failed to load liberr.so: %s\n", dlerror());
                exit(EXIT_FAILURE);
        }
        dlerror();
        errfcn = dlsym(handle, "err_ret");
        if((errmsg = dlerror()) != NULL) {
                fprintf(stderr, "Didn't find err_ret(): %s\n", errmsg);
                exit(EXIT_FAILURE);
        }
        if((pf = fopen("foobar", "r")) == NULL)
                errfcn("couldn't open foobar");
        dlclose(handle);
        return EXIT_SUCCESS;
 }
```

Mã nguồn chương trình dltest.c

Biên dịch ví dụ trên bằng lệnh:

$ gcc -g -Wall dltest.c -o dltest -ldl

Như tacó thể thấy, chúng ta không liên kết dựa vào liberr hay liberr.h trong mã nguồn. Tất cả truy cập tới liberr.so thông qua ***dl***. Chạy chương trình bằng cách sau:

$ LD_LIBRARY_PATH=$(pwd) ***./dltest***

Nếu thành công thì ta nhận được kết quả như sau:

couldn’t open foobar: No such file or directory

7.3.5 Các công cụ cho thư viện

Công cụ nm

Lệnh ***nm*** liệt kê toàn bộ các tên hàm (symbol) được mã hoá trong file đối tượng (object) và nhị phân (binary). Lệnh nm sử dụng cú pháp sau:

nm [options] file

Lênh nm liệt kê những tên hàm chứa trong file. Bảng dưới liệt kê các tuỳ chọn của lệnh nm:

Tuỳ chọn	Miêu tả
-C| -demangle	Chuyển tên ký tự vào tên mức người dùng để cho dễ đọc.
-s|-print-armap	Khi sử dụng các file lưu trữ (phần mở rộng là ".a"), in ra các chỉ số của module chứa hàm đó.
-u| -undefined-only	Chỉ đưa ra các hàm không được định nghĩa trong file này, tức là các hàm được định nghĩa ở một file khác.
- l | -line-numbers	Sử dụng thông tin gỡ rối để in ra số dòng nơi hàm được định nghĩa.

Các tuỳ chọn của lệnh nm

Công cụ ar

Lệnh ar sử dụng cú pháp sau:

ar {dmpqrtx} [thành viên] file

Lệnh ar tạo, chỉnh sửa và trích các file lưu trữ. Nó thường được sử dụng để tạo các thư viện tĩnh- những file mà chứa một hoặc nhiều file đối tượng chứa các chương trình con thường được sử dụng (subrountine) ở định dạng tiền biên dịch (precompiled format), lệnh ar cũng tạo và duy trì một bảng mà tham chiếu qua tên ký tự tới các thành viên mà trong đó chúng được định nghĩa. Chi tiết của lệnh này đã được trình bày trong chương trước.

Công cụ idd

Lệnh ***nm*** liệt kê các hàm được định nghĩa trong một file đối tượng, nhưng trừ khi ta biết những gì thư viện định nghĩa những hàm nào. Lệnh ***idd*** hữu ích hơn nhiều. ***idd*** liệt kê các thư viện được chia sẻ mà một chương trình yêu cầu để mà chạy. Cú pháp của nó là:

idd [options] file

Lệnh ***idd*** in ra tên của thư viện chia sẻ mà file này sử dụng. Ví dụ: chương trình thư "client mutt" cần 5 thư viện chia sẻ, như được minh hoạ sau đây:

```
$ idd /usr/bin/mutt
        libnsl.so.1 => /lib/libns1.so.1 (0x40019000)
        libslang.so.1 => /usr/lib/libslang.so.1 (0x4002e000)
        libm.so.6 => /lib/libm/so.6 (0x40072000)
        libc.so.6 => /lib/libc.so.6 (0x4008f000)
        /lib/id-linux.so.2 => /lib/id-Linux.so.2 (0x4000000)
```

Tìm hiểu lệnh idconfig

Lệnh ***idconfig*** sử dụng cú pháp sau:

ldconfig [tuỳ chọn] [libs]

Lệnh ***ldconfig*** xác định rõ các liên kết động (liên kết khi chạy) được yêu cầu bởi thư viện được chia sẻ nằm trong các thư mục /usr/lib và /lib. Dưới đây là các tùy chọn của lệnh này:

Các tuỳ chọn	Các miêu tả
-p	Đơn thuần chỉ in ra nội dung của /etc/ld.so.cache, một danh sách hiện thời các thư viện được chia sẻ mà ld.so biết.
-v	Cập nhật /etc/ld.so.cache , liệt kê số phiên bản của mỗi thư viện, quét các thư mục và bất kỳ liên kết mà được tạo ra hoặc cập nhật.

Các tuỳ chọn của hàm idconfig

Biến môi trường và file cấu hình.

Chương trình tải (loader) và trình liên kết (linker) ***ld.so*** sử dụng 2 biến môi trường. Biến thứ nhất là $LD_LIBRARY, chứa danh sách các thư mục chứa các file thư viện được phân cách bởi dấu hai chấm để tìm ra các thư viện cần thiết khi chạy. Nó giống như biến môi trường $PATH. Biến môi trường thứ hai là $LD_PRELOAD, một danh sách các thư viện được người dùng thêm vào được phân cách nhau bởi khoảng trống (space).

ld.so cũng cho phép sử dụng 2 file cấu hình mà có cùng mục đích với biến môi trường được đề cập ở trên. File /etc/ld.so.conf chứa một danh sách các thư mục mà chương trình tải và trình liên kết (loader/linker) nên tìm kiếm các thư viện chia sẻ bên cạnh /usr/lib và /lib. /etc/ld.so.preload chứa danh sách các file thư viện được phân cách bằng một khoảng trống các thư viện này là thư viện người dùng tạo ra.

TÀI LIỆU THAM KHẢO

……

CHÚ THÍCH MỘT SỐ THUẬT NGỮ

Vấn đề chọn thuật ngữ tiếng Việt cho các thuật ngữ gốc tiếng Anh luôn là vấn đề cần được thảo luận để việc chọn lựa như vậy là thích hợp. Dù cho trong tài liệu này, nhiều thuật ngữ tiếng Việt đã được sử dụng một cách phổ biến nhưng tài liệu cũng liệt kê vào bảng chú thích dưới đây với mục đích giúp bạn đọc thuận tiện trong tra cứu và liên hệ. Trong bảng chú thích, mỗi thuật ngữ tiếng Việt dùng trong tài liệu sẽ được kèm theo thuật ngữ tiếng Anh gốc và sau đó có thể là một lời chú thích ngắn. Bảng chú thích là một cố gắng nhỏ nhằm hỗ trợ sử dụng tài liệu này hiệu quả hơn và phạm vi sử dụng của nó giới hạn trong tài liệu.

Tiếng Việt	Tiếng Anh	Giải thích ngắn
File	File	Tập hợp dữ liệu có tổ chức theo một mục đích sử dụng: đoạn văn bản, một chương trình nguồn, một tập hợp dữ liệu ... Còn được gọi là *tệp* hay gọi theo cách giữ nguyên gốc tiếng Anh là *File*.
Tên file	Filename	Tên gọi của file (file thường, thư mục, file đặc biệt ...) được dùng để xác định file.
Thư mục	Directory	Nơi chứa một danh sách các file trong hệ thống File, được Linux coi như một dạng file đặc biệt.
Hệ thống file	File system	Hệ thống toàn bộ các file có trong hệ điều hành Linux (và UNIX nói chung).
Lệnh	Command	Linux cho phép người sử dụng dùng lệnh để trình bày một "thao tác" yêu cầu hệ thống thực hiện.
Tiện ích (còn gọi là lệnh thường trực)	Utility	Lệnh có sẵn trong Linux (trong shell hoặc được đặt tại những thư mục theo quy định của Linux).
Dòng lệnh	Command line	Dòng lệnh bao gồm một hoặc một số lệnh trong một lần yêu cầu của người dùng. Kết thúc dòng lệnh là dấu xuống dòng.
Đăng nhập	Login	Thủ tục bắt đầu một phiên làm việc của một người sử dụng. Login là lệnh cho phép đăng nhập. Thủ tục logout kết thúc phiên làm việc.
Người dùng	User	Người sử dụng Linux cho công việc của mình.
Siêu người dùng (người dùng tối cao/người quản trị)	Superuser	Linux quy định có một siêu người dùng thực hiện có quyền hạn tối cao trong hệ thống bao gồm các lệnh quản trị hệ thống. Với một số lệnh Linux (thêm, bớt người dùng ...) thì chỉ siêu người dùng được phép sử dụng.
Con trỏ	cursor	Điểm đánh dấu trên màn hình đánh dấu vị trí hiện thời để hiện thông tin. Trong trình soạn thảo văn bản, nó là vị trí soạn thảo hiện thời trong văn bản
Hệ điều hành	Operating System (OS)	Bộ chương trình bao gồm các file trên đĩa (băng) từ có chức năng quản lý tài nguyên máy tính và đóng vai trò là "máy tính ảo" đối với người dùng.

Hệ điều hành đa chương trình	Multiprogram-ming OS	Hệ điều hành hoạt động theo cách thức trong bộ nhớ đồng thời có nhiều chương trình được "bình đẳng" trong phân phối tài nguyên (CPU, bộ nhớ).
Hệ điều hành đa người dùng	Multi-users OS	Mỗi người dùng sử dụng một trạm cuối được kết nối máy tính để trực tiếp thực hiện công việc trên máy tính (có đa chương trình).
Nhân	Kernel	Bộ phận cốt lõi nhất của Linux, thường trực để thực hiện các chức năng cơ bản của hệ điều hành (còn được gọi là ***lõi***)
Quá trình	Process	Một lần thực hiện của một chương trình (Còn được gọi là tiến trình). Một số hệ điều hành gọi là bài toán (task).
Chương trình giải thích lệnh	Command Comment Program (CCP)	Một chương trình thuộc hệ điều hành có chức năng tiếp nhận, phân tích dòng lệnh người dùng để lệnh được thực hiện. Trong Linux là shell, trong MS-DOS là COMMAND.COM.
Đăng nhập	Login	Thủ tục mà một người dùng cần phải thực hiện khi bắt đầu phiên làm việc với Linux. Thủ tục này yêu cầu người dùng đưa vào tên kí hiệu và mật khẩu đã được đăng kí trong hệ thống.
Đăng xuất	Logout	Thủ tục mà một người dùng cần phải thực hiện khi kết thúc phiên làm việc với Linux. Trên máy tính cá nhân, sau khi đăng xuất sẽ kéo theo một đăng nhập mới.
Dấu nhắc shell		Còn gọi là dấu nhắc dòng lệnh
Kí hiệu mô tả nhóm	wildcards	Kí hiệu được dùng để xác định một nhóm file (ví dụ *, ?, [] và [] với -). Còn được gọi là kí hiệu thay thế.
Cài đặt Linux	Installation Linux	Quá trình tạo ra hệ điều hành Linux lên máy tính để sử dụng

PHỤ LỤC A. QUÁ TRÌNH CÀI ĐẶT REDHAT-LINUX

Linux sử dụng phần cứng của máy PC hiệu quả hơn MS-DOS, Window hay WinNT, và do đó khả năng chịu các lỗi do cấu hình sai phần cứng sẽ kém hơn. Chúng ta cần làm một số việc trước khi bắt đầu cài đặt để giảm thiểu các khả năng không thể cài đặt tiếp khi gặp phải vấn đề này.

Trước tiên, hãy cố gắng tìm càng nhiều càng tốt các tài liệu về phần cứng máy PC mà mình định cài, như mainboard, card đồ hoạ, màn hình, modem... và để chúng ở nơi có thể tìm thấy và tra cứu dễ dàng.

Tiếp theo, tìm hiểu thông tin về phần cứng máy tính của và tập hợp chúng lại. Chúng ta có thể làm điều này khi sử dụng chức năng in cấu hình máy của một số tiện ích như MSD trong DOS, hoặc System Information trong Windows. Những thông tin chính xác về bàn phím, chuột, màn hình... sẽ giúp rất nhiều trong quá trình cấu hình X sau này.

Sau đó, kiểm tra phần cứng máy tính của để tìm ra các vấn đề nếu có, bởi chúng có thể làm quá trình cài đặt Linux bị treo. Sau đây là một số vấn đề thường gặp.

Một hệ thống DOS hay Windows có thể quản lý ổ đĩa IDE và CDROM cả khi jumper master/slave không được đặt đúng. Trong khi Linux không giải quyết được vấn đề này. Vì vậy nếu có nghi ngờ hãy xem lại các jumper này đã được đặt đúng chưa.

Một số thiết bị ngoại vi cần có những tiện ích để đặt cấu hình cho chúng khi máy khởi động. Các thiết bị như card mạng, CD-ROM, card âm thanh hoặc băng từ có thể gặp phải vấn đề này. Nếu trường hợp này xảy ra, có thể sử dụng lệnh đặt cấu hình lại tại dấu nhắc khởi động.

Một số hệ điều hành khác cho phép chuột dạng bus chia sẻ một IRQ với các thiết bị khác, trong khi Linux không hỗ trợ điều này. Nếu thử làm điều đó, hệ thống có thể bị treo.

Nên liên hệ với người dùng Linux có kinh nghiệm để được giải đáp những vướng mắc.

Cần tiến hành công việc chuẩn bị thời gian cho việc cài đặt. Quá trình cài đặt có thể kéo dài một tiếng hoặc hơn với những hệ thống chỉ cài Linux, hoặc lên tới ba tiếng với hệ thống cần chạy nhiều hệ điều hành khác nhau (thường với những hệ thống này khả năng bị treo máy hoặc có lỗi khi cài đặt sẽ cao hơn).

Phụ lục này giới thiệu cách cài đặt hai phiên bản RedHat là 6.2 và 7.1 nhằm cung cấp tính đa dạng khi cài đặt, tạo điều kiện cho người sử dụng.

AA. Cài đặt phiên bản RedHat 6.2

AA.1. Tạo đĩa mềm khởi động

Bước này chỉ cần thiết khi không thể khởi động từ ổ CD-ROM.

Nếu mua Red Hat Linux trực tiếp từ Red Hat Linux sẽ nhận kèm đĩa khởi động. Còn nếu mua bản copy từ công ty thứ 3, ta phải tự tạo đĩa khởi động và cách tạo là như sau:

Trong Windows, đưa đĩa mềm vào ổ. Bấm phím phải chuột vào desktop để tạo folder mới, đặt tên ***Bootdisk*** rồi mở folder này.

Đưa đĩa CD Red Hat vào ổ CD. Mở My Computer, nhấn vào ổ CD, mở folder có tên ***Dosutils***, nhấn vào file ***Rawrite***, bấm phím phải chuột và kéo nó vào ***Bootdisk***. Chọn Copy here từ menu xuất hiện.

Đóng cửa sổ Dosutils. Mở folder ***Images*** trong CD-ROM. Chép file ***Boot.img*** vào folder Bootdisk, giống như đã làm với ***Rawrite***.

Chọn Start.Run gõ vào Command trong hộp hội thoại, nhấn OK. Một cửa sổ xuất hiện với dấu nhắc DOS "C:\Windows\Desktop". Gõ vào ***cd bootdisk***, nhấn Enter.

Khi đã hoàn tất việc tạo đĩa mềm khởi động: gõ ***rawrite*** tại dấu nhắc DOS. Nhập ***boot.img*** là tên file muốn copy, nhấn Enter. Gõ ***a:***\ (tên ổ đĩa mềm), và nhấn Enter khi được hỏi ổ đích.

AA.2. Phân vùng lại ổ đĩa DOS/Windows hiện thời

Trong hầu hết các hệ thống được sử dụng, ổ cứng thường được phân vùng cho MS-DOS, OS/2,... Cần thay đổi kích thước, sắp xếp lại các phân vùng này để tạo chỗ trống cho việc cài đặt Linux.

Cách tốt nhất để làm việc này là dùng một phần mềm chuyên dụng, chẳng hạn như phần mềm ***PQMagic*** của Power Quest. Dùng phần mềm này, có thể di chuyển / thay đổi kích thước / thêm / xoá / format các phân vùng trong ổ cứng một cách dễ dàng với giao diện đồ hoạ. Còn việc dùng FDisk của MS-DOS thì cực kỳ vất vả, muốn di chuyển / thay đổi kích thước của một phân vùng nào đó, đầu tiên phải ***backup*** tất cả các dữ liệu trong phân vùng, xoá phân vùng đó (việc này sẽ làm mất các thông tin về dữ liệu trong phân vùng), tiếp theo là tạo một phân vùng mới với kích thước mong muốn, cuối cùng là ***restore*** lại toàn bộ dữ liệu đã backup vào phân vùng mới tạo này! Tốt nhất dùng một phần mềm miễn phí cực mạnh như ***PQMagic***.

AA.3. Các bước cài đặt (bản RedHat 6.2 và khởi động từ CD-ROM)

Đưa đĩa CDROM Redhat 6.2 vào ổ CD, sau đó trong BIOS SETUP ta đặt chế độ khởi động từ ổ CD. Khi khởi động lại máy, quá trình sẽ được boot từ CDROM. Sau đây là chi tiết quá trình một đĩa khởi động cài đặt tiến hành.

Lựa chọn chế độ cài đặt

Hệ thống đưa ra các chế độ cho chúng ta lựa chọn :

- Gõ Enter chọn chế độ cài đặt đồ họa.
- Gõ "text" + Enter chọn chế độ Text.
- Gõ "expert" + Enter chọn chế độ Expert. Chọn chế độ này có nghĩa ta tự chọn cấu hình phần cứng còn hai chế độ trên sẽ tự động detect.
- Gõ "linux ks" + Enter chọn chế độ cài đặt từ mạng hoặc từ đĩa mềm.

Lựa chọn ngôn ngữ hiển thị.

Hệ thống đưa ra rất nhiều ngôn ngữ cho phép lựa chọn, ví dụ như Czech, English, French, German.... Thường chọn English.

Lựa chọn cấu hình bàn phím

Linux đòi hỏi lựa chọn cấu hình bàn phím từ các mô hình sau:

- Model :

  Brazilian ABNT 2
  Dell
  Generic.
  Microsoft.
  .....

Ta sẽ lựa chọn bàn phím tương thích theo các dạng trên, nếu không rõ bàn phím thuộc loại nào thì nên chọn kiểu Generic.

- **Layout**: Ngôn ngữ sử dụng để gõ (khoảng 24 ngôn ngữ)
- **Variant**: Có 2 chế độ là.

  Eliminate Dead Keys (khi chúng ta sử dụng các kí tự đặc biệt).
  None (kiểu mặc định).
- **Test**: chúng ta sẽ gõ các phím để kiểm tra thử.

Chọn cấu hình chuột

Hệ thống đưa ra 10 loại chuột để lựa chọn loại tương thích với chuột của mình, nếu không biết rõ chuột thuộc loại nào thì nên chọn kiểu Generic. Và phải chọn đúng kiểu chuột là PS/2 hay Serial nếu không thì sẽ không sử dụng được chuột.

Hệ thống đưa ra lời giới thiệu về bản Red Hat đang cài đặt.

Lựa chọn kiểu cài đặt.

Install (cài mới) gồm có các chế độ:

GNome Workstation.

KDE Workstation.

Server

Custom

Upgrade (Nâng cấp)

Tuỳ chọn cài đặt WorkStation

Nếu lựa chọn kiểu cài mới là GNome Workstation hoặc KDE Workstation thì hệ thống sẽ cài đặt X Window System và chương trình quản lý Desktop theo dạng GNome hoặc KDE. Nếu chưa thạo lắm về Linux thì hãy sử dụng tuỳ chọn này, nó sẽ bỏ qua nhiều bước. Lựa chọn **Custom** là chỉ phù hợp với người đã quen với Linux. Cả hai lựa chọn WorkStation này sẽ chuẩn bị các việc sau:

Nếu đĩa cứng của chưa hề được phân vùng trước đó, Linux sẽ xoá hết tất cả các phân vùng trên các ổ đĩa cứng và cài đặt các phân vùng sau:

+ Một phân vùng swap kích thước 64MB

+ Một phân vùng root (được mount là /) chứa đựng mọi file được cài đặt.

Chú ý là chúng ta sẽ phải cần tối thiểu 600MB ổ cứng để tiến hành cài đặt theo kiểu WorkStattion.

Nếu hệ thống đã được cài sẵn Windows (Windows 3.1/95/98), kiểu cài đặt WorkStation sẽ tự động cấu hình hệ thống để khởi động ở chế độ song song sử dụng LILO.

Tuỳ chọn cài đặt Server

Nếu lựa chọn kiểu cài đặt mới là Server thì hệ thống sẽ cài đặt theo kiểu máy chủ Linux và cũng như kiểu WorkStation, tuỳ chọn này sẽ tránh phải cấu hình nhiều thành phần phần cứng. Và khi cài đặt theo kiểu này, nên cẩn thận vì hệ thống sẽ xoá tất cả các partition trên

tất cả các ổ đĩa. Vì vậy, *chỉ chọn kiểu cài đặt **Server** nếu chắc chắn trong ổ cứng không có một dữ liệu gì cả*. Sau đây là các bước mà kiểu này tiến hành phân vùng ổ đĩa:

+ Tạo một phân vùng Swap kích thước 64MB.

+ Tạo một phân vùng kích thước 256MB với mount point là "/".

+ Tạo một phân vùng kích thước tối thiểu 512MB được mount là "/usr".

+ Tạo một phân vùng tối thiểu 215MB được mount là "/home".

+ Tạo một phân vùng kích thước 256MB được mount là "/var".

Như vậy chúng ta phải cần tối thiểu **1.6BG** ổ cứng để tiến hành cài đặt theo kiểu Server.

Tuỳ chọn cài đặt *Custom*

Đối với kiểu Custom hệ thống sẽ cho phép tự chọn các thành phần và cấu hình phần cứng để cài đặt một cách đầy đủ nhất, tuy nhiên cũng rối rắm và phức tạp nhất. Từ bước 7 trở đi, chỉ giới thiệu về tuỳ chọn cài đặt này. Sau đây sẽ giới thiệu tổng quát một số bước mà quá trình này thiết đặt:

Tạo các phân vùng: cần phải chỉ rõ Redhat sẽ được cài đặt vào phân vùng nào.

Format các phân vùng: Những phân vùng mới được thêm vào sẽ phải được format lại theo định dạng Linux filesystem. Tuy nhiên cũng có thể lựa chọn phân vùng nào cần phải format.

Lựa chọn và cài đặt các gói phần mềm đi kèm: Thực hiện sau khi đã phân vùng đĩa cứng.

Thiết đặt cấu hình LILO: có thể lựa chọn cài đặt LILO vào Master Boot Record hoặc Sector đầu tiên của phân vùng Root hoặc không lựa chọn cài LILO.

Xác định các Partition

Đầu tiên cần xác định các *điểm kích hoạt* (mount point) cho một hoặc nhiều partition.

Trong bảng partition có các thông tin sau:

- **Mount Point**: Xác định partition nào sẽ được kích hoạt khi Linux được cài đặt và chạy. Nếu partition tồn tại và có nhãn là ***not set*** thì ta xác định mount point bằng cách kích chuột vào nút Edit hoặc double - click trên partition.

Và hệ thống khuyên chúng ta nên tạo các partition theo cách sau:

Một ***swap partition*** (ít nhất 16MB) - dùng hỗ trợ bộ nhớ ảo. Nếu máy chúng ta có 16Mb Ram hoặc ít hơn thì bắt buộc chúng ta phải tạo swap partition. Thậm chí nếu có nhiều bộ nhớ hơn, chúng ta cũng nên tạo swap partition. Kích thước tối thiểu của swap partition = max |bộ nhớ Ram và 16Mb|.

Một ***boot partition*** (tối đa 16Mb) - chứa nhân của HĐH cùng với các file trong quá trình khởi động.

Một ***root partition*** (từ 500Mb - 1Gb) là nơi chứa thư mục gốc và tất cả các file (trừ các file ở trong boot partition). Với 500Mb cho phép cài theo kiểu Workstation và với 1Gb cho phép cài mọi thứ.

Device:

Hiện tên các device partition (Ví dụ: hda2 đại diện cho partition thứ 2 trên ổ cứng primary).

- **Request**: Cho biết không gian mà partition hiện có. Nếu muốn thay đổi kích thước thì chúng ta phải xoá partition đó và tạo lại bằng cách dùng nút "Add".
- **Actual**: Cho biết không gian mà partition đang sử dụng.
- **Type**: Cho biết kiểu của partition.

Và chúng ta có thể add, edit, và delete các partition bằng cách kích chuột vào các nút đó. Chức năng của từng nút là:

- **Add:** Dùng để tạo một partition mới, gồm có các thông tin sau:

Mount Point: gồm có các kiểu

/ :

/boot :

/usr :

/home :

/var :

/opt :

/tmp :

/usr/local :

Size (Megs): chọn kích thước của partition.

Grow to fill disk: nếu chọn thì partition này sẽ sử dụng toàn bộ vùng đĩa trống còn lại.

Partition type: gồm có các kiểu

Linux Swap: chọn kiểu này nếu chúng ta muốn tạo partition swap.

Linux Native : chọn kiểu này nếu chúng ta muốn tạo partition root.

Linux RAID :

DOS 16-bit < 32 :

DOS 16-bit > 32 :

Edit: Dùng để thay đổi mount point của partition.

Delete: Dùng để xoá partition.

Reset: Khôi phục lại những thay đổi.

Make RAID Device: Sử dụng Make Raid device chỉ khi đã có kinh nghiệm về RAID.

Drive Summaries: hiển thị thông tin về cấu hình đĩa.

Chọn Partition để Format

Gồm có các thông tin:

Partition muốn Format.

Lựa chọn “Check for bad blocks while formating”

Chọn “Check for bad blocks while formating” để tìm ra những bad bocks trên đĩa sau đó sẽ đánh dấu lại nhằm không ghi dữ liệu lên chúng nữa.

Chọn cấu hình LILO (Linux Loader)

Để chọn cấu hình LILO, phải không đặt dấu kiểm “Do not install LILO”. Nếu ổ Linux Native có tên là /dev/hda5 thì màn hình sẽ hiện ra cho phép cài đặt chương trình nạp Linux vào Master Boot Record (MBR) hoặc First Sector of boot partition (sector đầu tiên của phân vùng khởi động). Nói chung là nên chọn Master Boot Record để có thể khởi động từ nhiều ổ.

Nếu có các ổ đĩa SCSI hoặc đĩa cứng của hỗ trợ LBA thì cần phải đánh dấu kiểm vào mục “Use Linear Mode”.

Kernel Parameters: Các tham số sẽ dùng bất cứ khi nào nhân được khởi động.

Bảng ở dưới sẽ cho biết thông tin về các phân vùng: tên phân vùng, loại phân vùng, có phải là phân vùng khởi động không ?. cần chọn phân vùng khởi động là phân vùng có tên Linux (thường là phân vùng mà Redhat sẽ đặt mặc định và chúng ta không phải thay đổi gì). Nếu đã có phân vùng tên là ‘dos’, chính là phân vùng trước khi cài đặt thì sau này mỗi khi khởi động máy tính, chúng ta có thể chọn phân vùng này để khởi động một cách bình thường bằng cách gõ tên phân vùng đó tại dấu nhắc “LILO Boot” trong quá trình khởi động. Cho phép thay đổi tên của phân vùng tương ứng trong phần “Boot Label”.

Tuỳ chọn “Create boot disk”: chỉ cần chọn mục này nếu không cài LILO hoặc cài LILO nhưng không cài vào MBR. Chú ý là nếu không cài LILO thì bắt buộc phải chọn mục này để khởi động từ đĩa mềm.

Chọn múi giờ

Nếu cài trong chế độ đồ họa, màn hình mặc định sẽ hiện ra một bản đồ thế giới. Có thể thay đổi bản đồ theo các kiểu sau:

World (bản đồ thế giới)

North American (Bắc Mỹ)

South American (Nam Mỹ)

Pacific Rim (Châu úc)

Europe (Châu Âu)

Africa (Châu Phi)

Asia (Châu á)

Đối với Việt Nam ta thì chọn Asia/Saigon (trong đĩa CD cài đặt LinuxVN thì mục chọn là Asia/Hanoi).

Có thể chọn múi giờ theo kiểu UTC Offset (Universal Time Coordinated), tức là kiểu GMT +/- độ lệch. Nếu hệ thống giờ sở tại sử dụng UTC thì chọn “System clock uses UTC”.

Thiết đặt cấu hình Account (người sử dụng)

Như đã biết Linux kế thừa từ Unix, do đó nó cũng hỗ trợ chế độ đa người dùng như Unix. Bản cài đặt Redhat cho phép đặt luôn cấu hình người dùng ngay trong quá trình cài đặt.

Đầu tiên chúng ta cần đặt mật khẩu cho người dùng Root trong phần "Root Password" và xác nhận lại mật khẩu này trong mục "Confirm". Chú ý là mặc định tất cả các mật khẩu đều phải tối thiểu 6 ký tự.

Sau đó ta có thể thêm ngay một số người dùng đầu tiên: Tên đặt trong "Account Name", gõ password trong mục "Password" và xác nhận mật khẩu trong mục Password (confirm) ở ngay bên cạnh, tên đầy đủ của người dùng trong mục "Full Name". Sau đó gõ phím "Add" thể thêm người dùng mới vào danh sách các người dùng. Phím "Edit" cho phép hiển thị người dùng hiện hành trong bảng danh sách người dùng, phím "Delete" để xoá người dùng hiện thời.

Thiết đặt cấu hình quyền hạn (Authentication Configuration)

Thực hiện các mục sau:

Enable MD5 Password: cách mã hoá này cho phép mật khẩu dài tới 256 ký tự.

Shadow Password: đây là cách bảo mật tối đa cho mật khẩu, file chứa mật khẩu /etc/passwd sẽ được thay bằng etc/shadow và file này chỉ được phép hiển thị bởi người dùng Root. Chọn cả 2 mục này sẽ tăng tính bảo mật của hệ thống.

Enable NIS: chọn mục này nếu máy tính kết nối vào mạng NIS (Network Information System) cho phép một nhóm máy tính trong vùng Network Information Service với cùng một password.

NIS domain: tên của domain hoặc nhóm máy tính chứa hệ thống.

NIS server: cho phép máy tính dùng một NIS server rêng, hơn là một thông điệp rộng rãi trong mạng LAN.

Lựa chọn các gói phần mềm cài đặt (Pakage Selection)

Có rất nhiều mục để chọn sẽ nói chi tiết ở phần sau như: Printer Support, hệ thống X Window, GNome, KDE, DOS / Window Connectivity,.... Lựa chọn "Everything" sẽ cài đầy đủ tất cả mọi thứ, do đó cần phải lựa chọn các gói phần mềm phù hợp nếu ổ Linux Native / không đủ.

Các packages có thể được chọn cài đặt bao gồm:

Printer Support: Nếu được cài Linux sẽ cố gắng nhận máy in hoặc cho phép người dùng thiết lập các thông số về máy in của hệ thống.

X Window System: Môi trường đồ họa nguyên thuỷ trong Linux, được gọi tắt là X. X không phải là một giao diện đồ họa người dùng thực sự mà chỉ là một hệ cửa sổ cùng với các công cụ để một giao diện đồ họa người dùng có thể được xây dựng từ đó.

GNOME: Là một giao diện đồ họa người dùng đẹp hơn, tiện lợi và thân thiện hơn X, GNOME cũng cung cấp một giao diện lập trình ở mức cao hơn để tạo ra các ứng dụng với giao diện kiểu GNOME.

KDE: KDE là một giao diện đồ họa người dùng được phát triển dựa trên thư viện giao diện đồ họa C++ Qt. Thư viện giao diện này hỗ trợ UNIX, Windows và cả Mac. Do được phát triển trước GNOME nên KDE có nhiều điểm tốt hơn.

Mail/ WWW/ News tools: Bộ công cụ dùng để gửi, nhận thư tín điện tử, duyệt Web và đọc các bản tin.

Dos/ Windows Connectivity: Bộ giả lập DOS và Windows cho phép người dùng chạy các ứng dụng DOS và Windows ngay trong Linux.

Graphic Manupulation: Các tiện ích đồ họa như Gview, Imgedit... cho phép trình diễn và xử lý các file hình ảnh.

Games: Một số chương trình trò chơi giải trí trong chế độ text hoặc X Window.

Multimedia Support: Các chương trình hỗ trợ đa phương tiện, như nghe midi, wave, mp3... điều khiển joystick, thu âm ...

Dialup WorkStation: Cho phép người dùng sử dụng modem và quay số để sử dụng các dịch vụ qua đường điện thoại như gửi nhận mail, web...

News Server: Cung cấp dịch vụ máy chủ news, cho phép máy tính trở thành một server bản tin.

NFS Server: Máy chủ hệ thống file mạng NFS (Network File System).

SMB Server: Máy chủ hệ thống file mạng Samba.

IPX/ Netware Connectivity: Giúp hệ thống chạy Linux giao tiếp với các máy tính khác qua mạng Netware sử dụng giao thức IPX.

Anonymous FTP Server: Cài đặt package này giúp máy tính trở thành một máy chủ FTP (File Transfer Protocol), có thể cung cấp dịch vụ truyền file cho các máy khác trong mạng.

Web Server: Cài đặt package này giúp máy tính trở thành một máy chủ Web, có thể cung cấp dịch vụ web cho các máy khác trong mạng.

DNS Name Server: Cài đặt package này giúp máy tính trở thành một máy chủ DNS, có thể cung cấp dịch vụ đặt tên vùng cho các máy khác trong mạng.

Postgres (SQL) Server: Cài đặt package này giúp máy tính trở thành một máy chủ SQL, có thể cung cấp dịch vụ truy vấn dữ liệu cho các máy khác trong mạng.

Network Management Workstation: Giúp máy tính trạm làm việc điều hành trở thành một máy chủ SQL, có thể cung cấp dịch vụ truy vấn dữ liệu cho các máy khác trong mạng.

TeX Document Formatting: Hệ soạn thảo và định dạng văn bản dưới dạng Tex.

Emacs: Hệ soản thảo văn bản đơn giản.

Development: Các bộ biên dịch, gỡ rối, công cụ phát triển phần mềm ... dưới các ngôn ngữ như Perl, C, C++. Mã nguồn của các chương trình trong Linux.

Kernel Development: Mã nguồn nhân Linux và bộ công cụ phát triển dành cho phát triển nhân Linux.

Extra Documentation: Mọi tài liệu về Linux có trong đĩa CD, dưới những ngôn ngữ như Anh, Pháp, Italia, Tây Ban Nha...

Utilities: Các tiện ích cho Linux.

Thiết đặt cấu hình X (X Configuration)

Phần này sẽ đặt cấu hình card màn hình để thể hiện khi chúng ta sử dụng các hệ thống X Window. Ví dụ sau đây là một số thông số hiển thị khi chúng ta cài đặt:

Video Card: ATI Mach64

Video Ram: 4096KB

X server: Mach64

Monitor: 14" COLOR

Độ quét ngang: 30 - 54 Khz

Độ quét dọc: 50 - 120 Hz

Đây là phần khá quan trọng, nếu card màn hình của không phải là dạng chuẩn thì sẽ phải cài bằng tay, một công việc khá mệt mỏi.

Test this Configuration: ấn vào đây để kiểm tra tính năng đồ hoạ có hoạt động tốt không, nếu màn hình của qua khỏi cuộc kiểm tra này thì có thể coi là đã thành công tới 90% quá trình cài đặt Redhat.

Customize X Configuration: Nếu chọn mục này thì ghi gõ Next, Redhat sẽ cho một bảng cho thiết lập bằng tay các chế độ đồ họa 8 bits, 16 bits, 32 bits; các chế độ phân giải 640x480, 800x600, 1024x786, ... Chú ý là khi chọn độ phân giải nào thì luôn luôn gõ phím "Test this Configuration" để đảm bảo màn hình của luôn hiển thị đúng.

Bắt đầu quá trình copy từ đĩa CD vào ổ cứng

Bước cuối cùng này sẽ thực hiện các bước format ổ Linux Native / (nếu đã chọn ở phần trước), format ổ Linux Swap, và sau đó là giải nén tất cả các gói phần mềm mà đã lựa chọn trong bước thứ 13 vào ổ cứng. Quá trình này mất khoảng 10 phút đến 20 phút tuỳ theo số lượng các gói chọn.

Kết thúc quá trình này, gõ "Exit", khởi động lại máy, nhớ tháo đĩa CD Redhat ra khỏi ổ để bắt đầu khởi động từ ổ cứng.

AA.4. Các hạn chế về phần cứng đối với Linux

Các bộ vi xử lý mà Linux hỗ trợ

Linux chủ yếu chạy trên các máy PC thế hệ 386, 486, 586 sử dụng các phần cứng họ vi xử lý 80386. Việc cài đặt trên các phần cứng khác thì vẫn đang ở trong giai đoạn thiết kế.

Có thể chạy thử Linux bằng một máy với phần cứng tối thiểu là: bộ vi xử lý Intel 386, 486, 586, 4MB RAM và một ổ mềm. Dĩ nhiên là càng nhiều RAM thì càng tốt.

Linux hỗ trợ VESA Local Bus và PCI.

Linux cũng hỗ trợ phần lớn cho các ổ cứng chuẩn ESDI và MCA (bus độc quyền của IBM).

Linux cũng có thể chạy trên các laptop họ 386.

Có một cách cài đặt Linux trên 8086 được biết đến dưới tên gọi ELKS (Embeddable Linux Kernel Subset). Đây là một nhân Linux 16 bit được chủ yếu sử dụng trong các hệ thống nhúng. Thực ra phiên bản Linux hiện nay không thể chạy được đầy đủ trên 8086 hay 286 bởi vì những bộ vi xử lý này không hỗ trợ cho việc chuyển đổi tác vụ cũng như quản lý bộ nhớ.

Linux hỗ trợ đa quá trình cho các kiến trúc Intel MP.

Dưới đây là danh sách các bộ VXL mà Linux hỗ trợ:

- Dòng 68000 của Amigas và Ataris hiện đang được triển khai nghiên cứu.
- Các phiên bản GNU/Linux cũng được thử cài đặt cho các nền Alpha, Sparc, PowerPC, ARM.
- Một dự án về Linux trên PPC cũng đã được tiến hành.
- Apple đã hỗ trợ MkLinux trên các Power Macs dựa trên OSF của Mach vi nhân.
- Linux cho máy 64 bit DEC Alpha/AXP.
- Đang nghiên cứu về Linux cho MIPS, bắt đầu đối với R4 trên các máy Deskstation Type.
- Hiện có 2 bản Linux cho dòng máy dùng họ vi xử lý ARM. Một là của vi xử lý ARM3 trên các máy Acorn A5000 và nó còn bao gồm cả các thiết bị I/O cho 82710. Còn lại là cho họ vi xử lý ARM610 của máy Acorn RISC PC.
- Linux cho SPARC đang được tiến hành.
- Có bản ***Hardhat*** cho các máy SGI/Indy.

Các yêu cầu về không gian ổ cứng

Đối việc cài đặt tối thiểu là 10 MB, chủ yếu là để thử chứ không có nhiều các tính năng khác.

Ta có thể cài đặt thêm X với khoảng 80 MB. Nếu cài đặt cả bộ GNU/Linux sẽ cần khoảng 500MB-1GB bao gồm cả mã nguồn và nhiều thứ khác nữa.

Các yêu cầu về bộ nhớ

Tối thiểu là 4MB. Ta có thể sử dụng swapping để chạy thêm các cài đặt khác. Linux nói chung là chạy tương đối "thoải mái" với 4MB Ram nhưng các ứng dụng X Windows sẽ chạy chậm bởi vì chúng cần phải thực việc *swap* vào ra trên đĩa.

Một vài ứng dụng hiện tại lại chỉ chạy bình thường với 128MB bộ nhớ vật lý chẳng hạn như Netscape.

Sự tương thích với các hệ điều hành khác: DOS, OS/2, 386BSD, Win95

Linux sử dụng sắp xếp phân dạng giống như MS-DOS do đó nó có thể chia sẻ đĩa với các hệ điều hành khác. Tuy vậy, điều này cũng có nghĩa là các hệ điều hành khác cũng có thể không hẳn là hoàn toàn tương thích. Các trình *Fdisk* và *Format* của DOS thỉnh thoảng lại có thể viết đè lên dữ liệu trong phân vùng của Linux bởi vì chúng có thể sử dụng các thông tin phân vùng sai lệnh từ boot sector của phân vùng chứ không phải là từ bảng phân vùng. Để tránh hiện tượng này, một ý tưởng là đưa về 0 địa chỉ bắt đầu của một phân vùng

vừa mới tạo lập trong Linux trước khi sử dụng các lệnh format của MS-DOS. Sử dụng lệnh sau:

$ dd if=/dev/zero of=/dev/hdXY bs=512 count=1

Với hdXY là phân vùng liên quan, chẳng hạn /dev/hda1 là phân vùng đầu tiên trên đĩa IDE đầu tiên.

Linux có thể đọc và ghi các file trên các phân vùng FAT của DOS và OS/2 và các đĩa mềm bằng cách sử dụng hệ thống file DOS được tích hợp vào nhân hoặc các công cụ **mtool**. Nhân cũng cung cấp hỗ trợ cho hệ thống file VFAT của Windows 9x và Windows NT. Hiện tại các đĩa phân vùng theo NTFS cũng đang được nghiên cứu hỗ trợ cùng với việc hỗ trợ nén đĩa như là một tính năng chuẩn.

Linux cũng có thể truy cập được tới hệ thống file HPFS của OS/2 nhưng chỉ ở chế độ *read-only*. Người ta có thể thực hiện điều này như một lựa chọn khi biên dịch nhân.

Linux cũng hỗ trợ cho việc thao tác trên các định dạng AFFS (Amiga Fast File System) từ bản 1.3 trở về sau bằng cách như một lựa chọn lúc biên dịch hay như một mô đun riêng. Tuy vậy, điều này cũng chỉ dừng ở mức độ chỉ đọc. Các truy cập đĩa mềm thì chưa có hỗ trợ bởi vì sự khác biệt giữa các điều khiển đĩa của PC và Amiga.

Đối với các máy chạy các hệ điều hành của Unix như BSD, System V... thì các nhân hiện tại cũng mới chỉ có thể đọc hệ thống file UFS trên System V, Xenix, BSD, một số sản phẩm thừa kế khác như SunOS, FreeBSD, NetBSD, NeXTStep. Hỗ trợ UFS cũng được coi như một lựa chọn lúc biên dịch nhân hay như một mô đun.

Linux cho phép đọc/viết trên các ổ đĩa SMB của các nhóm Windows và WinNT. Có một chương trình tên là Samba cho phép truy cập và hệ thống file mạng WfW (miễn là dùng giao thức TCP/IP) .

Đối với các máy Macintosh thì có một tập hợp các chương trình ở cấp độ người dùng có thể đọc, ghi trên HFS (Macintosh Hierarchical File System).

Câu hỏi đặt ra là có thể chạy một chương trình Windows trong Linux hay không ? Chương trình tên WINE đang được phát triển để mô phỏng môi trường Windows trong Linux. Hiện tại khi muốn dùng hai hệ điều hành cùng lúc với Linux thì ta đã có chương trình LILO boot. LILO boot bắt buộc ta phải lựa chọn hệ điều hành vào lúc khởi động. Ngoài ra, còn có một chương trình tên LOADLIN là một chương trình DOS cho phép nạp Linux (cũng như bất kỳ hệ điều hành khác) khiến cho Linux cùng tồn tại với DOS. LOADLIN đặc biệt hữu dụng khi ta muốn cài Linux trên các ổ đĩa thứ 3, 4 của hệ thống (hoặc khi ta thêm một ổ SCSI vào một hệ thống có chứa ổ IDE). Trong trường hợp này thì LILO boot sẽ không có khả năng tìm kiếm và nạp nhân. Do đó ta sẽ phải tạo một thư mục chẳng hạn C:ữLINUX, đặt LOADLIN vào trong đó cùng với một bản copy của nhân và rồi sử dụng nó.

Chú ý: Cần tạo ít nhất một phân vùng Linux dưới giới hạn 1024 cylinder logic.

PHỤ LỤC B. TRÌNH SOẠN THẢO VIM

UNIX có hai bộ soạn thảo là **ed** và **vi** trong đó **vi** được ưa chuộng hơn do **vi** được phát triển từ bộ soạn thảo dòng lệnh **ed**. Trong chế độ văn bản, Linux cho phép người dùng sử dụng trình soạn thảo **vim** mà **vim** chính là bộ soạn thảo tương thích với **vi**. **vim** được phần lớn người dùng sử dụng để soạn thảo các file văn bản ASCII, đặc biệt là tạo ra các văn bản chương trình nguồn.

vim có sáu chế độ cơ bản:

- Chế độ thường (**Normal mode**): trong chế độ thường người dùng được phép nhập tất cả các lệnh soạn thảo thông thường. Nếu không thiết lập tuỳ chọn **insertmode**, ngầm định vào ngay chế độ thường khi khởi động **vim**. Chế độ thường còn được gọi là ***chế độ lệnh***.
- Chế độ ảo (**Visual mode**): chế độ này cũng gần giống như chế độ thường, chỉ khác ở chỗ là lệnh di chuyển có tác dụng đánh dấu văn bản. Mặt khác, các lệnh khác (không là lệnh di chuyển) thực sự tác dụng trong phạm vi những đoạn văn bản đã được đánh dấu.
- Chế độ chọn lựa (**Select mode**): chế độ này tương tự như chế độ lựa chọn của MS-Windows. Người dùng có thể nhập một ký tự thuộc loại in ấn được để xoá một sự lựa chọn và chạy chế độ chèn.
- Chế độ chèn (**Insert mode**): Trong chế độ này, có thể soạn thảo văn bản bình thường như các bộ soạn thảo quen biết khác. Văn bản đó sẽ được chèn vào trong bộ đệm.
- Chế độ dòng lệnh (**Command-line mode** hay **cmdline mode**): Trong chế độ này, một dòng lệnh được nhập tại đáy cửa sổ soạn thảo. Đó có thể là các lệnh **Ex** (**:**), các lệnh tìm kiếm (**/** hay **?**), và các lệnh lọc (**!**).
- Chế độ **Ex** (**Ex mode**): giống như chế độ dòng lệnh, nhưng sau khi nhập một lệnh, vẫn ở trong chế độ **Ex**. Tuy nhiên còn rất nhiều hạn chế đối với các lệnh ở chế độ này.

Ngoài ra còn có năm chế độ phụ sau:

- Chế độ chờ thực hiện (**Operator-pending mode**): chế độ này giống chế độ thường, nhưng sau khi gọi một lệnh, **vim** sẽ chờ cho đến khi đoạn văn bản chịu tác động của lệnh được đưa ra.
- Chế độ thay thế (**Replace mode**): chế độ thay thế là một trường hợp đặc biệt của chế độ chèn. Người dùng có thể nhập mọi ký tự như trong chế độ chèn, chỉ khác ở chỗ: mỗi ký tự nhập sẽ thay thế cho một ký tự đã tồn tại (có thể gọi là chế độ đè - overwrite).
- Chế độ chèn-lệnh (**Insert Normal mode**): gõ CTRL-O trong chế độ chèn để chuyển sang chế độ chèn-lệnh. Chế độ này cũng giống như chế độ thường, nhưng sau khi thực hiện một lệnh, **vim** sẽ trở lại chế độ chèn.
- Chế độ chèn-ảo (**Insert Visual mode**): chế độ này được sinh ra khi trong chế độ chèn thực hiện một sự lựa chọn ảo. **vim** sẽ trở về chế độ chèn sau khi sự lựa chọn ảo đó kết thúc.

- Chế độ chèn-lựa chọn (**Insert Select mode**): chế độ này được khởi tạo khi chạy chế độ lựa chọn trong chế độ chèn. Khi chế độ lựa chọn kết thúc, **vim** sẽ trở về chế độ chèn.

Việc chuyển đổi giữa các chế độ trong **vim** được thực hiện nhờ các ***lệnh*** (phím lệnh hoặc xâu lệnh) của **vim** và được tập hợp trong bảng dưới đây. Trong bảng này, cột đầu tiên là chế độ nguồn, hàng đầu tiên là chế độ đích, ô giao giữa hàng và cột chứa các phím lệnh chuyển chế độ (ký hiệu ***1, *2, *3, *4, *5, *6** là cách viết tắt danh sách lệnh được giải thích ở sau):

Chế độ hiện thời	***Chế độ cần chuyển tới***						
	Thường	**ảo**	**Lựa chọn**	**Chèn**	**Thay thế**	**Dòng lệnh**	**Ex**
Thường		v, V, ^V	***4**	***1**	R	:, /, ?, !	Q
ảo	***2**		^G	c, C	--	**:**	--
Lựa chọn	***5**	^O, ^G		***6**	--	**:**	--
Chèn	<Esc>	--	--		<Insert>	--	--
Thay thế	<Esc>	--	--	<Insert>		--	--
Dòng lệnh	***3**	--	--	:start	--		--
Ex	:vi	--	--	--	--	--	

Giải thích các lệnh viết tắt:

- ***1** Để chuyển sang chế độ chèn từ chế độ thường, sử dụng một trong các phím: i, I, a, A, o, O, c, C, s, S.
- ***2** Để chuyển sang chế độ thường từ chế độ ảo: ngoài <Esc>, v, V, CTRL-V có thể gõ một phím lệnh thông thường (ngoại trừ phím lệnh di chuyển con trỏ).
- ***3** Để chuyển sang chế độ thường từ chế độ dòng lệnh:
  - ❖ Thực hiện lệnh <Enter>
  - ❖ Gõ CTRL-C hoặc <Esc>
- ***4** Để chuyển sang chế độ lựa chọn từ chế độ thường:
  - ❖ Sử dụng chuột để lựa chọn văn bản
  - ❖ Sử dụng các phím không in được để di chuyển dấu nhắc trỏ trong khi ấn giữ phím SHIFT
- ***5** Để chuyển sang chế độ thường từ chế độ lựa chọn: sử dụng các phím không in được để di chuyển dấu nhắc trỏ mà không nhấn phím SHIFT.
- ***6** Để chuyển sang chế độ chèn từ chế độ lựa chọn: nhập một ký tự có thể in được.

Dưới đây trình bày nội dung một số các lệnh cơ bản trong **vim**.

B.1 Khởi động vim

B.1.1 Mở chương trình soạn thảo vim

Cách đơn giản nhất bắt đầu dùng **vim** để soạn thảo một file văn bản, là gõ một trong ba lệnh sau:

vim [**tuỳ-chọn**]	bắt đầu soạn thảo hay hiệu chỉnh một file
vim [**tuỳ-chọn]** <danh sách các file>	bắt đầu soạn thảo một hoặc nhiều file
vim [tuỳ chọn] **-**	soạn thảo một file từ thiết bị vào chuẩn

Nếu tham số danh sách các file không có thì **vim** sẽ thao tác với một file mới (vùng đệm soạn thảo rỗng). Ngược lại, file đầu tiên trong danh sách trở thành file hiện hành và được đọc vào trong vùng soạn thảo. Con trỏ sẽ xuất hiện ở đầu dòng đầu tiên của vùng này. Để hướng đến file kế tiếp, ta đánh lệnh "**:next**" ở chế độ lệnh. Để soạn thảo một file có tên bắt đầu bằng "**-**" thì phải điền vào tên file dấu "**--**".

Ví dụ:

```
# vim vdvim`
~       
~       
~
~
~
~
~
"vdvim"[New File] 0,0-1 All
```

Lệnh trên mở một cửa sổ cho người dùng soạn thảo một file mới có tên là **"vdvim"**

Một số các tuỳ chọn cơ bản:

+ [n]	đặt dấu nhắc trỏ tại dòng thứ n (ngầm định là dòng cuối)
+ <lệnh>	thực hiện lệnh sau khi nạp file
+/<mẫu> <file>	đặt dấu nhắc trỏ tại dòng đầu tiên có chứa mẫu trong file
-o[n]	mở n cửa sổ (ngầm định có một cửa sổ cho một file: n=1)
--help	hiển thị danh sách các tham số và thoát

B.1.2. Tính năng mở nhiều cửa sổ

Trong **vim**, có thể chia cửa sổ soạn thảo hiện thời thành nhiều phần hay mở nhiều cửa sổ cùng lúc để soạn thảo các file khác nhau.

Ví dụ lệnh sau sẽ mở hai file **vd1** và **vd2** trên hai cửa sổ soạn thảo:

```
# vim -o2 vd1 vd2
~       
~
~
vd1 0,0-1 All
~
```

~
~
vd2 0,0-1 All
"vd2" [New File]

Sau đây là một số các lệnh hay dùng:

CTRL-W	chia cửa sổ hiện tại thành hai phần
:split <file>	chia cửa sổ và soạn thảo <file> trên một phần chia của cửa sổ
:sf <file>	chia cửa sổ, tìm file trên đường dẫn và soạn thảo nó
CTRL-W CTRL-^	chia cửa sổ và edit alternate file
CTRL-W **n**	tạo một cửa sổ trống mới (giống :new)
CTRL-W **q**	dừng việc soạn thảo và đóng cửa sổ (giống **:q**)
CTRL-W **o**	phóng to cửa sổ hiện hành trên toàn màn hình
CTRL-W **j**	di chuyển trỏ soạn thảo xuống cửa sổ dưới
CTRL-W **k**	di chuyển trỏ soạn thảo lên cửa sổ trên
CTRL-W **t**	di chuyển trỏ soạn thảo lên đỉnh cửa sổ
CTRL-W **b**	di chuyển trỏ soạn thảo xuống đáy cửa sổ
CTRL-W **p**	di chuyển trỏ soạn thảo đến cửa sổ được kích hoạt lúc trước
CTRL-W **x**	di chuyển trỏ soạn thảo đến cửa sổ tiếp theo
CTRL-W **=**	tạo tất cả các cửa sổ có chiều cao như nhau
CTRL-W **-**	giảm chiều cao của cửa sổ hiện thời
CTRL-W **+**	tăng chiều cao của cửa sổ hiện thời
CTRL-W **Ỵ**	thiết đặt chiều cao của cửa sổ hiện thời

B.1.3. Ghi và thoát trong vim

Bảng dưới đây giới thiệu các lệnh để ghi nội dung file lên hệ thống file và thoát khỏi **vim** sau khi đã soạn thảo xong nội dung của file (tham số ửn, mứ nếu có mang ý nghĩa "*từ dòng n tới dòng m")*.

:[n,m] w [!]	ghi file hiện thời.
:[n,m] w <file>	ghi nội dung ra <file>, trừ khi file đó đã thực sự tồn tại
:[n,m] w! <file>	ghi nội dung ra <file>, nếu file đã tồn tại thì ghi đè lên nội dung cũ
:[n,m] w[!] >> [<file>]	chèn thêm vào <file>, nếu không có file, mặc định là file hiện thời
:[n,m] w !<lệnh>	thực hiện <lệnh> trên các dòng từ dòng thứ n đến dòng thứ m như thiết bị vào chuẩn
:[n,m] up [thời gian] [!]	ghi file hiện thời nếu nó được sửa đổi
:q [!]	thoát khỏi **vim**
:wq [!] [<file>]	ghi nội dung <file> (mặc định là file hiện thời) và thoát khỏi **vim**
:x [!] <file>	giống :wq nhưng chỉ ghi khi thực sự có sự thay đổi trong nội dung file (giống ZZ)
:st [!]	dừng **vim** và khởi tạo một shell (giống CTRL-Z)

B.2. Di chuyển trỏ soạn thảo trong Vim

B.2.1. Di chuyển trong văn bản

Di chuyển trỏ soạn thảo trong văn bản là một tính năng rất quan trọng trong một trình soạn thảo văn bản **vim**. Dưới đây là một số các lệnh để thực hiện việc trên (cột đầu tiên có n chỉ một số là số lượng):

N	**l**	di chuyển trỏ soạn thảo về bên phải n ký tự
N	**h**	di chuyển trỏ soạn thảo về bên trái n ký tự
n	**k**	di chuyển trỏ soạn thảo lên n dòng
n	**j**	di chuyển trỏ soạn thảo xuống n dòng
	0	di chuyển về đầu dòng
	^	di chuyển đến từ đầu tiên của dòng hiện tại
	$	di chuyển đến cuối dòng
	<Enter>	di chuyển đến đầu dòng tiếp theo
n	-	di chuyển đến đầu dòng trước dòng hiện tại n dòng
n	+	di chuyển đến đầu dòng sau dòng hiện tại n dòng
n	_	di chuyển đến đầu dòng sau dòng hiện tại n-1 dòng
	G	di chuyển đến dòng cuối cùng trong file
n	G	di chuyển đến dòng thứ n trong file (giống **:n**)
	H	di chuyển đến dòng đầu tiên trên màn hình
	M	di chuyển đến dòng ở giữa màn hình
n	gg	di chuyển đến đầu dòng thứ n (mặc định là dòng đầu tiên)
n	gk	di chuyển lên n dòng màn hình
n	gj	di chuyển xuống n dòng màn hình

B.2.2. Di chuyển theo các đối tượng văn bản

vim cung cấp các lệnh dưới đây cho phép di chuyển trỏ soạn thảo nhanh theo các đối tượng văn bản và điều đó tạo nhiều thuận tiện khi biên tập, chẳng hạn, trong các trường hợp người dùng cần xoá bỏ hay thay đổi một từ, một câu ...

N	W	di chuyển n từ tiếp theo
N	E	di chuyển đến cuối của từ thứ n
N	B	di chuyển ngược lại n từ
N	ge	di chuyển ngược lại n từ và đặt dấu nhắc trỏ tại chữ cái cuối từ
N	>	di chuyển đến n câu tiếp theo
N	<	di chuyển ngược lại n câu
N	|	di chuyển đến n đoạn tiếp theo
N	|	di chuyển ngược lại n đoạn
N	]]	di chuyển đến n phần tiếp theo và đặt dấu nhắc trỏ tại đầu phần
N	[[	di chuyển ngược lại n phần và đặt dấu nhắc trỏ tại đầu phần
n	][	di chuyển đến n phần tiếp theo và đặt dấu nhắc trỏ tại cuối phần
n	[]	di chuyển ngược lại n phần và đặt dấu nhắc trỏ tại cuối phần

B.2.3. Cuộn màn hình

Màn hình sẽ tự động cuộn khi di trỏ soạn thảo đến đáy hoặc lên đỉnh màn hình. Tuy nhiên các lệnh sau đây giúp người dùng cuộn màn hình theo ý muốn:

N	<CTRL-f>	cuộn lên n màn hình (mặc định là 1 màn hình)
N	<CTRL-b>	cuộn xuống n màn hình (mặc định là 1 màn hình)
N	<CTRL-d>	cuộn xuống n dòng (mặc định là 1/2 màn hình)
N	<CTRL-u>	cuộn lên n dòng (mặc định là 1/2 màn hình)
N	<CTRL-e>	cuộn xuống n dòng (mặc định là 1 dòng)
N	<CTRL-y>	cuộn lên n dòng (mặc định là 1 dòng)
	z<Enter>	vẽ lại cửa sổ soạn thảo, dòng hiện tại sẽ là dòng trên cùng của cửa sổ (giống zt)
	z.	vẽ lại cửa sổ soạn thảo, dòng hiện tại sẽ là dòng ở giữa của cửa sổ (giống zz)
	z-	vẽ lại cửa sổ soạn thảo, dòng hiện tại sẽ là dòng ở đáy của cửa sổ (giống zb)

B.3. Các thao tác trong văn bản

vim có rất nhiều các lệnh hỗ trợ thao tác soạn thảo hay hiệu chỉnh một file. Phần dưới đây giới thiệu chi tiết về các cách để thêm văn bản, hiệu chỉnh văn bản hay xoá một văn bản.

Khi soạn thảo văn bản, nhiều dòng có thể được nhập bằng cách sử dụng phím **Enter**. Nếu có một lỗi cần phải sửa, có thể sử dụng các phím mũi tên để di chuyển trỏ soạn thảo trong văn bản và sử dụng các phím **Backspace** hoặc **Delete** để hiệu chỉnh.

B.3.1. Các lệnh chèn văn bản trong vim

	A	chèn văn bản vào vị trí dấu nhắc trỏ hiện thời (n lần)
N	A	chèn văn bản vào cuối một dòng (n lần)
n	i	chèn văn bản vào bên trái dấu nhắc trỏ (n lần)
n	I	chèn văn bản vào bên trái ký tự đầu tiên khác trống trên dòng hiện tại (n lần)
n	gI	chèn văn bản vào cột đầu tiên (n lần)
n	o	chèn n dòng trống vào dưới dòng hiện tại
n	O	chèn n dòng trống vào trên dòng hiện tại
	:r file	chèn vào vị trí con trỏ nội dung của file
	:r! lệnh	chèn vào vị trí con trỏ kết quả của lệnh lệnh

B.3.2. Các lệnh xoá văn bản trong vim

Bên cạnh các lệnh tạo hay chèn văn bản, **vim** cũng có một số lệnh cho phép người dùng có thể xoá văn bản. Dưới đây là bảng liệt kê một số lệnh cơ bản:

N	x	xoá n ký tự bên phải dấu nhắc trỏ
N	X	xoá n ký tự bên trái dấu nhắc trỏ
N	dd	xoá n dòng kể từ dòng hiện thời

	D hoặc d$	xoá từ vị trí hiện thời đến hết dòng
N	dw	xoá n từ kể từ vị trí hiện thời
	dG	xoá từ vị trí hiện thời đến cuối file
	d1G	xoá ngược từ vị trí hiện thời đến đầu file
	dn$	xoá từ dòng hiện thời đến hết dòng thứ n
N,m	d	xoá từ dòng thứ n đến dòng thứ m
N	cc	xoá n dòng, kể cả dòng hiện thời rồi khởi tạo chế độ chèn (Insert)
N	C	xoá n dòng kể từ vị trí hiện thời rồi khởi tạo chế độ chèn (Insert)
	cn$	xoá từ dòng hiện thời đến hết dòng thứ n rồi khởi tạo chế độ chèn (Insert)
N	s	xoá n ký tự và chạy chế độ chèn (Insert)
N	S	xoá n dòng và chạy chế độ chèn (Insert)

B.3.3. Các lệnh khôi phục văn bản trong vim

Các lệnh sau cho phép khôi phục lại văn bản sau một thao tác hiệu chỉnh nào đó:

N	u	khôi phục lại văn bản như trước khi thực hiện n lần thay đổi
	U	khôi phục lại hoàn toàn dòng văn bản hiện thời như trước khi thực hiện bất kỳ sự hiệu chỉnh nào trên dòng đó
	: e!	hiệu chỉnh lại. Lưu trữ trạng thái của lần ghi trước
N	CTRL-R	làm lại (redo) n lần khôi phục (undo) trước đó !

6.3.4. Các lệnh thay thế văn bản trong vim

vim còn có các lệnh cho phép thay đổi văn bản mà không cần phải xoá văn bản rồi sau đó đánh mới.

n	r <ký tự>	thay thế n ký tự bên phải dấu trỏ bởi <ký tự>
	R	ghi đè văn bản bởi một văn bản mới (hay chuyển sang chế độ thay thế - Replace trong Vim)
n	~	chuyển n chữ hoa thành chữ thường và ngược lại
n	gUU	chuyển các ký tự trên n dòng, kể từ dòng hiện tại, từ chữ thường thành chữ hoa
n	guu	chuyển các ký tự trên n dòng, kể từ dòng hiện tại, từ chữ hoa thành chữ thường
n	CTRL-A	cộng thêm n đơn vị vào số hiện có
n	CTRL-X	bớt đi n đơn vị từ số hiện có
n	> [> ...]	chuyển dòng thứ n sang bên phải x khoảng trống (giống như phím TAB trong Win), nếu không có n mặc định là dòng hiện tại, x là số dấu ' > ' (ví dụ: >>> thì x bằng 3)
n	< [< ...]	chuyển dòng thứ n sang bên trái x khoảng trống (giống như phím SHIFT+TAB trong Win), nếu không có n mặc định là dòng

		hiện tại, x là số dấu ' < '
n	J	kết hợp n dòng, kể từ dòng hiện tại, thành một dòng
n	gJ	giống như J nhưng không chèn các khoảng trống
	:[n,m] ce [width]	căn giữa từ dòng thứ n đến dòng thứ m với độ rộng là width, nếu không có width, mặc định độ rộng là 80
	:[n,m] ri [width]	căn phải từ dòng thứ n đến dòng thứ m với độ rộng là width, nếu không có width, mặc định độ rộng là 80
	:[n,m] le [width]	căn trái từ dòng thứ n đến dòng thứ m với độ rộng là width, nếu không có width, mặc định độ rộng là 80
	:[n,m]s/<mẫu1>/<mẫu2>/[g][c]	tìm từ dòng thứ n đến dòng thứ m và thay thế mẫu1 bởi mẫu2. Với [g], thay thế cho mọi mẫu tìm được. Với [c], yêu cầu xác nhận đối với mỗi mẫu tìm được
	:[n,m]s[g][c]	lặp lại lệnh tìm và thay thế trước (:s) với phạm Vim mới từ dòng n đến dòng m kèm theo là các tuỳ chọn
	&	lặp lại việc tìm kiếm và thay thế trên dòng hiện thời mà không có các tuỳ chọn

B.3.5. Sao chép và di chuyển văn bản trong vim

Phần này giới thiệu với các các lệnh cơ bản để cắt và dán văn bản trong **vim**.

Để sao chép văn bản phải thực hiện ba bước sau:

- Sao chép văn bản vào một bộ nhớ đệm (**`Yanking`**)
- Di chuyển dấu nhắc trỏ đến vị trí cần sao chép (**Moving**)
- Dán văn bản (**Pasting**)

Sau đây là các lệnh cụ thể của từng bước:

*** <u>Sao chép văn bản vào bộ nhớ đệm</u>**

n	yw	sao chép n ký tự
n	Y	sao chép n dòng văn bản, kể từ dòng hiện tại, vào bộ nhớ đệm (giống yy)
	: [n] co [m]	sao chép dòng thứ n vào dưới dòng thứ m

*** <u>Dán văn bản:</u>**

n	P	dán đoạn văn bản được sao chép vào bên phải vị trí hiện thời (n lần)
n	P	dán n đoạn văn bản được sao chép vào bên trái vị trí hiện thời (n lần)
n	Gp	giống như p, nhưng đưa dấu nhắc trỏ về sau đoạn văn bản mới dán
n	gP	giống như P, nhưng đưa dấu nhắc trỏ về sau đoạn văn bản mới dán

: [n] put m	dán m dòng văn bản vào sau dòng thứ n (nếu không có n ngầm định là dòng hiện tại)
: [n] put! m	dán m dòng văn bản vào trước dòng thứ n (nếu không có n ngầm định là dòng hiện tại)

Ngoài các lệnh trên, khi sử dụng **vim** trong **xterm**, người dùng có thể sử dụng chuột để thực hiện các thao tác cho việc sao chép văn bản. Việc này chỉ thực hiện được khi đang ở trong chế độ soạn thảo của **vim**. Nhấn phím trái chuột và kéo từ điểm bắt đầu đến điểm kết thúc của đoạn văn bản cần sao chép. Đoạn văn bản đó sẽ được tự động sao vào bộ nhớ đệm. Sau đó di trỏ soạn thảo đến vị trí cần dán và nháy nút chuột giữa, văn bản sẽ được dán vào vị trí muốn.

Để di chuyển văn bản trong **vim**, cũng phải thực hiện qua ba bước sau:

- Cắt đoạn văn bản và dán vào bộ đệm
- Di chuyển dấu nhắc trỏ tới vị trí mới của đoạn văn bản
- Dán đoạn văn bản vào vị trí mới

Di chuyển văn bản chỉ khác sao chép ở bước đầu tiên là bước cắt đoạn văn bản. hãy sử dụng các lệnh xoá trong **vim** để cắt đoạn văn bản. Ví dụ, khi dùng lệnh **dd**, dòng bị xoá sẽ được lưu vào trong bộ đệm, khi đó có thể sử dụng các lệnh dán để dán văn bản vào vị trí mới.

Ngoài ra còn có thể sử dụng một số lệnh sau:

: [n] m [x]	di chuyển dòng thứ n vào dưới dòng thứ x
' '	dịch chuyển đến vị trí lúc trước
' "	dịch chuyển đến vị trí lúc trước thực hiện việc hiệu chỉnh file

B.3.6. Tìm kiếm và thay thế văn bản trong vim

vim có một số các lệnh tìm kiếm như sau:

/ <xâu>	tìm xâu từ dòng hiện tại đến dòng cuối trong file
? <xâu>	tìm xâu từ dòng hiện tại ngược lên dòng đầu trong file
N	tìm tiếp xâu được đưa ra trong lệnh / hoặc ? (từ trên xuống dưới)
N	tìm tiếp xâu được đưa ra trong lệnh / hoặc ? (từ dưới lên trên)

Xâu được tìm kiếm trong lệnh **/** hay **?** có thể là một biểu thức. Một biểu thức thông thường là một tập các ký tự. Tập ký tự này được xây dựng bằng cách kết hợp giữa các ký tự thông thường và các ký tự đặc biệt. Các ký tự đặc biệt trong biểu thức thường là:

.	thay thế cho một ký tự đơn ngoại trừ ký tự xuống dòng
\	để hiển thị các ký tự đặc biệt
*	thay thế cho 0 hoặc nhiều ký tự
\+	thay thế cho 1 hoặc nhiều ký tự
\=	thay thế cho 0 hoặc một ký tự
^	thay thế cho ký tự đầu dòng
$	thay thế cho ký tự cuối dòng
\ <	thay thế cho chữ bắt đầu của từ

\ >	thay thế cho chữ cuối của từ
[]	thay thế cho một ký tự nằm trong cặp dấu []
[^]	thay thế cho ký tự không thuộc trong cặp dấu [] và đứng sau dấu ^
[-]	thay thế cho một tập có thứ tự các ký tự
\p	thay thế cho một ký tự có thể in được
\s	thay thế cho một ký tự trống
\e	thay thế cho phím Esc
\t	thay thế cho phím Tab

vim sử dụng chế độ lệnh **Ex** để thực hiện các việc tìm kiếm và thay thế. Tất cả các lệnh trong chế độ này được bắt đầu bằng dấu '**:**'. có thể kết hợp lệnh tìm kiếm và thay thay thế để đưa ra được các lệnh phức tạp theo dạng tổng quát sau:

:<điểm bắt đầu>,<điểm kết thúc> s/<mẫu cần thay thế>/<mẫu được thay thế>/[g][c]

Ví dụ, lệnh sau đây:

:1,$s/the/The/g

tìm trong file đang soạn thảo các từ **the** và thay chúng bởi các từ **The**.

B.3.7. Đánh dấu trong vim

m |a-zA-Z|	đánh dấu văn bản tại vị trí hiện thời với dấu là các chữ cái |a-zA-Z|
'|a-z|	dịch chuyển con trỏ tới vị trí đã được đánh dấu bởi các chữ cái |a-z| trong phạm Vim file hiện thời
'|A-Z|	dịch chuyển con trỏ tới vị trí đã được đánh dấu bởi các chữ cái |A-Z| trong một file bất kỳ
:marks	hiển thị các đánh dấu hiện thời

B.3.8. Các phím sử dụng trong chế độ chèn

	Chuyển đổi chế độ
<Insert>	chuyển vào chế độ chèn hoặc chế độ thay thế
<Esc>	thoát khỏi chế độ chèn, trở lại chế độ thông thường
CTRL-C	giống như <Esc>, nhưng ???
CTRL-O <lệnh>	thực hiện <lệnh> và trở về chế độ chèn
	Di chuyển
Các phím mũi tên	di chuyển trỏ soạn thảo sang trái/phải/lên/xuống một ký tự
SHIFT-left/right	di chuyển trỏ soạn thảo sang trái/phải một từ
<Home>	di chuyển trỏ soạn thảo về đầu dòng
<End>	di chuyển trỏ soạn thảo về cuối dòng
	Các phím đặc biệt
<Enter>, CTRL-M, CTRL-J	bắt đầu một dòng mới
CTRL-E	chèn ký tự vào bên phải dấu nhắc trỏ
CTRL-Y	chèn một ký tự vào bên trái dấu nhắc trỏ
CTRL-A	chèn vào trước đoạn văn bản được chèn
CTRL-ừ	chèn vào trước đoạn văn bản được chèn và dừng chế độ chèn

CTRL-R <thanh ghi>	chèn nội dung của một thanh ghi
CTRL-N	chèn từ tiếp theo vào trước dấu nhắc trỏ
CTRL-P	chèn từ trước đó vào trước dấu nhắc trỏ
CTRL-X ...	hoàn thành từ trước dấu nhắc trỏ theo nhiều cách khác nhau
<Backspace>, CTRL-H	xoá một ký tự trước dấu nhắc trỏ
<Del>	xoá một ký tự sau dấu nhắc trỏ
CTRL-W	xoá từ trước dấu nhắc trỏ
CTRL-U	xoá tất cả các ký tự trên dòng hiện tại
CTRL-T	chèn một khoảng trống trước dòng hiện thời
CTRL-D	xoá một khoảng trống trước dòng hiện thời

B.3.9. Một số lệnh trong chế độ ảo

	v	khi nhấn phím này, có thể sử dụng các phím di chuyển để đánh dấu đoạn văn bản hoặc bỏ đánh dấu (văn bản được đánh dấu có màu trắng)
	V	khi nhấn phím này, một dòng văn bản sẽ được đánh dấu và có thể sử dụng các phím di chuyển để đánh dấu đoạn văn bản hoặc bỏ đánh dấu
	CTRL-V	nhấn phím này sẽ đánh dấu một khối văn bản và có thể sử dụng các phím di chuyển để đánh dấu đoạn văn bản hoặc bỏ đánh dấu
	o	di chuyển vị trí dấu nhắc trỏ trên khối được đánh dấu hoặc bỏ đánh dấu
	gv	đánh dấu lại đoạn văn bản được đánh dấu lúc trước
n	aw	chọn đánh dấu n từ
n	as	chọn đánh dấu n câu
n	ap	chọn đánh dấu n đoạn
n	ab	chọn đánh dấu n khối

B.3.10. Các lệnh lặp

n	.	lặp lại n lần thay đổi cuối
	q |a-z|	ghi các ký tự được nhập vào trong thanh ghi |a-z|
n	@|a-z|	thực hiện nội dung có trong thanh ghi |a-z| n lần
n	@@	lặp lại n lần sự thực hiện của lệnh @|a-z| trước
	:@|a-z|	thực hiện nội dung của thanh ghi |a-z| như một lệnh Ex
	:@@	lặp lại sự thực hiện của lệnh :@|a-z| trước
	:[n,m]g/mẫu/[lệnh]	thực hiện lệnh (mặc định là :p) trên các dòng có chứa mẫu nằm trong khoảng từ dòng thứ n đến dòng thứ m
	:[n,m]g!/<mẫu>/[lệnh]	thực hiện lệnh (mặc định là :p) trên các dòng không chứa mẫu nằm trong khoảng từ dòng thứ n đến dòng thứ m
	:sl [n]	tạm dừng trong n giây
n	gs	tiếp tục dừng trong n giây

B.4. Các lệnh khác

B.4.1. Cách thực hiện các lệnh bên trong Vim

:sh	khởi tạo một shell

:! <lệnh>	thực hiện một lệnh shell trong Vim
:!!	lặp lại lệnh ':! <lệnh>' lúc trước
K	mở trang man của lệnh trùng với nội dung từ tại dấu nhắc trỏ
q	thoát khỏi lệnh đang thực hiện trở lại Vim

B.4.2. Các lệnh liên quan đến file

Ngoài các lệnh cơ bản như sao chép hay cắt dán, trong **vim** còn có một số lệnh cho phép có thể có được những thông tin cần thiết về file.

	CTRL-G	hiển thị tên file hiện thời kèm theo trạng thái file và vị trí dấu nhắc trỏ (trạng thái có thể là: chỉ đọc, được sửa, lỗi khi đọc, file mới) (giống **:f**)
n	CTRL-G	hiển thị thông tin như CTRL -G và có thêm đường dẫn đầy đủ của file (nếu n>1, tên buffer hiện thời sẽ được đưa ra)
g	CTRL-G	đưa ra vị trí dấu nhắc trỏ theo dạng: cột/tổng số cột, dòng/tổng số dòng và ký tự/tổng số ký tự
	:f <tên mới>	đổi tên file hiện thời thành tên mới
	:ls	liệt kê tất cả các file hiện thời đang được sử dụng trong Vim (giống **:buffer** và **:files**)
	:cd	đưa thêm đường dẫn vào tên file
	:w <tên file>	tạo một bản sao của file hiện thời với tên mới là tên file (giống như save as trong Win)
		Xác định file cần soạn thảo
	:e[n, /mẫu] <file>	soạn thảo file, từ dòng thứ n hoặc từ dòng có chứa mẫu, trừ khi có sự thay đổi thực sự trong file
	:e[n, /mẫu]! <file>	luôn soạn thảo file, từ dòng thứ n hoặc từ dòng có chứa mẫu, bỏ qua mọi sự thay đổi trong file
	:e	nạp lại file hiện thời, trừ khi có sự thay đổi thực sự trong file
	:e!	luôn nạp lại file hiện thời, bỏ qua mọi sự thay đổi thực sự trong file
	:fin [!] <file>	tìm file trên đường dẫn và soạn thảo
	:e #n	soạn thảo file thứ n (giống n CTRL-^)
		Các lệnh khác
	:pw	đưa ra tên thư mục hiện thời
	:conf <lệnh trong **vim** >	thực hiện lệnh trong **vim** và đưa ra hộp thoại yêu cầu xác nhận khi có thao tác đòi hỏi sự xác nhận

PHỤ LỤC C. MIDNIGHT COMMANDER

C.1. Giới thiệu về Midnight Commander (MC)

Người sử dụng hệ điều hành MS-DOS đều biết tính năng tiện ích Norton Commander (NC) rất mạnh trong quản lý, điều khiển các thao tác về file, thư mục, đĩa cũng như là môi trường trực quan trong chế độ văn bản (text). Dù trong hệ điều hành Windows sau này đã có sự hỗ trợ của tiện ích Explorer nhưng không vì thế mà vai trò của NC giảm đi: Nhiều người dùng vẫn thích dùng NC trong các thao tác với file và thư mục. Linux cũng có một tiện ích mang tên Midnight Commander (viết tắt là MC) có chức năng và giao diện gần giống với NC của MS-DOS và sử dụng MC trong Linux tương tự như sử dụng NC trong MS-DOS.

C.2. Khởi động MC

Lệnh khởi động MC:

 # mc [Tùy-chọn]

Có một số tuỳ chọn khi dùng tiện ích này theo một số dạng thông dụng sau:

-a	Không sử dụng các ký tự đồ hoạ để vẽ các đường thẳng khung.
-b	Khởi động trong chế độ màn hình đen trắng.
-c	Khởi động trong chế độ màn hình màu.
-d	Không hỗ trợ chuột
-P	Với tham số này, Midnight Commander sẽ tự động chuyển thư mục hiện hành tới thư mục đang làm việc. Như vậy, sau khi kết thúc, thư mục hiện hành sẽ là thư mục cuối cùng thao tác.
-v file	Sử dụng chức năng View của MC để xem nội dung của file được chỉ ra.
-V	Cho biết phiên bản chương trình đang sử dụng.

Nếu chỉ ra đường dẫn (path), đường dẫn đầu tiên là thư mục được hiển thị trong panel chọn (selected panel), đường dẫn thứ hai được hiển thị panel còn lại.

C.3. Giao diện của MC

Giao diện của MC được chia ra làm bốn phần. Phần lớn màn hình là không gian hiển thị của hai panel. Panel là một khung cửa sổ hiển thị các file thư mục cùng các thuộc tính của nó hoặc một số nội dung khác. Theo mặc định, dòng thứ hai từ dưới lên sẽ là dòng lệnh còn dòng dưới cùng hiển thị các phím chức năng. Dòng đầu tiên trên đỉnh màn hình là thực đơn ngang (menu bar) của MC. Thanh thực đơn này có thể không xuất hiện nhưng nếu kích hoạt bằng cả hai chuột tại dòng đầu tiên hoặc nhấn phím <F9> thì nó sẽ hiện ra và được kích hoạt.

Midnight Commander cho phép hiển thị cùng một lúc cả hai panel. Một trong hai panel là panel hiện hành (panel chọn). Thanh sáng chọn nằm trên panel hiện hành. Hầu hết các thao tác đều diễn ra trên Panel này. Một số các thao tác khác về file như Rename hay Copy sẽ mặc định sử dụng thư mục ở Panel còn lại làm thư mục đích. Tuy nhiên ta vẫn có thể sửa được thư mục này trước khi thao tác vì các thao tác này đầu tiên bao giờ cũng yêu cầu nhập đường dẫn. Trên panel sẽ hiển thị hầu hết các file và thư mục con của thư mục hiện hành. Midnight Commander có cơ chế hiển thị các kiểu file khác nhau bằng các ký hiệu và màu

sắc khác nhau, ví dụ như các file biểu tượng liên kết sẽ có ký hiệu '@' ở đầu, các file thiết bị sẽ có màu đỏ tím, các file đường ống có màu đen, các thư mục có ký hiệu '/' ở đầu, các thư mục liên kết có ký hiệu '~'...

Cho phép thi hành một lệnh hệ thống từ MC bằng cách gõ chúng lên màn hình. Tất cả những gì có gõ vào đều được hiển thị ở dòng lệnh phía dưới trừ một số ký tự điều khiển và khi nhấn Enter, Midnight Commander sẽ thi hành lệnh gõ vào.

C.4. Dùng chuột trong MC

Midnight Commander sẽ hỗ trợ chuột trong trường hợp không gọi với tham số **-d**. Khi kích chuột vào một file trên Panel, file đó sẽ được chọn, có nghĩa là thanh sáng chọn sẽ nằm tại vị trí file đó và panel chứa file đó sẽ trở thành panel hiện hành. Còn nếu kích chuột phải vào một file, file đó sẽ được đánh dấu hoặc xoá dấu tuỳ thuộc vào trạng thái kích trước đó.

Nếu kích đôi chuột tại một file, file đó sẽ được thi hành nếu đó là file thi hành được (executable program) hoặc nếu có một chương trình đặc trưng cho riêng phần mở rộng đó thì chương trình đặc trưng này sẽ được thực hiện.

Người dùng cũng có thể thực hiện các lệnh của các phím chức năng bằng cách nháy chuột lên phím chức năng đó.

Nếu kích chuột tại dòng đầu tiên trên khung panel, toàn bộ panel sẽ bị kéo lên. Tương tự kích chuột tại dòng cuối cùng trên khung panel, toàn bộ panel sẽ bị kéo xuống.

Có thể bỏ qua các thao tác chuột của MC và sử dụng các thao tác chuột chuẩn bằng cách giữ phím <Shift>

C.5. Các thao tác bàn phím

Một số thao tác của Midnight Commander cho phép sử dụng nhanh bằng cách gõ các phím tắt (hot key). Để tương thích với một số hệ thống khác, trong các bảng dưới đây về Midnight Commander, viết tắt phím CTRL là "C", phím ALT là "M" (Meta), phím SHIFT là "S". Các ký hiệu tổ hợp phím có dạng như sau:

C-<chr>	Có nghĩa là giữ phím CTRL trong khi gõ phím <char>. Ví dụ C -f có nghĩa là giữ CTRL và nhấn <f>.
C-<chr1><char2>	Có nghĩa là giữ phím CTRL trong khi gõ phím <char1> sau đó nhả tất cả ra và gõ phím <char2>.
M-<chr>	Có nghĩa là giữ phím ALT trong khi gõ phím <char>. Nếu không có hiệu lực thì có thể thực hiện bằng cách gõ phím <Esc> nhả ra rồi gõ phím <char>.
S-<chr>	Có nghĩa là giữ phím SHIFT trong khi gõ phím <char>.

Sau đây là chức năng một số phím thông dụng.
Các phím thực hiện lệnh:

Enter	Nếu có dòng lệnh, lệnh đó sẽ được thi hành. Còn nếu không thì sẽ tuỳ vào vị trí của thanh sáng trên panel hiện hành là file hay thư mục mà hoặc việc chuyển đổi thư mục hoặc thi hành file hay thi hành một chương trình tương ứng sẽ diễn ra.
C-l	Cập nhật lại các thông tin trên Panel.

Các phím thao tác trên dòng lệnh:

M-Enter hay C-Enter	chép tên file ở vị trí thanh sáng chọn xuống dòng lệnh
M-Tab	hoàn thành tên file, lệnh, biến, tên người dùng hoặc tên máy giúp
C-x t, C-x C-t	sao các file được đánh dấu (mặc định là file hiện thời) trên panel chọn (C-x t) hoặc trên panel kia (C-x C-t) xuống dòng lệnh
C-x p, C-x C-p	đưa tên đường dẫn hiện thời trên panel chọn (C-x p) hoặc trên panel kia (C-x C-p) xuống dòng lệnh
M-p, M-n	sử dụng để hiện lại trên dòng lệnh các lệnh đã được gọi trước đó. M-p sẽ hiện lại dòng lệnh được thi hành gần nhất, M-n hiện lại lệnh được gọi trước lệnh đó
C-a	đưa dấu nhắc trỏ về đầu dòng
C-e	đưa dấu nhắc trỏ về cuối dòng
C-b, Left	đưa dấu nhắc trỏ di chuyển sang trái một ký tự
C-f, Right	đưa dấu nhắc trỏ di chuyển sang phải một ký tự
M-f	đưa dấu nhắc trỏ đến từ tiếp theo
M-b	đưa dấu nhắc trỏ ngược lại một từ
C-h, Space	xoá ký tự trước đó
C-d, Delete	xoá ký tự tại vị trí dấu nhắc trỏ
C-@	đánh dấu để cắt
C-k	xoá các ký tự từ vị trí dấu nhắc trỏ đến cuối dòng
M-C-h, M-Backspace	xoá ngược lại một từ

Các phím thao tác trên panel:

Up,Down, PgUp, PgDown, Home, End	sử dụng các phím này để di chuyển trong một panel
b, C-b, C-h, Backspace, Delete	di chuyển ngược lại một trang màn hình
Space	di chuyển tiếp một trang màn hình
u, d	di chuyển lên/ xuống 1/2 trang màn hình
g, G	di chuyển đến điểm đầu hoặc cuối của một màn hình
Tab, C-i	hoán đổi panel hiện hành. Thanh sáng chọn sẽ chuyển từ panel cũ sang panel hiện hành
Insert, C-t	chọn đánh dấu một file hoặc thư mục
M-g, M-h, M-j	lần lượt chọn file đầu tiên, file giữa và file cuối trên panel hiển thị

	tìm kiếm file trong thư mục. Khi kích hoạt chế độ này, những ký tự gõ vào sẽ được thêm vào xâu tìm kiếm thay vì hiển thị trên dòng lệnh. Nếu tuỳ chọn Show mini-status trong option được đặt thì xâu tìm kiếm sẽ được hiển thị ở dòng trạng thái.
C-s, M-s	Khi gõ các kí tự, thanh sáng chọn sẽ di chuyển đến file đầu tiên có những ký tự đầu giống những ký tự gõ vào. Sử dụng phím Backspace hoặc Del để hiệu chỉnh sai sót. Nếu nhấn C-s lần nữa, việc tìm kiếm sẽ được tiếp tục
M-t	chuyển đổi kiểu hiển thị thông tin về file hoặc thư mục
C-\	thay đổi thư mục hiện thời
+	sử dụng dấu cộng để lựa chọn đánh dấu một nhóm file. Có thể sử dụng các ký tự đại diện như '*', '?'... để biểu diễn các file sẽ chọn
ữ	sử dụng dấu trừ để xoá đánh dấu một nhóm file. Có thể sử dụng các ký tự dại diện như '*', '?' để biểu diễn các file sẽ xoá
*	sử dụng dấu * để đánh dấu hoặc xoá đánh dấu tất cả các file trong panel
M-o	một panel sẽ hiển thị nội dung thư mục hiện thời hoặc thư mục cha của thư mục hiện thời của panel kia
M-y	di chuyển đến thư mục lúc trước đã được sử dụng
M-u	di chuyển đến thư mục tiếp theo đã được sử dụng

C.6. Thực đơn thanh ngang (menu bar)

Thực đơn thanh ngang trong Midnight Commander được hiển thị ở dòng đầu tiên trên màn hình. Mỗi khi nhấn <F9> hoặc kích chuột tại dòng dầu tiên trên màn hình thực đơn ngang sẽ được kích hoạt. Thực đơn ngang của MC có năm mục "Left", "File", "Command", "Option" và "Right".

Thực đơn Left và Right giúp ta thiết lập cũng như thay đổi kiểu hiển thị của hai panel left và right. Các thực đơn mức con của chúng gồm:

Listing Mode ...		thực đơn này được dùng khi muốn thiết lập kiểu hiển thị của các file. Có bốn kiểu hiển thị: * Full - hiển thị thông tin về tên , kích thước, và thời gian sử đổi của file; * Brief - chỉ hiển thị tên của file; * Long - hiển thị thông tin đầy đủ về file (tương tự lệnh ls -l); * User - hiển thị các thông tin do tự chọn về file;
Quick view	C-x q	xem nhanh nội dung của một file
Info	C-x i	xem các thông tin về một thư mục hoặc file
Tree		hiển thị dưới dạng cây thư mục
Sort order...		thực hiện sắp xếp nội dung hiển thị theo tên, theo tên mở rộng, thời gian sửa chữa, thời gian truy nhập, thời gian thay đổi, kích thước, inode
Filter ...		thực hiện việc lọc file theo tên
Network link ...		thực hiện liên kết đến một máy tính
FTP link ...		thực hiện việc lấy các file trên các máy từ xa

Rescan	C-r	quét lại

Thực đơn File chứa một danh sách các lệnh mà có thể thi hành trên các file đã được đánh dấu hoặc file tại vị trí thanh chọn. Các thực đơn mức con:

User menu	F2	thực đơn dành cho người dùng
View	F3	xem nội dung của file hiện thời
View file ...		mở và xem nội dung của một file bất kì
Filtered view	M-!	thực hiện một lệnh lọc với tham số là tên file và hiển thị nội dung của file đó
Edit	F4	soạn thảo file hiện thời với trình soạn thảo mặc định trên hệ thống
Copy	F5	thực hiện copy
cHmod	C-x c	thay đổi quyền truy nhập đối với một thư mục hay một file
Link	C-x l	tạo một liên kết cứng đến file hiện thời
Symlink	C-x s	tạo một liên kết tượng trưng đến file hiện thời
edit sYimlink	C-x C-s	hiệu chỉnh lại một liên kết tượng trưng
chOwn	C-x o	thay đổi quyền sở hữu đối với thư mục hay file
Advanced chown		thay đổi quyền sở hữu cũng như quyền truy nhập của file hay thư mục
Rename/Move	F6	thực hiện việc đổi tên hay di chuyển đối với một file
Mkdir	F7	tạo một thư mục
Delete	F8	xoá một hoặc nhiều file
Quick cd	M-c	chuyển nhanh đến một thư mục
select Group	M-+	thực hiện việc chọn một nhóm các file
Unselect group	M-ữ	ngược với lệnh trên
reverse selecTion	M-*	chọn các file trong thư mục hiện thời
Exit	F10	thoát khỏi MC

Thực đơn Command cũng chứa một danh sách các lệnh.

Directory tree		hiển thị thư mục dưới dạng cây thư mục
Find file	M-?	tìm một file
Swap panels	C-u	thực hiện tráo đổi nội dung giữa hai panel hiển thị
Switch panels on/of	C-o	đưa ra lệnh shell được thực hiện lần cuối (chỉ sử dụng trên xterm, trên console SCO và Linux)
Compare directories	C-x d	thực hiện so sánh thư mục hiện tại trên panel chọn với các thư mục khác
Command history		đưa ra danh sách các lệnh đã thực hiện
Directory hotlist	C-\	thay đổi thư mục hiện thời
External panelize	C-x !	thực hiện một lệnh trong MC và hiển thị kết quả trên panel chọn (ví dụ: nếu muốn trên panel chọn hiển thị tất cả các file liên kết trong thư mục hiện thời, hãy chọn mục thực đơn này và nhập lệnh find . -type l -print

		sẽ thấy kết quả thật tuyệt vời)
Show directory size		hiển thị kích thước của thư mục
Command history		hiển thị danh sách các lệnh đã thực hiện
Directory hotlist	C-\	chuyển nhanh đến một thư mục
Background	C-x j	thực hiện một số lệnh liên quan đến các quá trình nền
Extension file edit		cho phép hiệu chỉnh file ~/.mc/ext để xác định chương trình sẽ thực hiện khi xem, soạn thảo hay làm bất cứ điều gì trên các file có tên mở rộng

Thực đơn Options cho phép thiết lập, huỷ bỏ một số tuỳ chọn có liên quan đến hoạt động của chương trình MC.

Configuration ...	thiết lập các tuỳ chọn cấu hình cho MC
Lay-out ...	xác lập cách hiển thị của MC trên màn hình
Confirmation ...	thiết lập các hộp thoại xác nhận khi thực hiện một thao tác nào đó
Display bits ...	thiết lập cách hiển thị của các ký tự
Learn keys ...	xác định các phím không được kích hoạt
Virtual FS ...	thiết lập hệ thống file ảo
Save setup	ghi mọi sự thiết lập được thay đổi

C.7. Các phím chức năng

Các phím chức năng của Midnight Commander được hiển thị tại dòng cuối cùng của màn hình. Có thể thực hiện các chức năng đó bằng cách kích chuột lên nhãn của các chức năng tương ứng hoặc nhấn trên bàn phím chức năng đó.

F1	hiển thị trang trợ giúp
F2	đưa ra thực đơn người dùng
F3	xem nội dung một file
F4	soạn thảo nội dung một file
F5	thực hiện sao chép file
F6	thực hiện di chuyển hoặc đổi tên file
F7	tạo thư mục mới
F8	xoá thư mục hoặc file
F9	đưa trỏ soạn thảo lên thanh thực đơn nằm ngang
F10	thoát khỏi MC

C.8. Bộ soạn thảo của Midnight Commander

Midnight Commander cung cấp một bộ soạn thảo khá tiện dụng trong việc soạn thảo các văn bản ASCII. Bộ soạn thảo này có giao diện và thao tác khá giống với tiện ích Edit của DOS hay NcEdit của Norton Commander. Để hiệu chỉnh một số file văn bản, hãy di chuyển thanh sáng chọn đến vị trí file đó rồi nhấn F4, nội dung của file đó sẽ hiện ra trong vùng soạn thảo. Sau khi hiệu chỉnh xong, nhấn F2 để ghi lại. Bộ soạn thảo này có một thực đơn ngang cung cấp các chức năng đầy đủ như một bộ soạn thảo thông thường. Nếu đã từng là người dùng DOS và mới dùng Linux thì nên dùng bộ soạn thảo này để hiệu chỉnh và soạn thảo văn bản thay vì bộ soạn thảo Vim. Sau đây là bảng liệt kê các phím chức năng cũng như các mức thực đơn trong bộ soạn thảo này:

*** Thanh thực đơn**

Thực đơn File:

các thao tác liên quan đến file

Open/load	C-o	mở hoặc nạp một file
New	C-n	tạo một file mới
Save	F2	ghi nội dung file đã được soạn thảo
Save as ...	F12	tạo một file khác tên nhưng có nội dung trùng với nội dung file hiện thời
Insert file ...	F15	chèn nội dung một file vào file hiện thời
Copy to file ...	C-f	sao đoạn văn bản được đánh dấu đến một file khác
About ..		thông tin về bộ soạn thảo
Quit	F10	thoát khỏi bộ soạn thảo

Thực đơn Edit:

các thao tác liên quan đến việc soạn thảo nội dung file

Toggle Mark	F3	thực hiện đánh dấu một đoạn văn bản
Mark Columns	S-F3	đánh dấu theo cột
Toggle Ins/overw	Ins	chuyển đổi giữa hai chế độ chèn/đè
Copy	F5	thực hiện sao chép file
Move	F6	thực hiện di chuyển file
Delete	F8	xoá file
Undo	C-u	trở về trạng thái trước khi thực hiện một sự thay đổi
Beginning	C-PgUp	di chuyển đến đầu màn hình
End	C-PgDn	di chuyển đến cuối màn hình

Thực đơn Sear/Repl:

các thao tác liên quan đến việc tìm kiếm và thay thế

Search ..	F7	thực hiện tìm kiếm một xâu văn bản
Search again	F17	tìm kiếm tiếp
Replace ...	F4	tìm và thay thế xâu văn bản

Thực đơn Command:

Các lệnh có thể được thực hiện trong khi soạn thảo

Goto line ...	M-l	di chuyển trỏ soạn thảo đến một dòng
Insert Literal ...	C-q	chèn vào trước dấu nhắc trỏ một ký tự
Refresh screen	C-l	làm tươi lại màn hình
Insert Date/time		chèn ngày giờ hiện tại vào vị trí dấu nắhc trỏ
Format paragraph	M-p	định dạng lại đoạn văn bản
Sort	M-t	thực hiện sắp xếp

Thực đơn Options:

Các tuỳ chọn có thể thiết lập cho bộ soạn thảo

General ...	thiết lập các tuỳ chọn cho bộ soạn thảo
Save mode ...	ghi lại mọi sự thiết lập được thay đổi

*** Các phím chức năng**

F1	hiển thị trang trợ giúp
F2	ghi nội dung file
F3	thực hiện việc đánh dấu đoạn văn bản
F4	tìm và thay thế xâu văn bản
F5	thực hiện việc sao chép
F6	di chuyển file
F7	tìm kiếm xâu văn bản
F8	xoá đoạn văn bản được đánh dấu
F9	hiển thị thanh thực đơn ngang
F10	thoát khỏi bộ soạn thảo

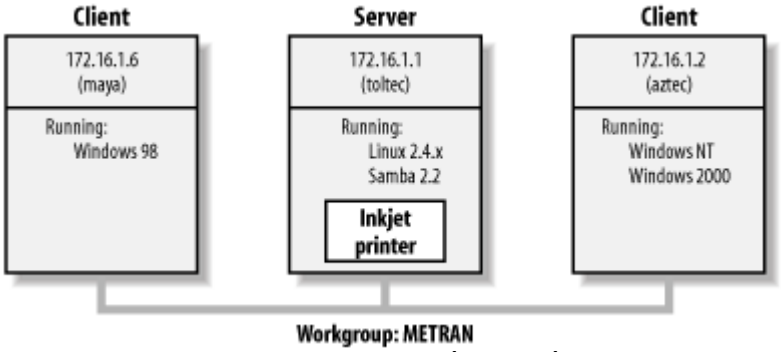
PHỤ LỤC D. SAMBA

D.1 Cài đặt Samba

Nếu các dịch vụ chia sẻ file giữa các máy Windows với nhau và giữa các máy Linux với nhau đã được giải quyết thì còn lại là vấn đề chia sẻ file giữa các máy Linux và Windows lại là một nhu cầu quan trọng. Samba hiện đã trở thành cây cầu nối giữa Linux và Windows. Samba cho phép các máy tính chạy Linux có thể hoạt động và giao tiếp trên cùng một giao thức mạng với máy Windows. Samba server thực hiện các dịch vụ sau:

- Chia sẻ một hay nhiều hệ thống file.
- Chia sẻ các máy in đã được cài đặt ở cả hai phía server và client của nó.
- Hỗ trợ các client duyệt Network Neighborhood trong các máy windows.
- Kiểm tra xác nhận các client đăng nhập vào vùng của Windows.
- Cung cấp hoặc hỗ trợ việc phân giải địa chỉ của server bằng WINS (Windows Internet Name Server).

Để dễ hình dung hơn, ta lấy ví dụ mạng đơn giản có dùng Samba. Giả sử ta có một cấu hình mạng cơ bản sau: một máy chủ Linux có sử dụng Samba với tên được đặt là Toltec, và hai client trên MS Windows, với tên là Maya và Aztec, được nối với nhau trong một mạng Lan và tham gia vào nhóm làm việc có tên METRAN. Ta còn giả sử máy chủ Toltec còn có một máy in phun tên là lp được nối tại chỗ, đồng thời việc chia sẻ đĩa cứng cho phép các máy "nhìn" thấy nhau. Mô tả mạng trên bằng hình vẽ D.1 được biểu diễn dưới đây:



Hình D.1 Một mạng có nhiều hệ điều hành

Các vai trò của Samba đối với mạng Windows NT (phiên bản Samba 2.0.4b.)

Vai Trò	*Có thể thực hiện được*
File Server	Có
Printer Server	Có
Primary Domain Controller	Có (Samba 2.1 hay muộn hơn)
Backup Domain Controller	Không
Window 95/98 Authentication	Có
Local Master Browser	Có
Local Backup Browser	Không
Domain Master Browser	Có
Primary WINS Server	Có
Secondary WINS Server	Không

D.2 Các thành phần của Samba

Samba thực chất chứa một số chương trình phục vụ cho những mục đích khác nhau nh-ng có liên quan với nhau. Hạt nhân của Samba là hai deamon có những nhiệm vụ sau:

smbd Daemon: smbd chịu trách nhiệm điều khiển các tài nguyên được chia sẻ giữa máy chủ Samba và các máy trạm của nó. Nó cung cấp các dịch vụ về file, in, và trình duyệt cho các máy trạm SMB thông qua một hay nhiều mạng. smdb xử lý tất cả các trao đổi giữa máy chủ Samba và các client mạng của nó. Ngoài ra, deamon này còn chịu trách nhiệm kiểm tra xác nhận người dùng, khoá tài nguyên, và chia sẻ dữ liệu thông qua giao thức SMB.

nmbd Deamon: nmbd là một máy chủ dịch vụ tên đơn giản bắt chước các chức năng máy chủ dịch vụ tên, chạy với các giao thức WINS và NetBIOS. Daemon này ỏlắng ngheÕ các yêu cầu của máy chủ dịch vụ tên và cung cấp các thông tin thích hợp khi được gọi tới. Nó còn cung cấp danh sách duyệt Network Neighborhood và tham gia vào lựa chọn các đối tượng mạng trong đó.

Bộ cài đặt Samba còn có một tập hợp nhỏ các công cụ dòng lệnh Linux:

smbclient: Một client Linux theo kiêu ftp có thể dùng tiện ích này để kết nối với tài nguyên được Samba chia sẻ.

Smbtar: Chương trình để lưu trữ các tài nguyên được chia sẻ, tương tự như lệnh tar của Linux.

nmblookup: Chương trình cung cấp NetBIOS thông qua việc tìm tên bằng TCP/IP.

smbpasswd: Chương trình cho phép ngưười quản trị thay đổi mật khẩu đã mã hóa của Samba.

testparm: Chương trình đơn giản để làm cho file cấu hình Samba có hiệu lực.

testprns: Chương trình kiểm tra liệu các máy in khác nhau có được deamon smbd nhận ra hay không.

Nếu muốn xem từng deamon thực hiện những gì, Samba có chương trình với tên smbstatus sẽ đa tất cả các thông tin đó lên màn hình như sau:

```
Samba version 2.2.7-security-rollup-fix
Service uid gid pid machine
----------------------------------------------
IPC$ root root 21608 httt-09 (10.10.16.5) Fri Nov 28 09:42:52 2003
No locked files
```

Việc cài đặt samba cũng khá đơn giản, ta cần chuẩn bị các package sau:

```
samba-client-xxx.rpm
samba-xxx.rpm
samba-common-xxx.rpm
```

Trong đó xxx là số hiệu phiên bản của samba. Đăng nhập với quyền root và sau đó ra lệnh:

```
#rpm –ivh samba-client-xxx.rpm samba-xxx.rpm samba-
common-xxx.rpm
```

Nếu ta không nhận lời thông báo lỗi nào cả thì quá trình cài đặt đã hoàn tất.

D.3 File cấu hình Samba

Những tên được bao trong các ngoặc vuông dùng để ký hiệu cho các phần của file cấu hình smb.conf, mô tả các chia sẻ hay dịch vụ mà Samba cung cấp. Ví dụ, các phần "test" và "homes" là các chia sẻ riêng rẽ đối với đĩa cứng; chúng chứa các tuỳ chọn được ánh xạ tới các thư mục cụ thể trên server Samba. Phần chia sẻ "printers" chứa các tuỳ chọn ánh xạ tới các máy in khác nhau của servers. Tất cả các phần được xác định trong file smb.conf, trừ phần [global, sẽ được coi như các chia sẻ đĩa cứng hoặc máy in cho những dùng kết nối với server Samba.

Các dòng vào còn lại là các tuỳ chọn riêng được quy định cụ thể cho sự chia sẻ đã được đề cập tới. Các tuỳ chọn đó có tác dụng cho tới khi bắt đầu một phần mới được ký hiệu trong cặp ngoặc vuông, hoặc cho tới điểm cuối của file smb.conf được thiết lập bằng cách gán giá trị cho chúng. Mỗi một tuỳ chọn cấu hình đều có cú pháp đơn giản: `option = value`

Cuối cùng, ta có thể dùng khoảng trống dể ngăn cách chuỗi các giá trị trong danh sách, hoặc có thể dùng các dấu phảy, Hai cách trên là tơng đơng nhau nhng ta chỉ nên dùng một cách hoặc dấu phảy hoặc khoảng trống.

Chữ viết hoa không có ý nghĩa gì đối với file cấu hính Samba, ngoại trừ trong các vị trí mà hệ điều hành được chỉ tới không cho phép viết, bởi vì hệ điều hành Linux phân biệt chữ viết thường và viết hoa.

Trong trường hợp dòng vào quá dài không thể gói gọn trong không gian mà cửa sổ dòng lệnh cho phép, ta có thể viết tiếp dòng trong file cấu hình Samba bằng cách dùng ký hiệu dấu gạch ngược "\", ví dụ:

```
comment = Su chia se dau tien la ban sao chinh cua s\
an pham phan mem Teamworks moi
```

Có thể thay đổi file smb.conf và bất kỳ tuỳ chọn nào của nó vào thời điểm bất kỳ trong khi các daemon Samba đang chạy. Theo mặc định, Samba kiểm tra file cấu hình của mình cứ 60 giây một lần để tiếp nhận cá thay đổi mới. Nếu không muốn chờ đợi lâu như vậy, bạn có thể bắt các daemon đó nạp lại bằng cách gửi tín hiệu SIGHUP tới chúng, hoặc chỉ đơn giản là khởi động lại. Ví dụ, nếu tiến trình **smbd** có PID là 893, ta có thể bắt nó đọc lại file cấu hình bằng lệnh sau đây:

```
# kill –SIGHUP 893
```

Không phải tất cả các thay đổi đều được cá máy client chấp nhận ngay. Ví dụ, các tài nhuyên chia sẻ hiện thời đang được sử dụng sẽ không được đăng ký cho đến khi các máy client cắt các nối kết rồi kết nối lại tới các tài nguyên đó. Thêm nữa, cũng sẽ không được đăng ký ngay lập tức. Điều đó giúp cho các máy client tích cực không bị ngắt nối kết một cách bất ngờ hoặc gặp phải các vấn đề không được chờ đợi về truy cập tài nguyên khi phiên làm việc vẫn đang được mở.

***Các biến*:** Samba có một tập hợp đầy đủ cá biến xác định các đặc trưng của server Samba và của cá máy client nối với nó. Mỗi một biến được bắt đầu bằng dấu phần trăm "%", tiếp theo là một ký tự đơn viết hoa hoặc viết thường và chỉ có thể được dùng bên vế phải của dòng liịnh tuỳ chọn cấu hình theo cú pháp potion = variable nh trong câu li lệnh ví dụ sau:

```
[pub]
path = /home/ftp/pub/%a
```

Ký hiệu biến %a có nghĩa đại diện cho kiến trúc của máy tính client, như WinNT để chỉ các máy tính chạy trên Windows NT, Win95 – cho máy Windows 95 hoặc 98, hay như WfWg – cho Windows for Worrgroups (Windows 3.11). Theo cách viết trên, Samba sẽ gán đường dẫn chung chỉ tới tài nguyên được chia sẻ trong phần [pub] cho các máy client chạy trên Windows NT, gán đường dẫn khác cho các máy Windows 9x, và một đường dẫn nữa cho các máy với Windows for Worrkgroups. Nói cách khác, các đường dẫn mà theo đó mỗi máy client nhận thấy tài nhuyên chia sẻ sẽ khác nhau, tuỳ thuộc vào kiến trúc của client.

Biến	Định nghĩa
Các biến của máy client	
%a	Kiến trúc của các máy client(ví dụ, Samba, wfwg, winNT, win95, hoặc UNKNOWN).
%I	Địa chỉ IP của client(ví dụ, 192.168.220.100).
%m	Tên NetBIOS của client.
%M	Tên DNS của client.
Các biến về người dùng	
%g	Nhóm chính của %u
%G	Nhóm chính của %U
%u	Thư mục home hiện thời của %u.
%U	Tên người dùng được yêu cầu trên máy client
Các biến về tài nguyên được chia sẻ	
%p	Đường dẫn cho automounter tới thư mục gốc của tài nguyên được chia sẻ, nếu thư mục đó khác với %P
%P	Th mục gốc hiện thời của tài nguyên được chia sẻ.
%S	Tên hiện thời của tài nguyên được chia sẻ.
Các biến của server	
%d	Định danh tiến trình (PID) server hiện thời.
%h	Tên host DNS của server Samba.
%L	Tên host NetBIOS của server Samba
%N	Th mục home của server Samba, lấy từ file ánh xạ (map) của **automount.**
%v	Phiên bản Samba.
Các biến khác	
%R	Mức của giao thức SMB được thoả thuận thiết lập.
%T	Ngày giờ hiện tại.

Danh sách các biến của Samba

D.4 Các phần đặc biệt của file cấu hình Samba

***Phần [global]*:** Phần [global] xuất hiện hầu như trong mọi file cấu hình Samba, thậm chí khi trong đó không có dòng lệnh bắt buộc nào. Mọi tuỳ chọn được thiết lập trong phần này đều được áp dụng đối với tất cả các tài nguyên chia sẻ khác, vì nội dung của phần này sẽ được sao chép vào các phần khác. Nhưng các phần khác, nếu cũng có các tuỳ chọn giống như trong phần [global], thì trong các phần đó, các tuỳ chọn sẽ được xác định với các giá trị mới được ghi đè lên các lên các giá trị cũ của `[global]`. Ta có thể cấu hình server

Samba, đầu tiên ta phải chú ý đến ba tuỳ chọn cấu hình cơ bản xuất hiện trong phần ửglobalứ của file cấu hình smb.conf:

```
[global]
Server configuration parameters
netbios name = HYDRA
server string = Samba %v on (%L)
workgroup = SIMPLE
```

Tuỳ chọn netbios name: cho phép đặt tên NetBIOS cho server. Ví dụ:
netbios name = DHQGHN

Giá trị mặc định cho tuỳ chọn này là tên máy của server (phần bên trái cùng của tên DNS đầy đủ). Ví dụ, tên NetBIOS mặc định của máy hut.edu.vn sẽ là HUT. Thông thường, người ta đặt tên NetBIOS khác với tên DNS hiện thời.

Việc thay đổi tên NetBIOS của server không được khuyến khích nếu không có lý do chính đáng, nếu như tên đó không phải là duy nhất vì mạng LAN được chia ra thành hai hay nhiều vùng DNS. Ví dụ, khi mạng `hut.edu.vn` bị chia thành hai vùng với các server là `hut.lythuyet.edu.vn` và hut.thuchanh.edu.vn thì tên NetBIOS cũ là `HUT` bây giờ có thể thành các tên `HUTLYTHUYET` và `HUTTHUCHANH`.

Tùy chọn server string: Thông số của server string xác định nội dung dòng chú thích sẽ xuất hiện cạnh tên của server Samba trong cả cửa sổ Network Neighborhood (khi ở chế độ Details) lẫn cửa sổ quản lý in của Microsoft Windows. Bạn có thể dùng các biến chuẩn để cung cấp thông tin cho dòng mô tả đó, ví dụ trên ta đã sử dụng hai biến là %v và %L.

Tùy chọn workgroup: Thông số của tuỳ chọn workgroup thiết lập nhóm làm biệc hiện thời, nơi mà server Samba tự thông báo cho các thành biên của mạng về mình. Các clients muốn truy cập được tài nguyên được chia sẻ trên server Samba phải cùng thuộc về một nhóm làm việc NetBIOS. Nên nhớ rằng các nhóm làm việc phải có các tên nhóm NetBIOS thực thụ, tuân theo quy tắc đặt tên NetBIOS.

Cấu hình chia sẻ đĩa cứng

Trong ví dụ ở phần trước ta đã nhắc đến rằng do chưa có tài nguyên được chia sẻ nên cửa sổ chi tiết của server hydra đang còn trống. Bây giờ ta tiếp tục làm việc cới file cấu hình Samba và tạo ra một đĩa cứng được chia sẻ còn rỗng có tên là `[data]`. Đây là các động tác cần thêm vào để đạt được kết quả vừa nêu:

```
SampleDataDrive]
comment=Data Drive
path = /export/samba/data
writable = yes
guest ok = yes
```

Tài nguyên được chia sẻ ửSampleDataDriveứ thường là đĩa cứng được Samba chia sẻ và ánh xạ tới thư mục `/export/samba/data` trên server Samba. Ta đã cho thêm một dòng vào để có chú thích mô tả tài nguyên được chia sẻ đó là Data Drive, cũng như gán cho bản thân tài nguyên đó một cái tên `SampleDataDrive`.

Tài nguyên được chia sẻ được thiết lập có quyền ghi cho các người dùng. Giá trị mặc định của tuỳ chọn này là chỉ đọc. Trong phương án không cần nức độ bảo mật chặt chẽ như ở đây, ta đặt giá trị yes cho tuỳ chọn `guest ok`, để cho bất kỳ ai cũng có thể kết nối được

tới tài nguyên vừa được chia sẻ. Trên máy UNIX có cài đặt Samba ta tạo thư mục `/export/samba/data` với quyền root bằng các lệnh sau:

```
# mkdir /export/samba/data
#chmod 777 /export/samba/data
```

Bây giờ, nếu ta lại kết nối với server hydra (bằng cách kích phím chuột vào biểu tượng của server trong cửa sổ Network Neighborhood của Windows), một thư mục được chia sẻ với tên data đã xuất hiện.

Tuỳ chọn	Thông số	Chức năng	Mặc định	Phạm vi
Path (directory)	String (đường dẫn đến thư mục)	Đặt thư mục UNIX dùng cho chia sẻ đĩa cứng hoặc cho việc xếp hàng chờ bởi máy in được chia sẻ.	/tmp	Share
Guest ok (public)	Nhị phân (yes/no)	Nếu đặt là ***yes***, sẽ không dần kiểm tra xác nhận người dùng để truy cập tài nguyên được chia sẻ này.	***no***	Share
Comment	String (xâu ký tự)	Đặt chú thích sẽ xuất hiện cùng tài nguyên được chia sẻ.	Không có	Share
volume	String	Đặt tên cho ổ đĩa, theo dạng của DOS.	Tên của tài nguyên được chia sẻ	Share
Read only	Nhị phân (yes/no)	Nếu là ***yes***, cho phép truy cập chỉ đọc tới tài nguyên được chia sẻ.	***yes***	Share
Writeable (write ok)	nhị phân (yes/no)	Nếu là ***no***, cho phép truy cập chỉ đọc tới tài nguyên được chia sẻ.	***no***	Share

Các tuỳ chọn cơ bản chia sẻ đĩa cứng

Các tuỳ chọn về mạng của Samba

Tuỳ chọn	Thông số	Chức năng	Mặc định	Phạm vi
hosts allow (allow hosts)	String (danh sách tên máy)	Xác định các máy có thể kết nối với Samba.	Không có	Share
Hosts deny (deny hosts)	String (danh sách tên máy)	Xác định các máy không thể kết nối với Samba.	Không có	Share
Bind interfaces only	Nhị phân (yes/no)	Nếu đặt là ***yes***, Samba sẽ chỉ liên kết tới các giao diện xác định bởi tuỳ chọn	***no***	Global

		interfaces.		
socket address	String (địa chỉ IP)	Đặt địa chỉ IP để ỏngheÕ, dùng cho trờng hợp có nhiều giao thức ảo trên servers.	Không có	Global

Các tuỳ chọn cấu hình mạng

Tùy chọn hosts allow: Tuỳ chọn này xác định các máy có quyền truy cập các tài nguyên được chia sẻ trên servers Samba, được viết như danh sách các máy hay địa chỉ IP của chúng, cách nhau bằng dấu phảy hoặc khoảng trống. Ta có thể đặt được một chút ít mức độ bảo mật, chỉ đơn giản bằng cách đặt địa chỉ mạng con LAN của mình vào chỗ giá trị của tuỳ chọn này. Ví dụ:

```
hosts allow = 192.168.200. localhost
```

Chú ý rằng ta đặt localhost (hoặc địa chỉ 127.0.0.1) ở vị trí sau địa chỉ của mạng con. Một trong số các lỗi thường thấy khi dùng tuỳ chọn hosts allow là cấm luôn servers Samba liên hệ với chính nó. Chương trình smbpasswd sẽ cần được kết nối với servers Samba như là một client để thay đổi mật khẩu mã hóa của người dùng. Thêm nữa, việc duyệt tại chỗ cũng đòi hỏi có được đăng nhập tại chỗ.

Sau đây là các quy tắc của Samba quy định cho việc dùng các tuỳ chọn `hosts allow` và `hosts deny`:

- Nếu không có tuỳ chọn allow hoặc deny nào được xác định trong file cấu hình smb.conf Samba sẽ cho phép các kết nối từ bất kỳ máy nào mà hệ thống Unix chấp nhận
- Nếu có các tuỳ chọn allow hoăc deny được xác trong phần ửglobalứ của file cấu hình smb.conf, chúng sẽ được áp dụng cho tất cả các tài nguyên được chia sẻ, thậm chí khi một tài nguyên nào đó có tuỳ chọn ghi đè lên được xác định.
- Nếu chỉ có tuỳ chọn allow được xác định cho một tài nguyên được chia sẻ, chỉ có các máy được liệt kê mới có quyền truy cập tài nguyên đó. Các máy khác đều bị cấm.
- Nếu chỉ có tuỳ chọn deny được xác định cho một tài nguyên được chia sẻ, mọi máy không có trong danh sách đều có quyền sử dụng tài nguyên đó.
- Nếu cả hai tuỳ chọn allow và deny được xác định, một máy đã xuất hiện trong danh sách được phép thì không có mặt trong danh sách bị cấm. Nếu không máy đó sẽ bị cấm truy cập vào tài nguyên được chia sẻ.

Chý ý: Cẩn thận tránh trờng hợp ta cho phép một máy nào đó, nhưng sau đấy lại cấm cả mạng con mà máy ấy tham gia.

hosts deny: Tuỳ chọn hosts deny xác định các máy không có quyền truy nhập tài nguyên được chia sẻ, được viết như danh sách các tên máy hoặc địa chỉ IP của chúng, cách nhau bằng dấu phẩy hay khoảng trống với cú pháp giống như đới với tuỳ chọn `hosts allow` ở trên. Ví dụ, để hạn chế truy cập tới servers từ một máy, trừ từ vùng example.com, ta có thể viết:

```
hosts deny = ALL EXCEPT.example.com
```

Giống như hosts allow, không có giá trị mặc định cho tuỳ chọn hosts deny. Nếu muốn cho phép hay cấm truy cập tới tài nguyên được chia sẻ cụ thể ta phải qua cả hai tuỳ chọn `hosts allow` và `hosts deny` trong phần ửglobalứ hay nếu có dụng thì phải ghi đè giá trị mới trong phần cấu hình cho tài nguyên được chia sẻ đó.

interfaces: Tuỳ chọn interfaces liệt kê các địa chỉ mạng mà ta muốn servers Samba nhận biết và đáp ứng. Tuỳ chọn này rất tiện lợi nếu ta muốn máy tính tham gia đồng thời nhiều mạng con. Nếu không được dùng Samba tìm giao diện mạng chính của servers (thường là card Ethernet đầu tiên) khi khởi động và tự cấu hình để hoạt động chỉ trong mạng con có giao diện mạng đó. Ta phải dùng tuỳ chọn này để bắt buộc Samba phải thực hiện mạng con khác nữa trong mạng của ta.

Giá trị của tuỳ chọn là một hay nhiều bộ gồm các đôi địa chỉ IP/ mặt nạ mạng, giống như trong ví dụ sau:

```
interfaces = 192.168.220.100/255.255.255.0 192.168.210.30/255.255.0
```

Có thể dùng định dạng mặt nạ bít CIDR nh sau:

```
interfaces = 192.168.220.100/24 192.168.210.30/24
```

Số của mặt nạ bít chỉ số đầu tiên được bật trong mặt nạ mạng, ví dụ số 24 nghĩa là 24 bít đầu tiên (trong số tất cả 32 bit) sẽ được kích hoạt, hay đông nghĩa với giá trị mặt nạ mạng 255.255.255.0. Tương tự nh vậy, số 16 tương đương với mặt nạ 255.255.0.0, và 8- với 255.0.0.0. Tuy nhiên, tuỳ chọn này có thể hoạt động không đúng nếu ta dùng DHCP (phân phối địa chỉ IP động).

Bind interfaces only: Tuỳ chọn này có thể được dùng để bắt buộc các tiến trình smbd và nmbd phục vụ các yêu cầu SMB chỉ cho các địa chỉ được xác định bởi tuỳ chọn interfaces mà thôi. Tiến trình nmbd bình thường liên kết giao diện (0.0.0.0.) trên các cổng 137 và 138 tới tất cả các địa chỉ, cho phép chúng nhận các thông báo phân phối công cộng từ khắp mọi nơi. Tuy nhiên, nếu ta ghi đè lên giá trị đó bằng:

```
bind interfaces only = yes
```

Thì chỉ các gói đi từ các địa chỉ nguồn xác thông qua tuỳ chọn interfaces mới được chấp nhận. Với smbd, tuỳ chọn này cũng bắt Samba không phục vụ các yêu cầu về file của các mạng con ngoài danh sách của tuỳ chọn interfaces. Nếu muốn cho phép có các nối kết mạng tạm thời, như dụng SLIP hoặc ppp, ta không được dụng tuỳ chọn này. Nói chung, tuỳ chọn này ít được dùng, và thường chỉ có nhng người quản trị đầy kinh nghiệm mới để ý tới nó.

Nếu đặt giá trị cho bind intefaces only là yes, ta phải thêm địa chỉ của máy tại chỗ (127.0.0.1) vào danh sách của interfaces, nếu không smbpasswd sẽ không thể hoạt động được.

socket address: Tuỳ chọn socket address quy định địa chỉ nào trong số được xác định bởi interfaces sẽ "ghe" tức là chờ các kết nối. Samba theo mặc định chấp nhận tất cả các nối kết với tất cả các địa chỉ. Khi được dùng trong file smb.conf, tuỳ chọn này hạn chế số địa chỉ mà Samba sẽ dùng để chờ các nối kết. Vi dụ:

```
Interfaces = 192.168.220.100/24 192.168.210.30/24
Socket address = 192.168.210.30
```

Bình thường, tuỳ chọn này không được khuyên dùng.

Nếu như có dùng các mật khẩu đã mã hoá, ta phải thêm vào một dòng có nội dung `encrypt passwords=yes` vào file cấu hình trên.

Sau khi đã soạn thảo nội dung như trên của file smb.conf và đặt nó vào đúng vị trí cần thiết, ta khởi động lại server Samba và dùng các máy client Windows để kiểm tra kết quả.

Tất nhiên các máy client Windows đó cũng phải thuộc về nhóm SIMPLE – trong ví dụ ta vẫn dùng từ đầu chơng – đó là các máy phoenix và chimaera.

Mọi tuỳ chọn xuất hiện trớc phần được đánh dấu bằng ngoặc vuông "[]"ầu tiên, tức là bên ngoài phần [lobal] cũng được coi là những tuỳ chọn chung.

Phần [mes] Nếu một client nào đó cố gắng kết nối tới tài nguyên được chia sẻ không được nêu trong file cấu hình smb.conf, Samba sẽ tìm tài nguyên được chia sẻ ửhomesứ trong file cấu hình. Nếu phần này tồn tại, tên của tài nguyên được chia sẻ không xác định kia sẽ được coi như tên người dùng của Linux và được yêu cầu tìm trong cơ sở dữ liệu mật khẩu của server Samba. Nếu như có tên ngưười dùng đó, Samba coi máy được nói tới ở trên là một ngưười dùng Linux đang cố kết nối tới th mục home của mình trong server.

Ví dụ, giả sử một máy client kết nối với server Samba hydra lần đầu tiên, và cố truy cập tới tài nguyên được chia sẻ có tên là [dung]. Trong file smb.conf, không có tài nguyên được chia sẻ nào tên là ửdungứ được xác định, nhng lại có phần [homes], vì thế Samba tìm file cơ sở dữ liệu mật khẩu và tìm xem có tài khoản ngưười dùng dung trong hệ thống hay không, Sau đó Samba kiểm tra mầy khẩu được client cung cấp và so sánh với mật khẩu của ngưười dùng Linux dung - hoặc trong file cơ sở dữ liệu mật khẩu nếu dùng mật khẩu mã hoá. Nếu các mật khẩu đó trùng nhau, Samba nhận biết chắc là ngưười dùng dung có quyền và đang muốn kết nối tới th mục home của mình trong máy Linux. Sau đó Samba sẽ tự tạo tài nguyên được chia sẻ được gọi là ửdungứ cho người dùng dung.

Người ta cũng áp dụng phơng pháp được thực hiện với phần [homes] để tạo tài khoản người dùng mới, kèm theo mật khẩu.

Phần [printers]: Phần đặc biệt thứ ba gọi là [printers] tương tự như phần [homes]. Nếu một client cố kết nối tới tài nguyên được chia sẻ không có mặt trong file cấu hình smb.conf file, và nếu tên của nó không thể tìm được trong file mật khẩu, Samba sẽ kiểm tra xem nó có phải sự chia sẻ máy in cho client đó. Samba thực hiện điều đó thông qua việc đọc file dữ liệu máy in (thường là /etc/printcap hay /etc/terminfo) để xem có tên của tài nguyên được chia sẻ đó hay không. Nếu có, Samba tạo ra tài nguyên được chia sẻ với tên liên quan tới việc chia sẻ máy in. Để có thể in được trong Samba ta phải thêm các tuỳ chọn printer driver, printer driver file, và printer driver location vào file cấu hình smb.conf của Samba. Tuỳ chọn chung printer driver file chỉ đến file printers.def phải được đặt vào phần [global]. Các tuỳ chọn còn lại được đặt vào phần tài nguyên máy in được chia sẻ mà ta muốn cấu hình một cách tự động các trình điều khiển máy in. Giá tri cho printer driver phải trùng với xâu được hiện ra trong *Printer Winzard* trên hệ thống *Windows*. Giá trị của printer driver location là đ-ờng dẫn của tài nguyên PRINTER$ mà ta thiết lập, chứ không phải là đờng dẫn UNIX trên server. Do đó, ta có thể dùng các dòng mã sau đây trong file cấu hình Samba:

```
[global]
printer driver file = /usr/local/samba/print/printers.def
[hpdeskjet]
path = /var/spool/samba/printers
printable = yes
printer driver = HP DeskJet 560C Printer
printer driver location = \\%L\PRINTER$
```

Giống như đối với phần [home], ta không cần phải bảo trì tài nguyên được chia sẻ cho mỗi một may in của hệ thống trong file cấu hình smb.conf. Thực vậy, Samba luôn dựa vào

việc đăng ký máy in của Linux nếu ta cần đến, và cung cấp các máy in đã đăng ký cho các client. Tuy nhiên, có một hạn chế nhỏ: nếu tài khoản ngưười dùng và máy in đều có tên là hai, Samba bao giờ cũng tìm tài khoản ngưười dùng trớc tiên, bất kể là client thực ra là cần kết nối với máy in.

Các chi tiết vè việc thiết lập tài nguyên được chia sẻ ửprintersứ được trình bày trong phần liên quan tới việc in và phân giải tên.

Các tuỳ chọn cấu hình: Các tuỳ chọn trong file cấu hình Samba được chia sẻ làm hai loại: global (toàn cục) và share (chia sẻ). Mỗi một loại quy định một tuỳ chọn sẽ được xuất hiện ở đâu trong file cấu hình.

Global (toàn cục): Các tuỳ chọn `global` phải có mặt chỉ trong phần `[global]` mà thôi. Đây là các tuỳ chọn thường chỉ ps dụng để xác định hoạt động của chính server Samba.

Share: Các tuỳ chọn `share` có thể xuất hiện trong các tài nguyên được chia sẻ cụ thể, hoặc cả trong phần `[global]`. Nếu có mặt trong phần `[global]`, chúng sẽ xác định các giá trị mặc định cho tất cả các tài nguyên được chia sẻ, chừng nào cha bị các tuỳ chọn cùng tên tại các phần tài nguyên được chia sẻ cụ thể ghi đè những giá trị mới.

D.5 Quản lý người dùng trong Samba

Samba có khả năng quản lý người dùng có khả năng truy cập vào máy chủ Samba. Nó có khả năng quản lý người dùng khá độc lập với hệ thống người dùng hệ thống. Thông thường các thông tin về người dùng sẽ được lưu trong file smbpasswd, file này nằm trong thư mục /etc/samba. Để thêm một người dùng cho samba quản lý, người dùng đó phải là một người dùng trong hệ thống. Sau đó, để thao tác với những người dùng của samba, ta có công cụ smbpasswd.

`smbpasswd [-a][-x][-d][-e][-h][-s] [` `tên người dùng` **`]`**

Trong đó,

- a : tuỳ chọn này cho phép ta thêm một người dùng mới vào trong danh sách người dùng của samba.
- x : tuỳ chọn này cho phép xoá bỏ một người dùng trong danh sách người dùng của samba.
- d : tuỳ chọn này cho phép ta khoá (disable) một người dùng trong danh sách người dùng của samba.
- e : tuỳ chọn này cho phép ta mở khoá (enable) một người dùng trong danh sách người dùng của samba mà người dùng đó đã bị khoá bằng tham số -d.
- <tên người dùng>: tên của người dùng ta muốn xử lý.

Chẳng hạn, muốn thêm một người dùng vào trong danh sách người dùng của samba, ta dùng lệnh (sử dụng lệnh này với quyền root):

`#smbpasswd –a thanhnt`

Trong đó người dùng thanhnt phải là một người dùng hệ thống. Sau khi đánh lệnh này, máy sẽ hỏi ta đánh vào mật khẩu cho người dùng mới này, và samba cho phép người dùng do nó quản lý có thể có mật khẩu khác với mật khẩu hệ thống của người dùng đó.

New SMB password:
Retype new SMB password:
Password changed for user thanhnt.

Lưu ý là mật khẩu sẽ được hỏi hai lần để đảm bảo tính chính xác và mật khẩu sẽ không được hiển thị ra màn hình. Nếu thành công thì ta sẽ nhận được thông báo như trên. Ta cũng có thể dùng lệnh này để thay đổi mật khẩu của một người dùng bằng lệnh (thực hiện bằng quyền root):

```
#smbpasswd thanhnt
```

Khi đó nó sẽ thông báo cho ta nhập mật khẩu hai lần giống như trên. Còn trong trường hợp là một người dùng bình thường thì muốn thay đổi mật khẩu samba cho chính người dùng đó ta chỉ cần đánh:

```
#smbpasswd
Old SMB password:
New SMB password:
Retype new SMB password:
Mismatch - password unchanged.
Unable to get new password.
```

Trong trường hợp trên, máy sẽ yêu cầu ta nhập mật khẩu cũ trước khi nhập mật khẩu mới, nếu có sai sót (mật khẩu cũ không đúng hoặc mật khẩu mới không khớp nhau) thì ta sẽ nhận được thông báo lỗi.

Nếu muốn xoá người dùng ra khỏi danh sách người dùng thì sử dụng lệnh (với quyền root):

```
#smbpasswd –x thanhnt
```

Còn nếu muốn một người dùng trong danh sách vẫn tồn tại nhưng không có hiệu lực, thì ta có thể khoá người dùng đó bằng lệnh:

```
#smbpasswd –d thanhnt
```

Khi đó người dùng thanhnt tuy vẫn còn nằm trong danh sách nhưng không được samba coi là người dùng hợp lệ nữa. Khi muốn khôi phục người dùng này có các quyền như ban đầu thì ta có thể khôi phục bằng lệnh:

```
#smbpasswd –e thanhnt
```

D.6 Cách sử dụng Samba từ các máy trạm

D.6.1 Cách sử dụng từ các máy trạm là Linux

Samba có cung cấp một công cụ nhằm sử dụng các thư mục chia sẻ theo giao thức SMB trong mạng LAN, đó chính là smbclient. Với công cụ này ta có thể thao tác với tài nguyên được chia sẻ trên mạng, chẳng hạn như kết nối vào một thư mục chia sẻ trên một máy nào đó để thao tác, sao chép file từ thư mục đó. smbclient cũng giống như một chương trình client ftp.

```
smbclient <tên dịch vụ> [-U <tên người dùng> ] [ -W <tên miền hoặc group> ] –L [<tên netbios>]
```

Trong đó:

- <tên dịch vụ> : là tên của dịch vụ muốn sử dụng, có dạng //Maychu/dichvu. ***Maychu*** là tên netbios của máy chủ cung cấp dịch vụ, còn ***dichvu*** là tên của dịch vụ muốn sử dụng. Chẳng hạn như //dulieu/setups, thì tên máy chủ cần truy nhập là dulieu, còn

setups là tên thư mục muốn tham chiếu đến. Ta cũng có thể sử dụng địa chỉ IP thay cho tên netbios dưới dạng //192.168.0.12/setups.

- U <tên người dùng> : là tên người dùng muốn sử dụng tài nguyên đó.
- W <tên miền hoặc group> : là tên miền hoặc group mà máy chủ đó thuộc vào.
- L <tên netbios> : là tên netbios của máy chủ ta muốn xem các dịch vụ mà máy chủ đó đang cung cấp.

Ví dụ, để xem thông tin về các thư mục chia sẻ của một máy đồng thời cùng với các thông tin về các máy trong miền, các máy miền khác ta dùng lệnh:

```
# smbclient -L 10.10.16.5 -U thanhnt -W httt
```

Thì máy sẽ hỏi ta mật khẩu ứng với người dùng trên, sau khi đánh đúng mật khẩu ta sẽ thu được kết quả:

```
added interface ip=10.10.16.23 bcast=10.10.255.255 nmask=255.255.0.0
Password:
Domain=ửCHTTTMPIứ OS=ửUnixứ Server=ửSamba 2.2.3aứ
 Sharename       Type  Comment
 ---------       ----  -------
 netlogon        Disk  Network Logon Service
 public    Disk  Public Stuff
 Source    Disk  Source and documents for vietseek
 IPC$            IPC   IPC Service (Samba Server)
 ADMIN$ Disk   IPC Service (Samba Server)
 thanhnt   Disk  Home Directories

 Server   Comment
 ---------  -------
 HTTT-23 Samba Server

 Workgroup            Master
 ---------            -------
 BCNK.FOTECH      VINHTQ
 BMVT             NGUYENHONG
 CHTTTMPI     HTTT-23
 ECC                  HUNGTN
 FOTECH               ANHNV
 FOTECH-CTSV      MAIPT
```

Để sử dụng một dịch vụ (một thư mục chia sẻ chẳng hạn) ta có thể dùng lệnh như sau:

```
# smbclient //10.10.16.5/setup -U thanhnt -W httt
```

Trong trường hợp này ta sẽ được máy hỏi mật khẩu, nếu thành công thì nó sẽ cho ta một phiên làm việc với dịch vụ đó, cụ thể ta sẽ được một phiên làm việc với thư mục, ta có thể sao chép file ở trên thư mục này vào máy hiện tại và ngược lại.

```
added interface ip=10.10.16.23 bcast=10.10.255.255
nmask=255.255.0.0
Password:
Domain=[CHTTTMPI] OS=[Unix] Server=[Samba 2.2.3a]
```

```
smb: \>
smb: \> ls
 .           D    0    Tue Sep 11 12:03:53 2001
 ..          D    0    Tue Sep 11 12:03:53 2001
 ee120-ta    D    0    Wed Aug 29 09:37:14 2001
 fa01        D    0    Fri Sep 21 09:47:34 2001

 60472 blocks of size 2097152. 52606 blocks available
 smb: \> cd fa01\ee120-kmm
 smb: \> put hello.p [send files from local to remote]
 smb: \> get interruptq.doc [receive files to local from
remote]
 smb: \> quit
```

Khi dấu nhắc hiện ra, để xem các lệnh thao tác, ta có thể đánh lệnh help. Sau khi kết thúc phiên làm việc, ta dùng lệnh ***quit*** để thoát.

Kết gắn một thư mục chia sẻ vào một thư mục trong hệ thống file hiện tại: Trong trường hợp ta không muốn dùng các lệnh smbclient cho từng phiên làm việc khi mà ta sẽ có nhiều thao tác với thư mục được chia sẻ đó, giải pháp tốt nhất là kết gắn thư mục chia sẻ đó vào thành một thư mục ở trên máy cục bộ. Khi đó thư mục được kết gắn sẽ trở thành một thư mục bình đẳng như các thư mục trên máy cục bộ. Mọi việc thao tác sẽ trở nên thuận tiện hơn rất nhiều. Để làm điều đó ta dùng lệnh (với quyền root):

```
#smbmount //10.10.16.5/setup /mnt/smb –o username=thanhnt
```

hoặc

```
#mount –t smbfs //10.10.16.5/setup /mnt/smb –o
username=thanhnt
```

Khi đó máy sẽ hỏi mật khẩu, khi thành công thì ta sẽ ánh xạ được thư mục chia sẻ setup trên máy 10.10.16.5 thành một thư mục /mnt/smb trên máy của mình. Khi nào xong ta có thể bỏ kết gắn đó bằng lệnh:

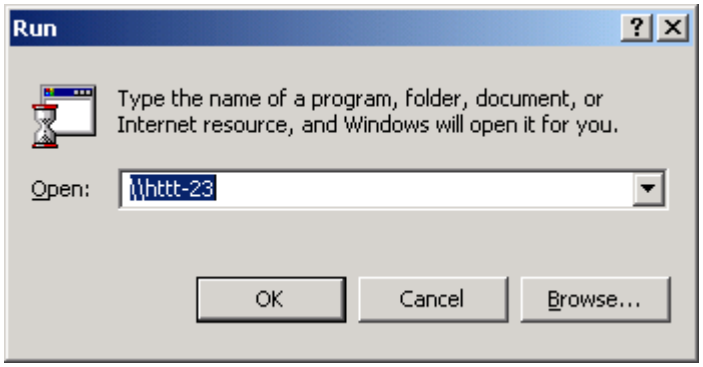
```
#smbmount /mnt/smb
```

hoặc

```
#umount /mnt/smb
```

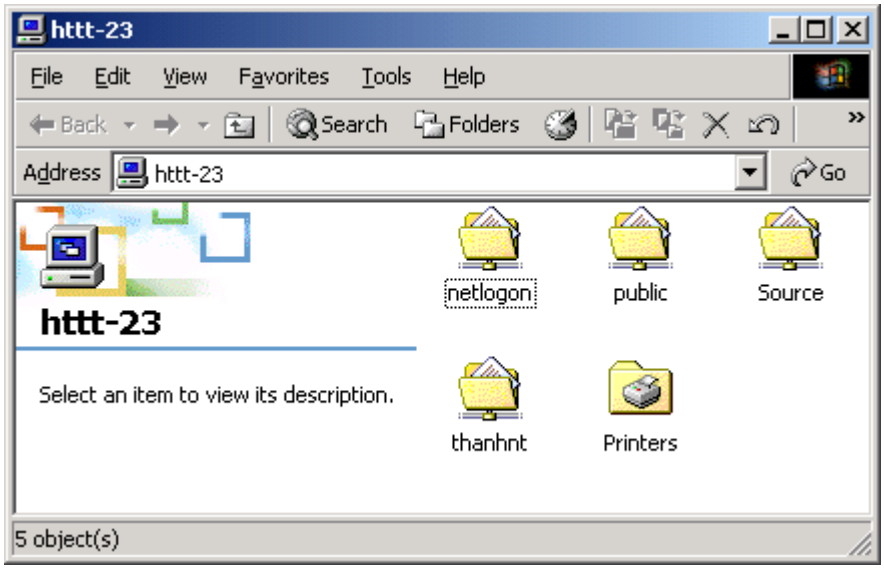
D.6.2 Cách sử dụng từ các máy trạm là Windows

Ta chọn menu start, ta chọn run, sau đó đánh vào tên máy mà ta muốn sử dụng một dịch vụ nào đó như trên hình D.2.



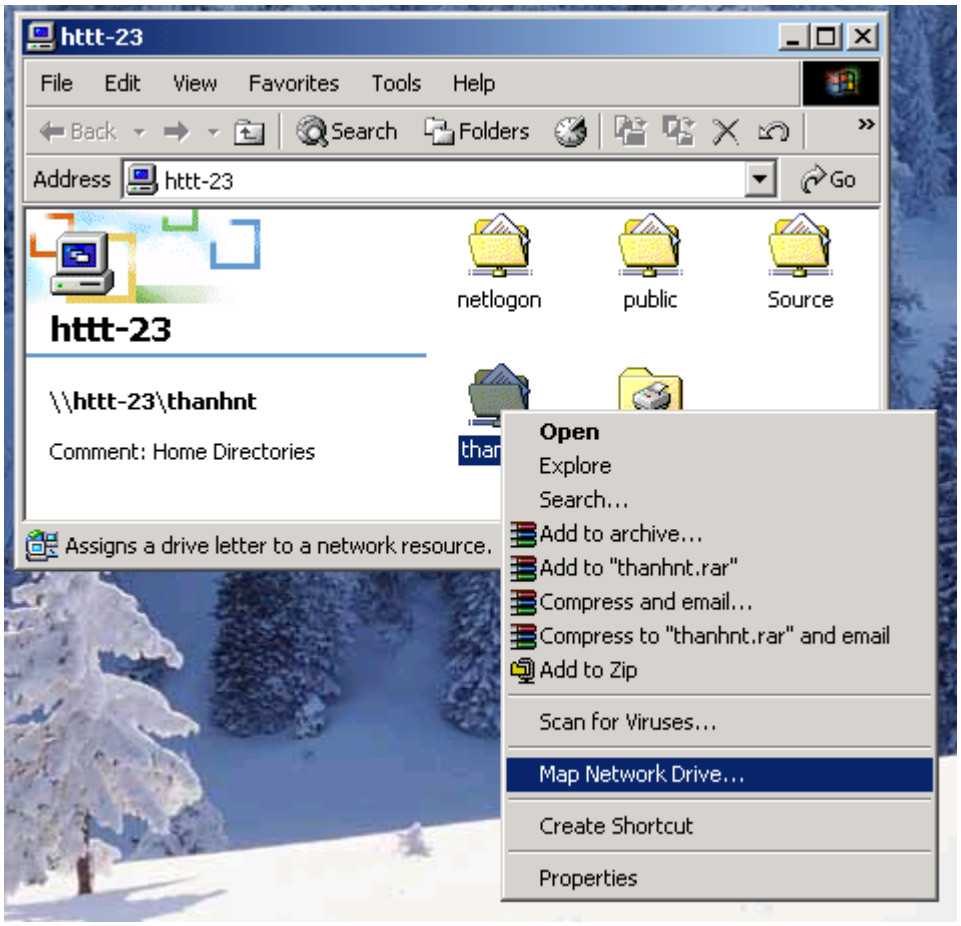
Hình D.2 Sử dụng dịch vụ samba từ máy trạm Windows

Sau đó máy sẽ hỏi ta tên người dùng và mật khẩu dùng để truy cập. Sau khi nhập đủ các thông tin, nếu thành công thì ta sẽ được một cửa sổ hiển thị danh sách các dịch vụ của máy chủ samba đó cung cấp như hình D.3.



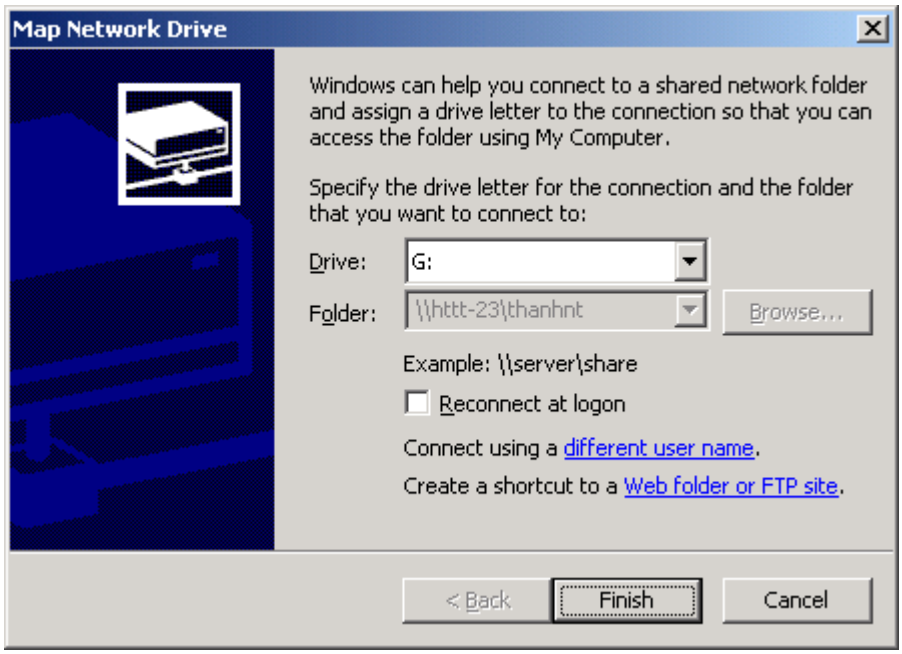
Hình D.3 Danh sách các dịch vụ trên một samba server

Cho phép ánh xạ một thư mục trên một samba server thành một ổ đĩa trêm máy trạm Windows bằng cách trên cửa sổ hiện danh sách các tài nguyên trên ta nháy phải chuột vào thư mục ta muốn ánh xạ, sau đó chọn "Map network drive" như trên hình D.4.



Hình D.4 Tạo một ánh xạ ổ đĩa trên máy trạm Windows

Sau đó máy sẽ hỏi tên ổ đĩa mà ta muốn đặt cho ổ mới này như hình D.5:



Hình D.5 Đặt tên ổ đĩa cho một ánh xạ ổ đĩa

CHƯƠNG 1

GIỚI THIỆU
HỆ ĐIỀU HÀNH LINUX

Nội dung

1. Giới thiệu về Linux.

2. Lịch sử phát triển của Linux.

3. Điểm khác biệt của Linux.

4. Những phiên bản của Linux.

- RedHat Linux.
- Slackware.
- Caldera Open Linux.
- Su.S.E Linux.
- Debian.

5. Các đặc tính cơ bản của Linux.

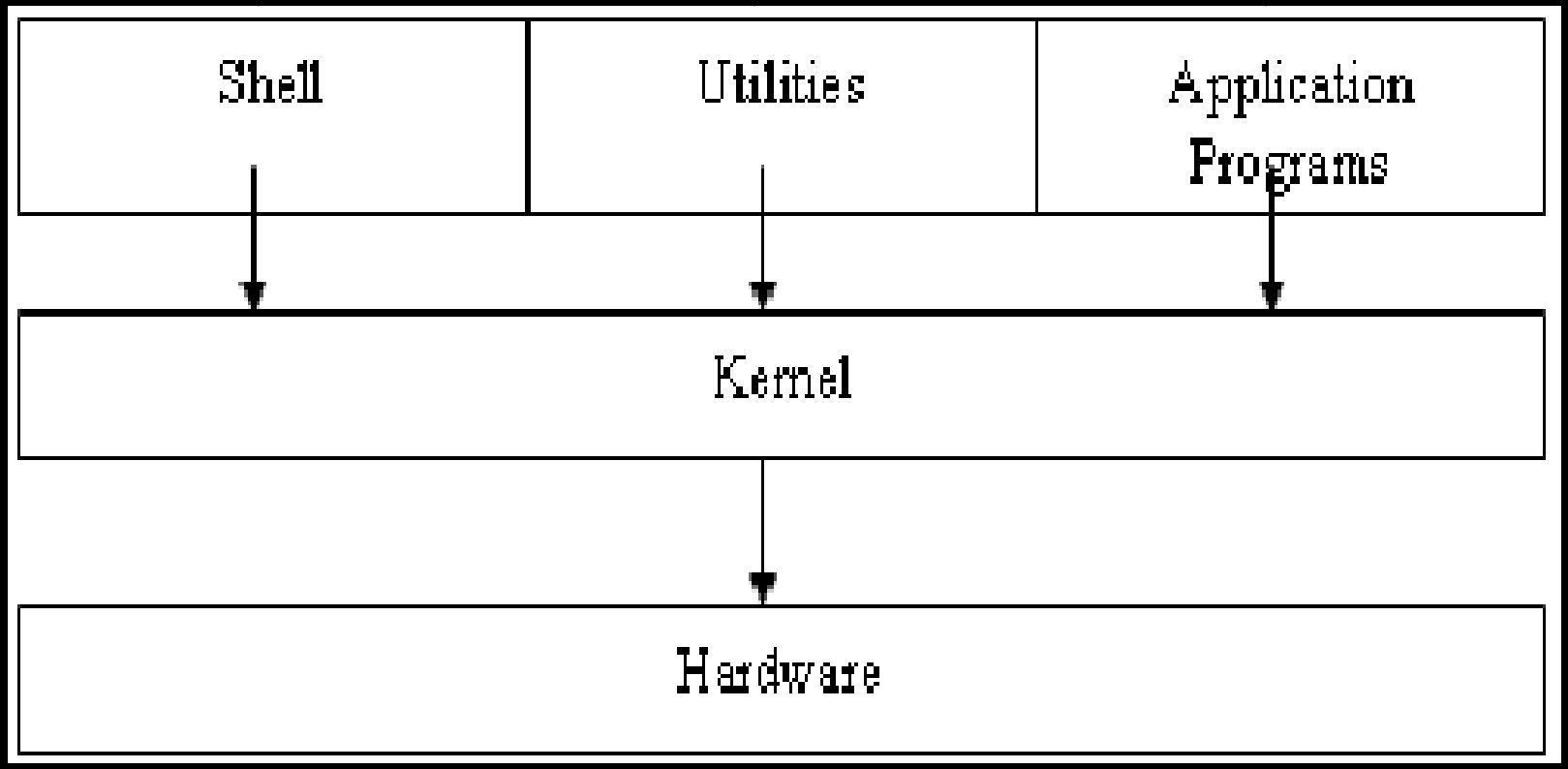
Các đặc tính cơ bản ... (tt)

6. Các khuyết điểm của Linux.

Thiếu trợ giúp kỹ thuật.

Các vấn đề về phần cứng.

7. Kiến trúc của Linux.



8. Linux khác UNIX như thế nào?

9. So sánh Linux với Windows NT.

CHƯƠNG 2

CÀI ĐẶT HỆ ĐIỀU HÀNH LINUX

Nội dung

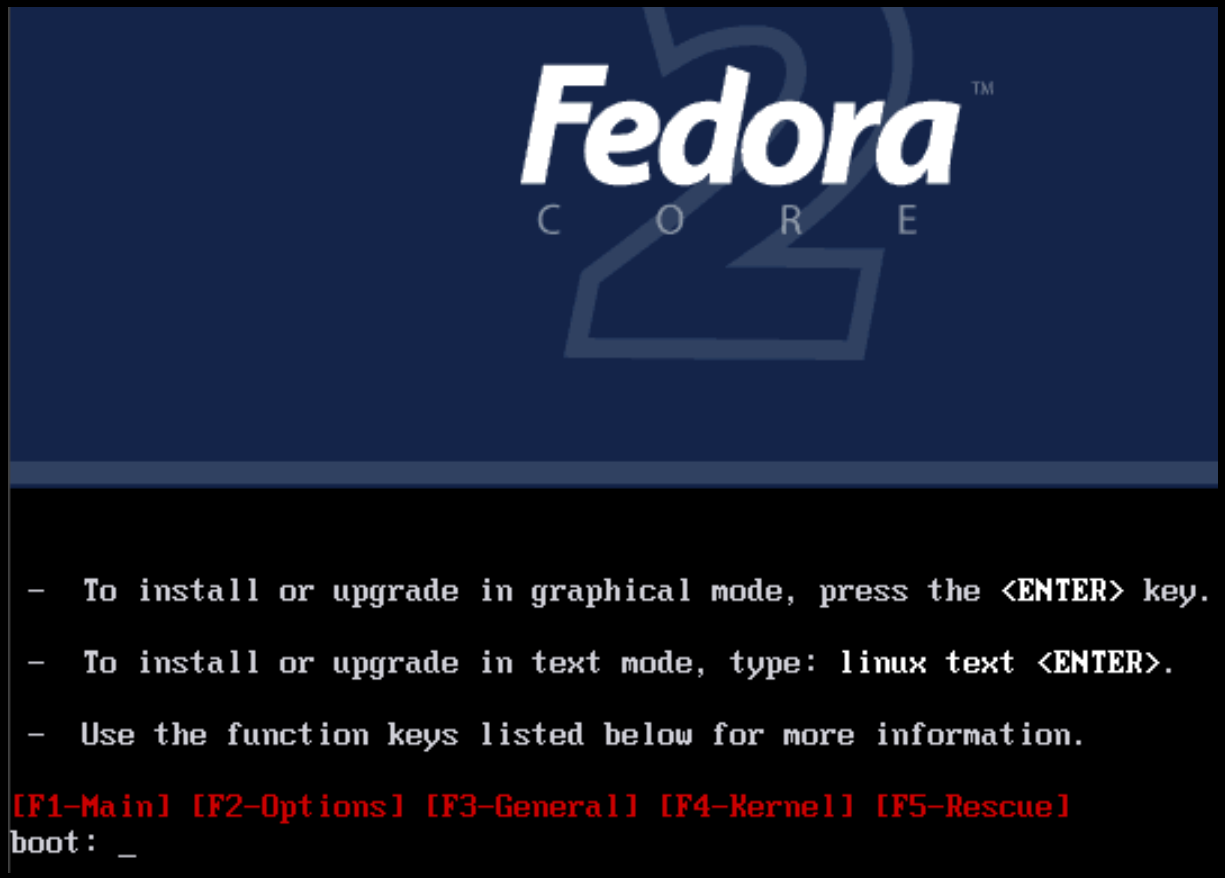
1. Tổng quan.

2. Những chuẩn bị trước khi cài đặt Linux.

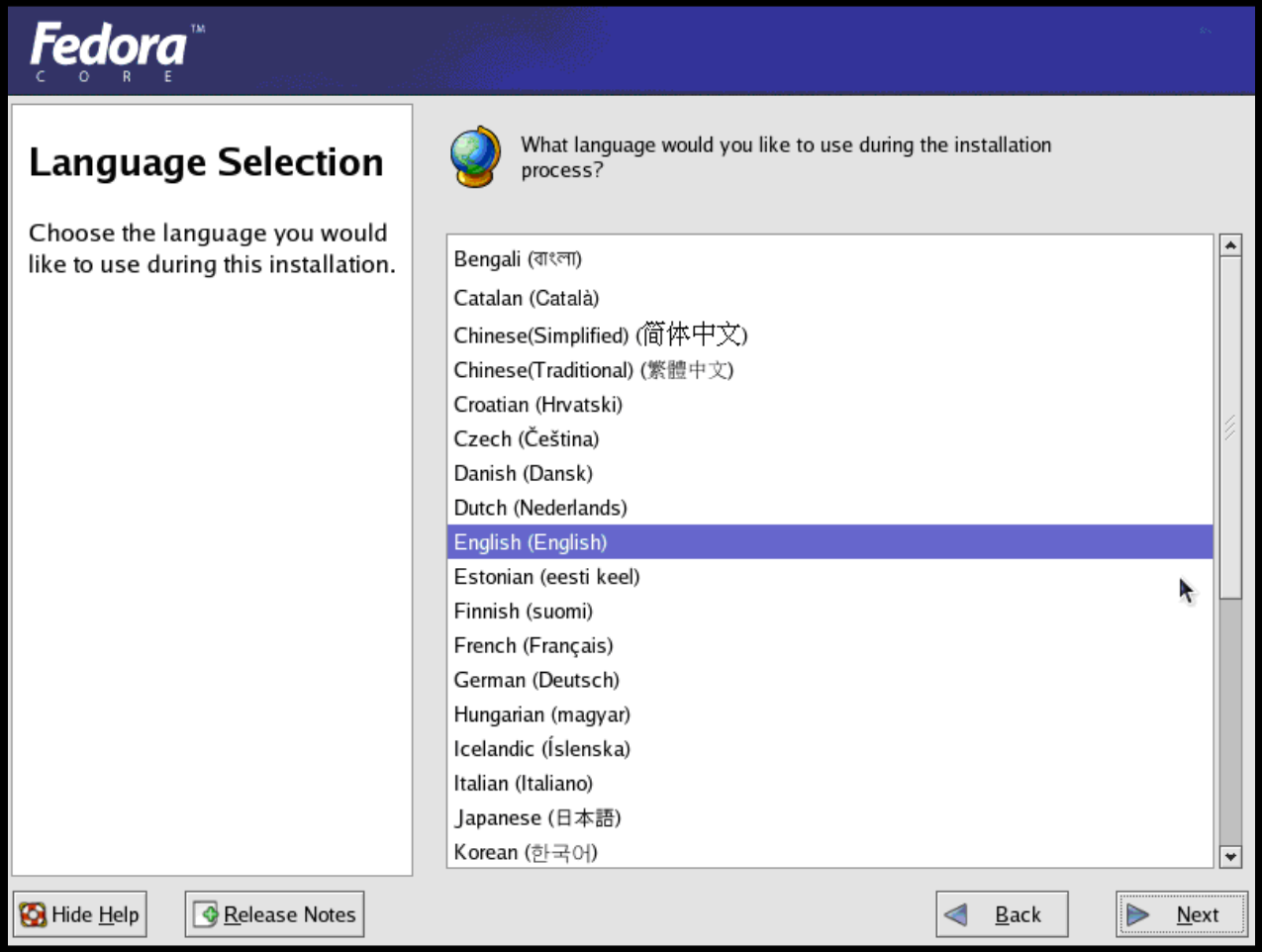
3. Quá trình cài đặt.

-
-
-

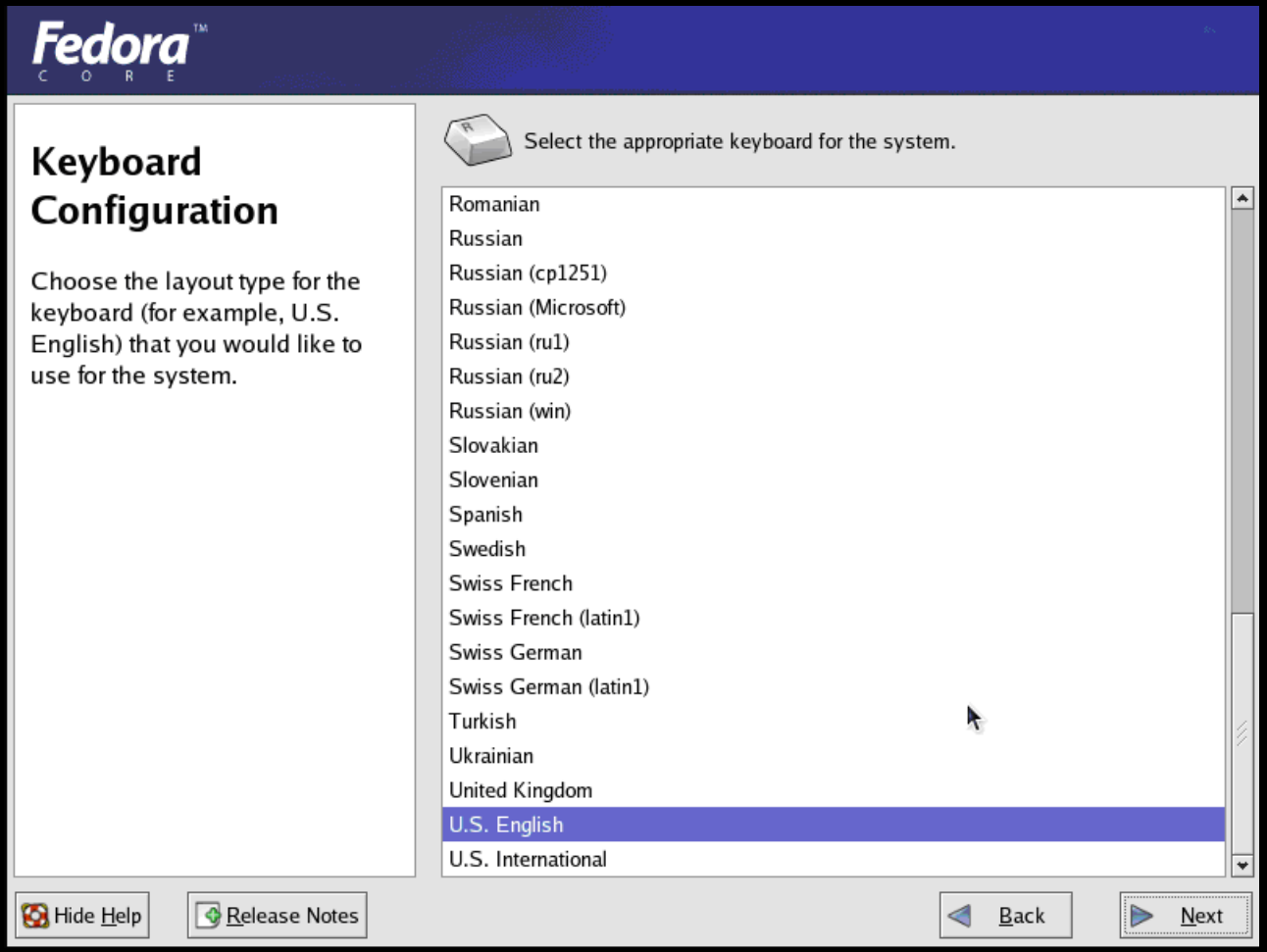
Quá trình cài đặt (tt)



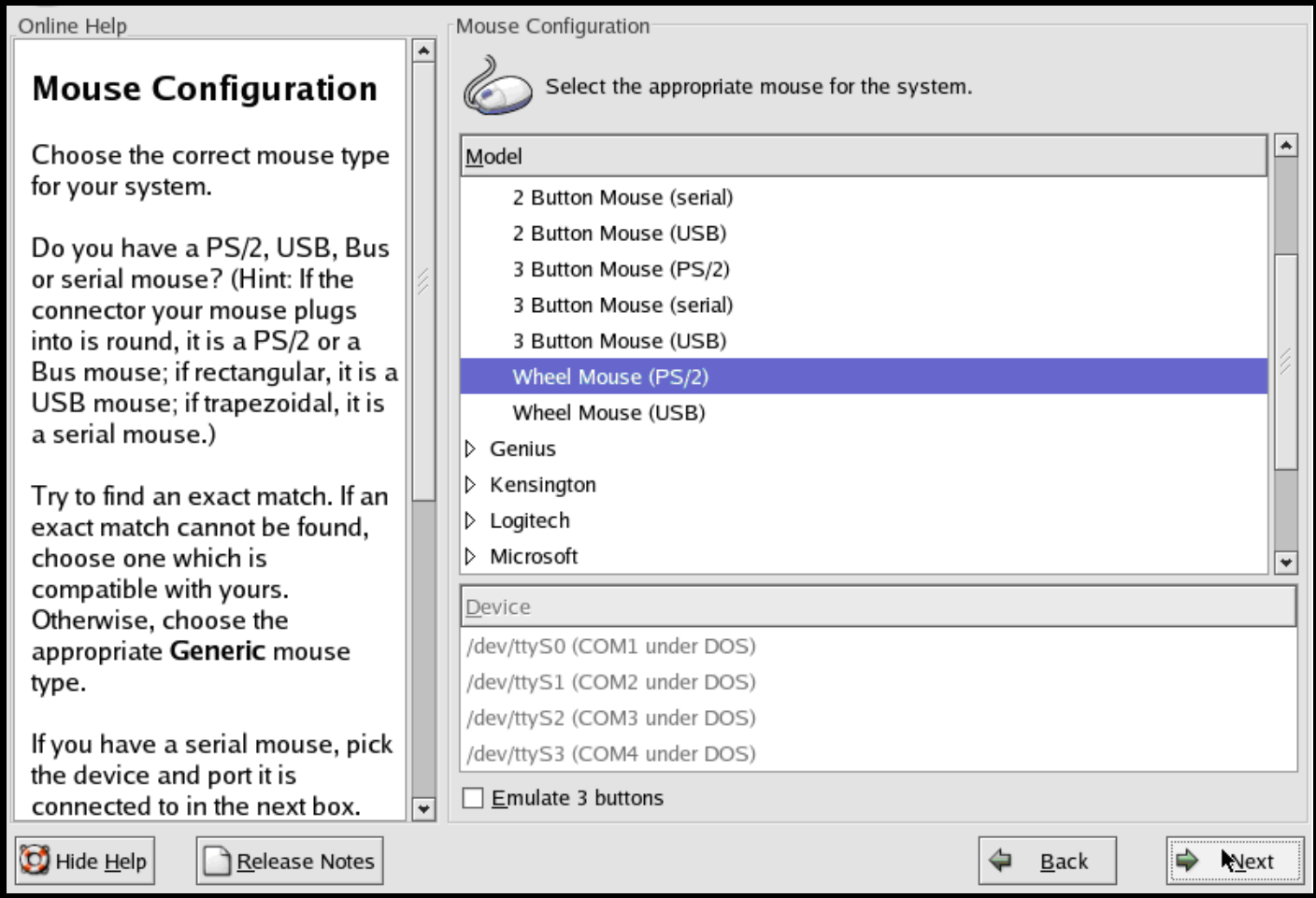
Quá trình cài đặt (tt)



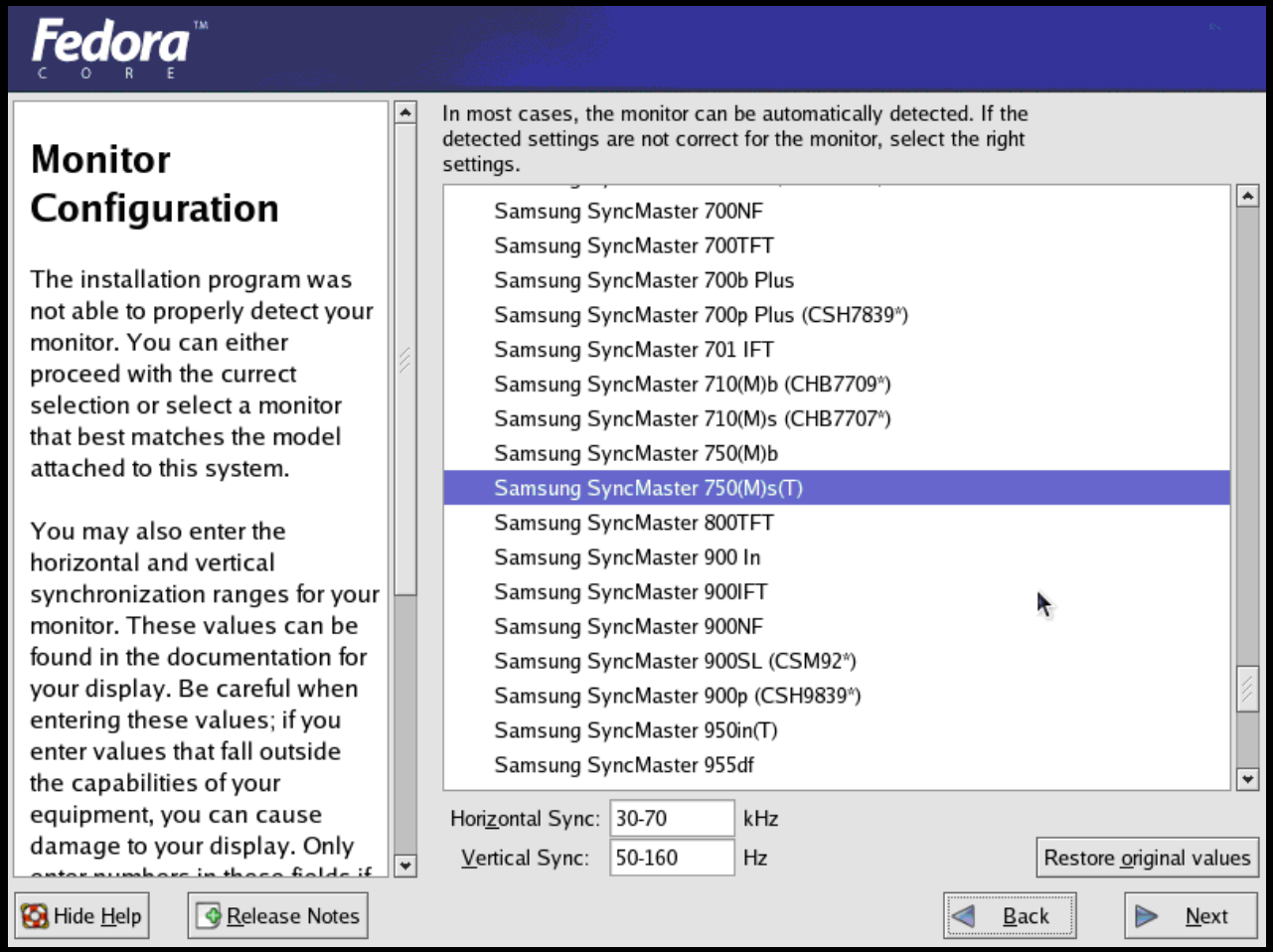
Quá trình cài đặt (tt)



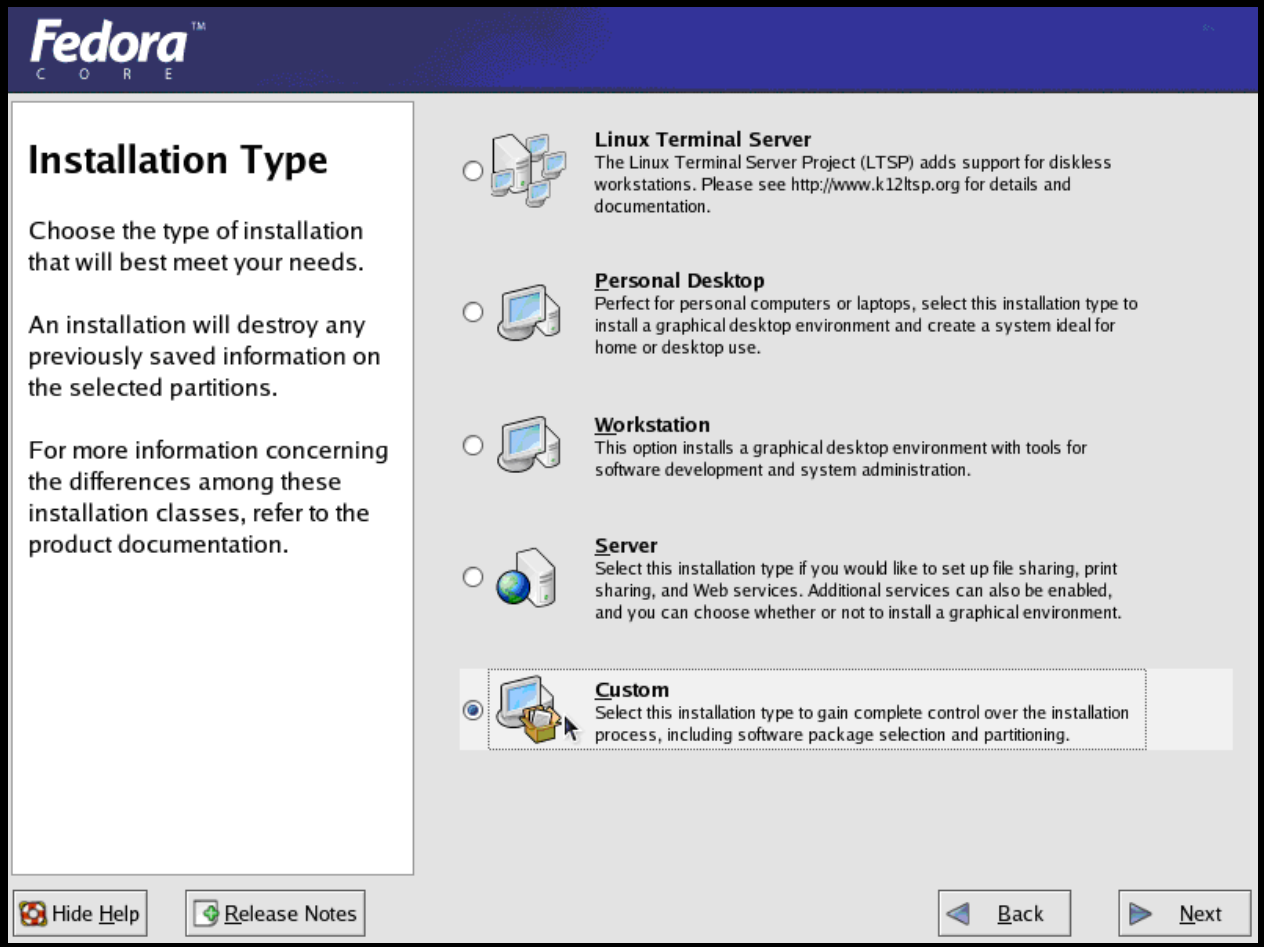
Quá trình cài đặt (tt)



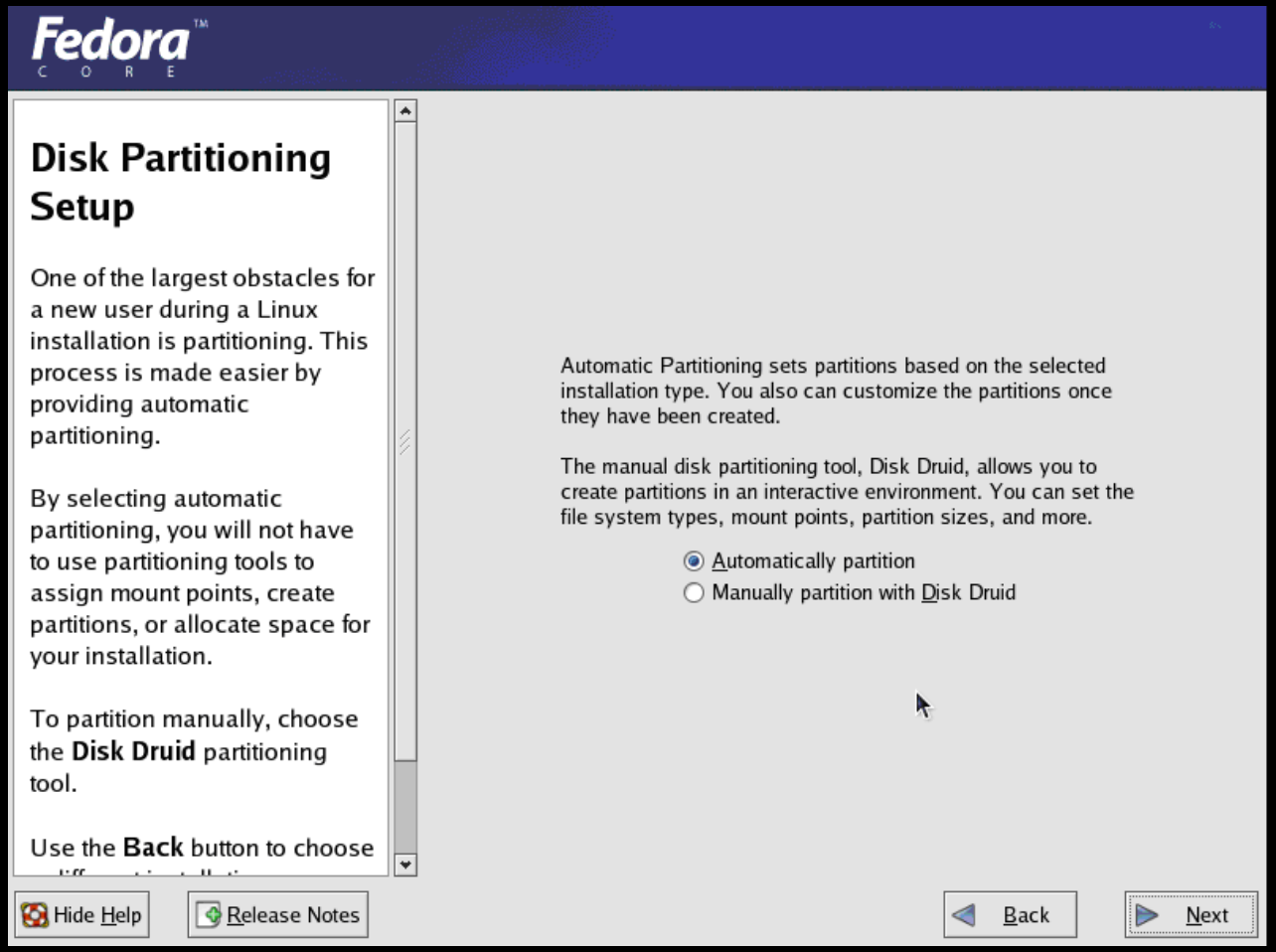
Quá trình cài đặt (tt)



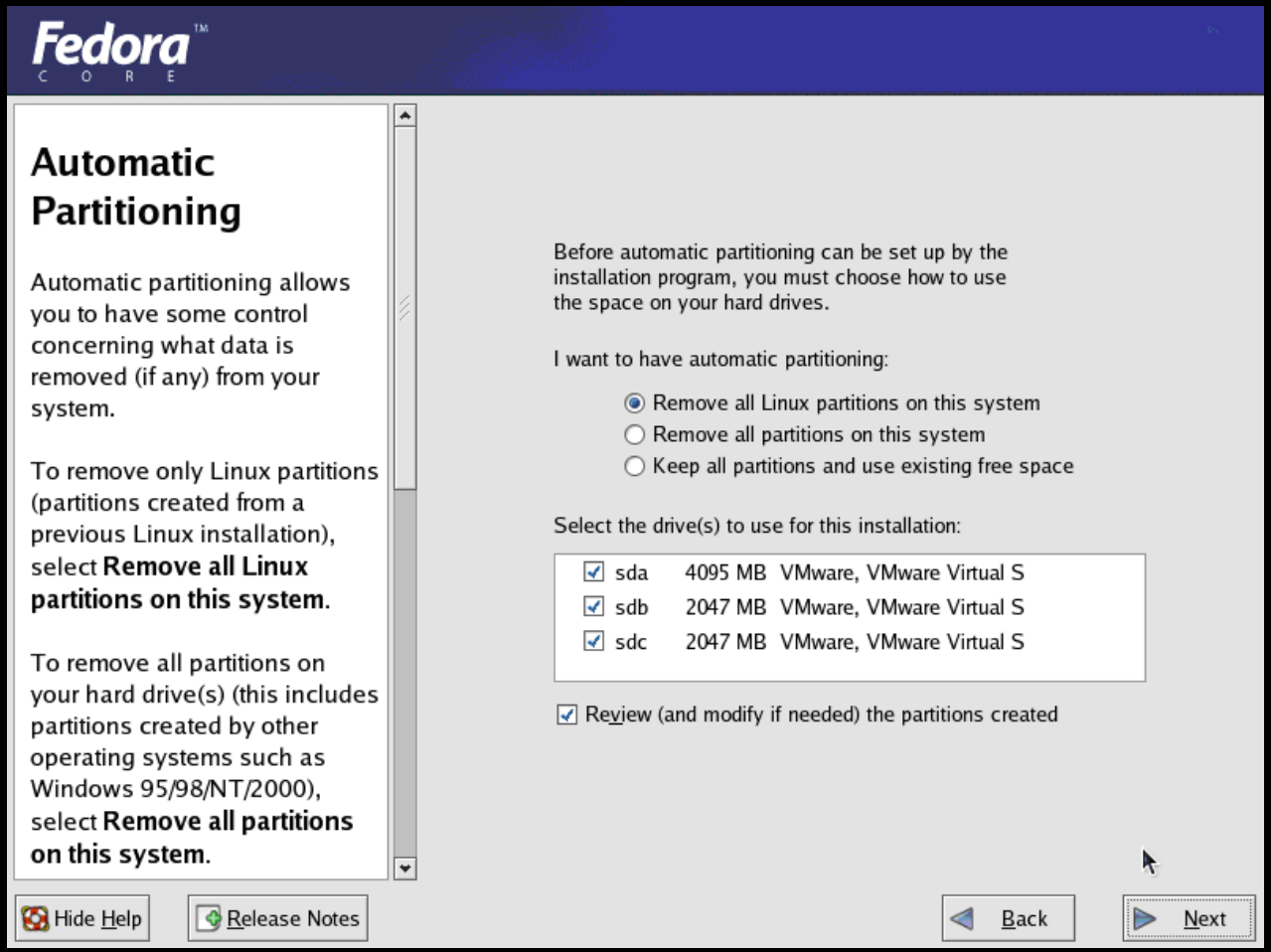
Quá trình cài đặt (tt)



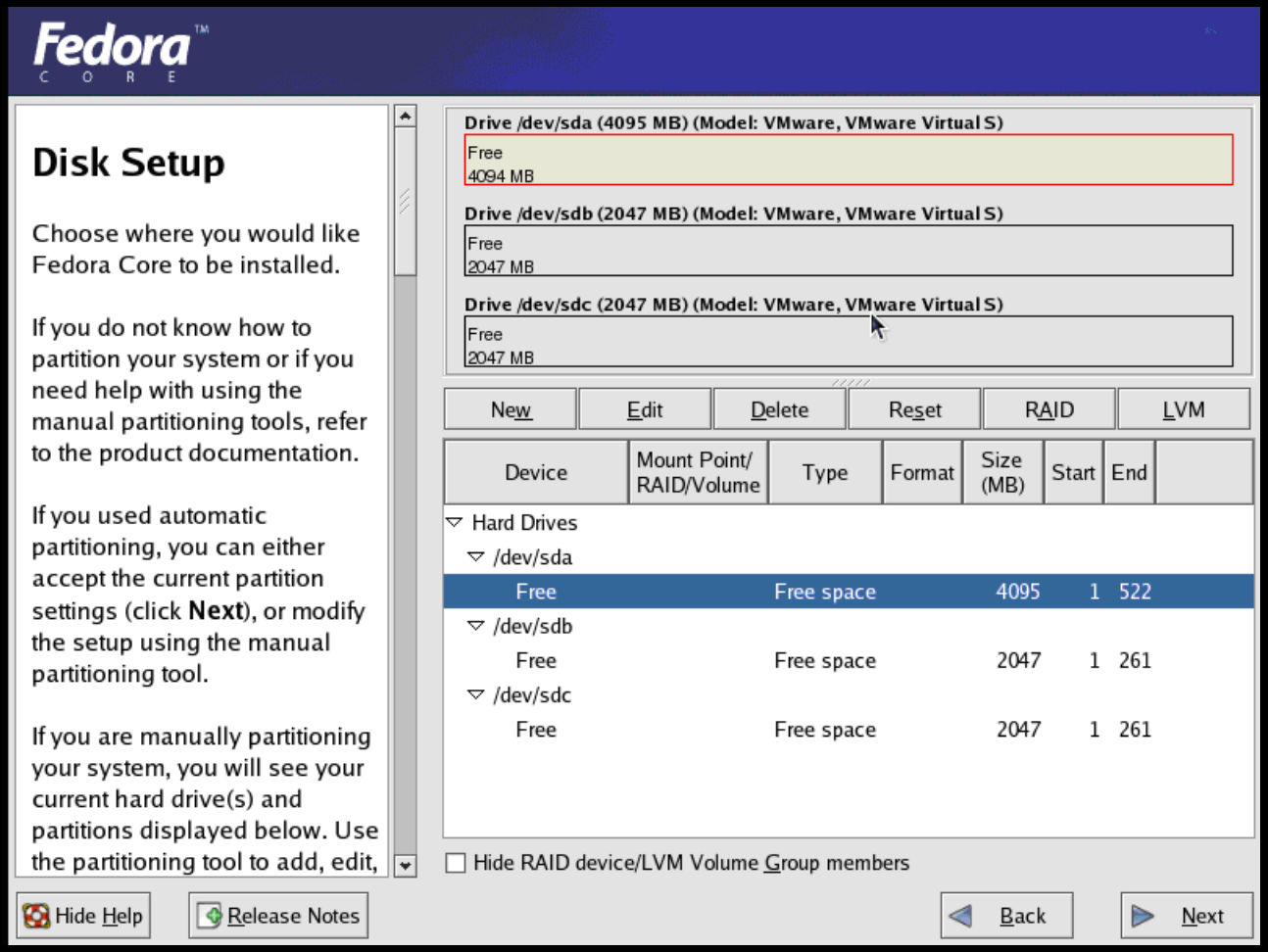
Quá trình cài đặt (tt)



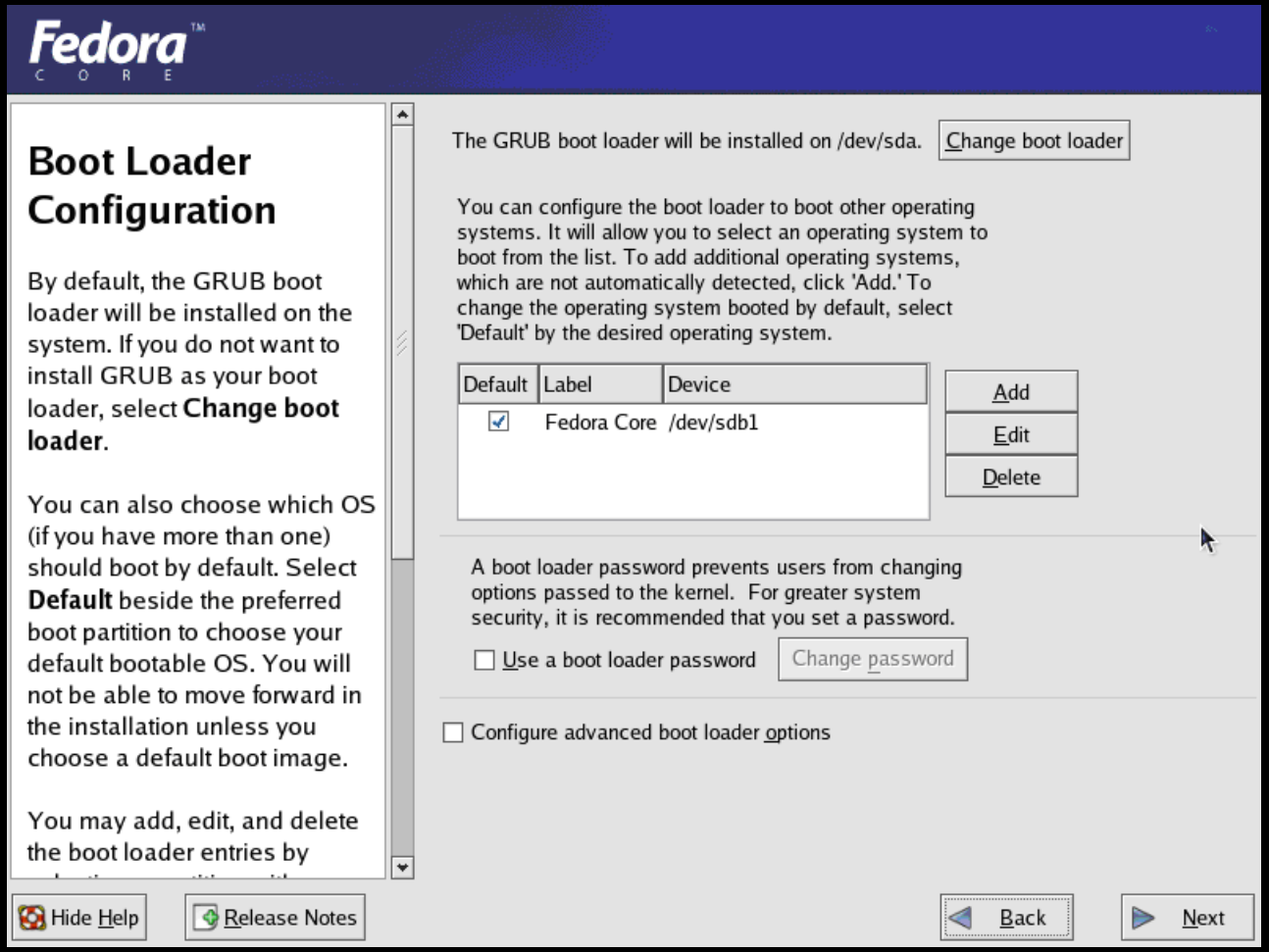
Quá trình cài đặt (tt)



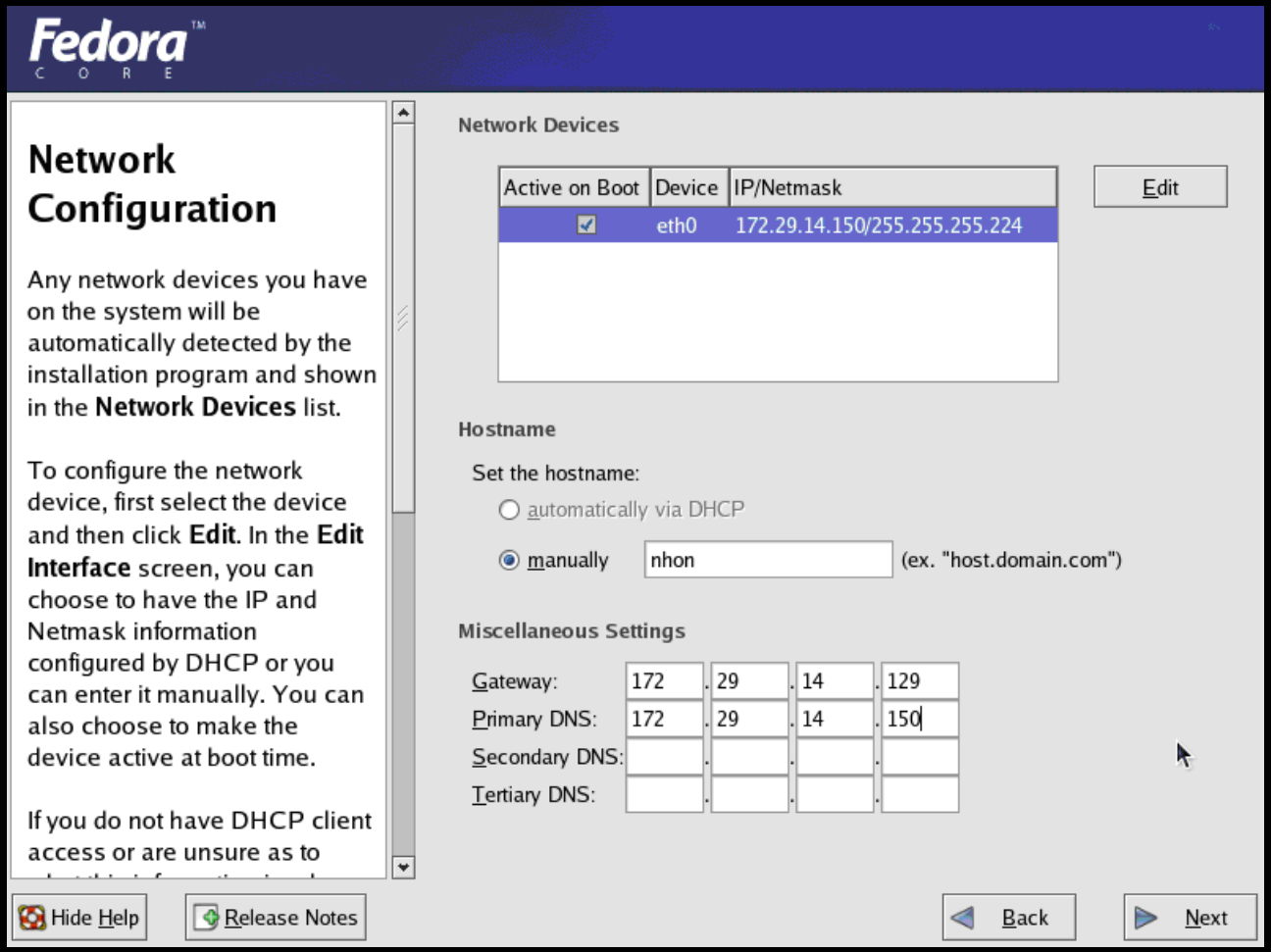
Quá trình cài đặt (tt)



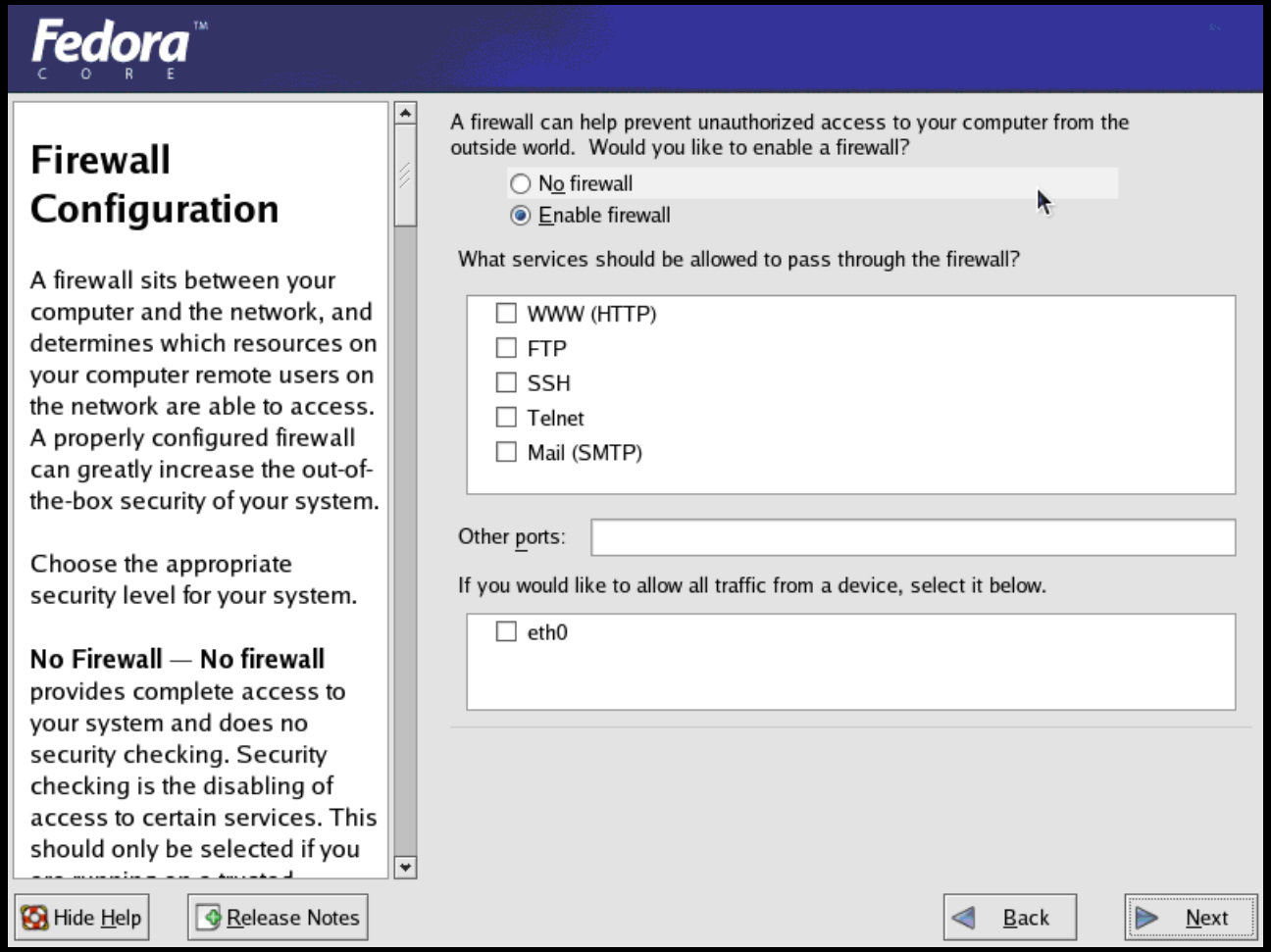
Quá trình cài đặt (tt)



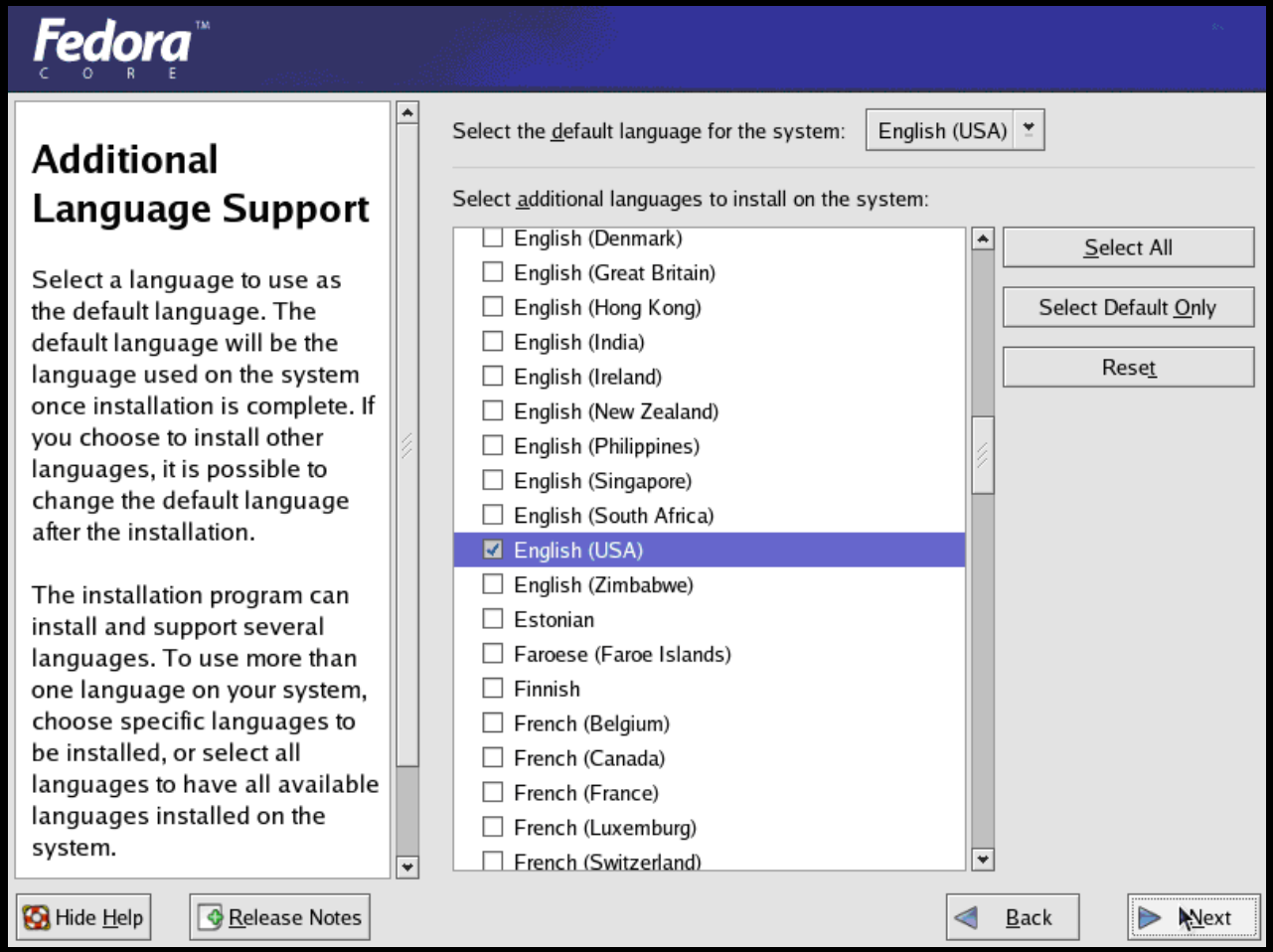
Quá trình cài đặt (tt)



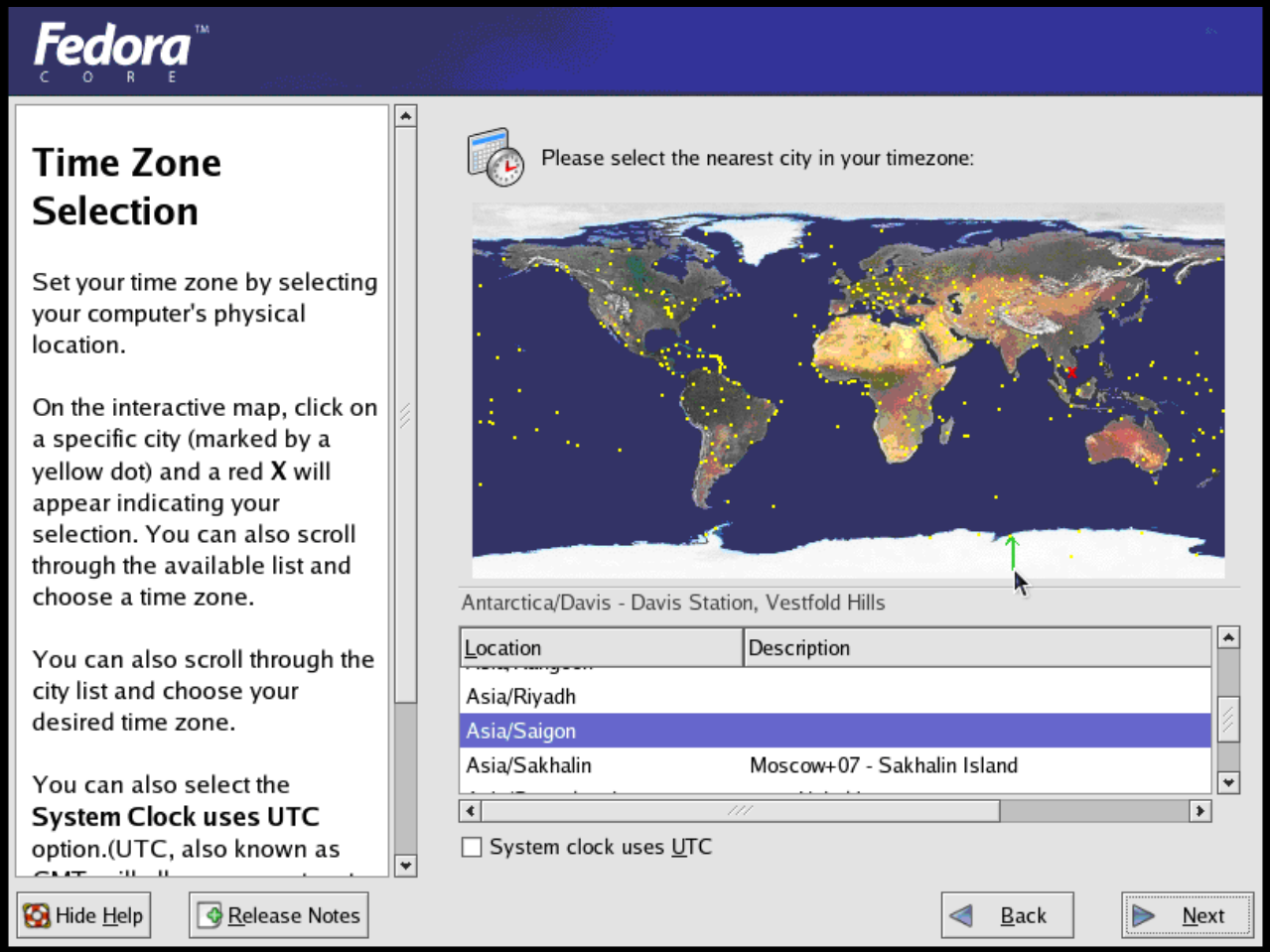
Quá trình cài đặt (tt)



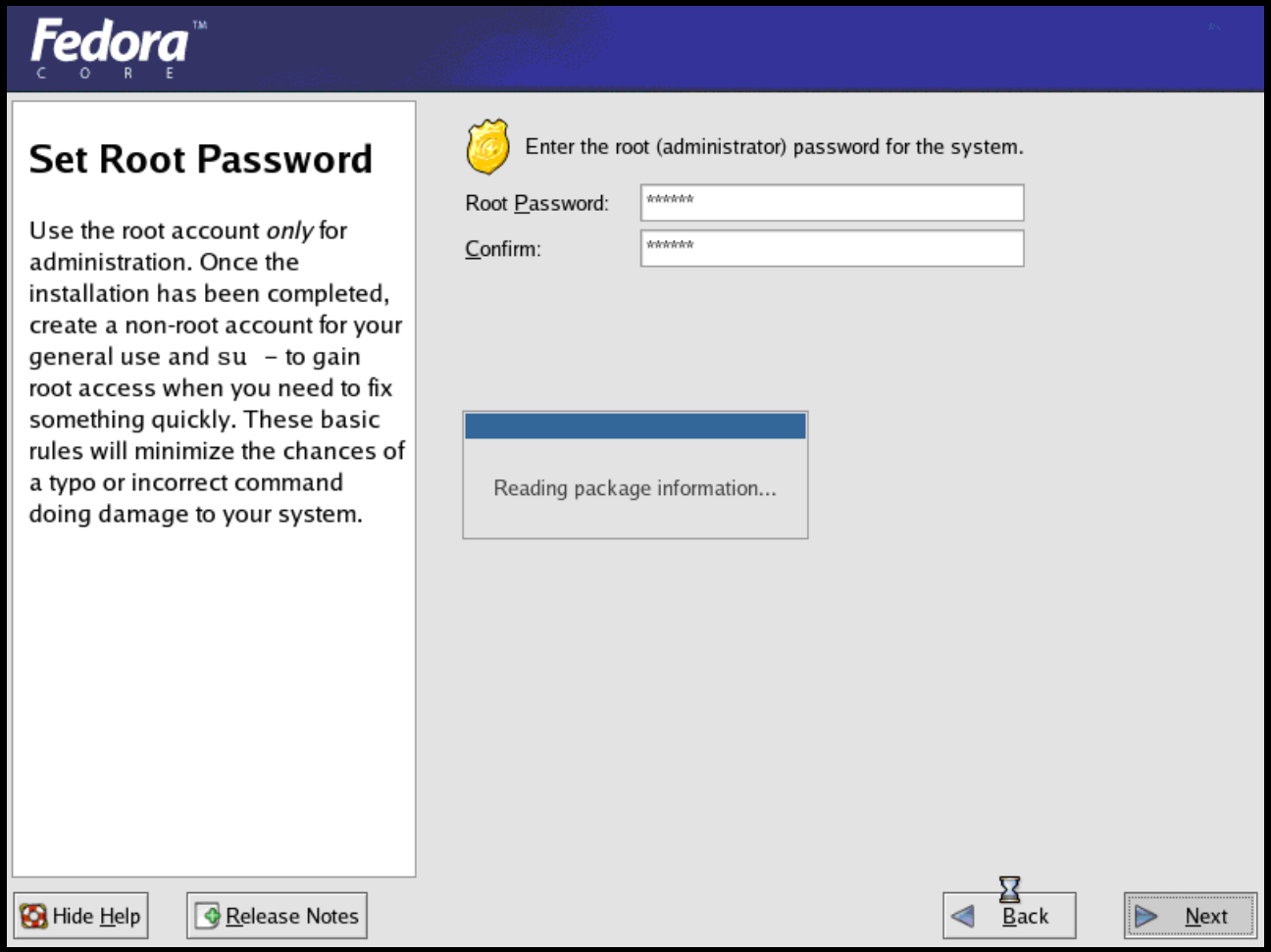
Quá trình cài đặt (tt)



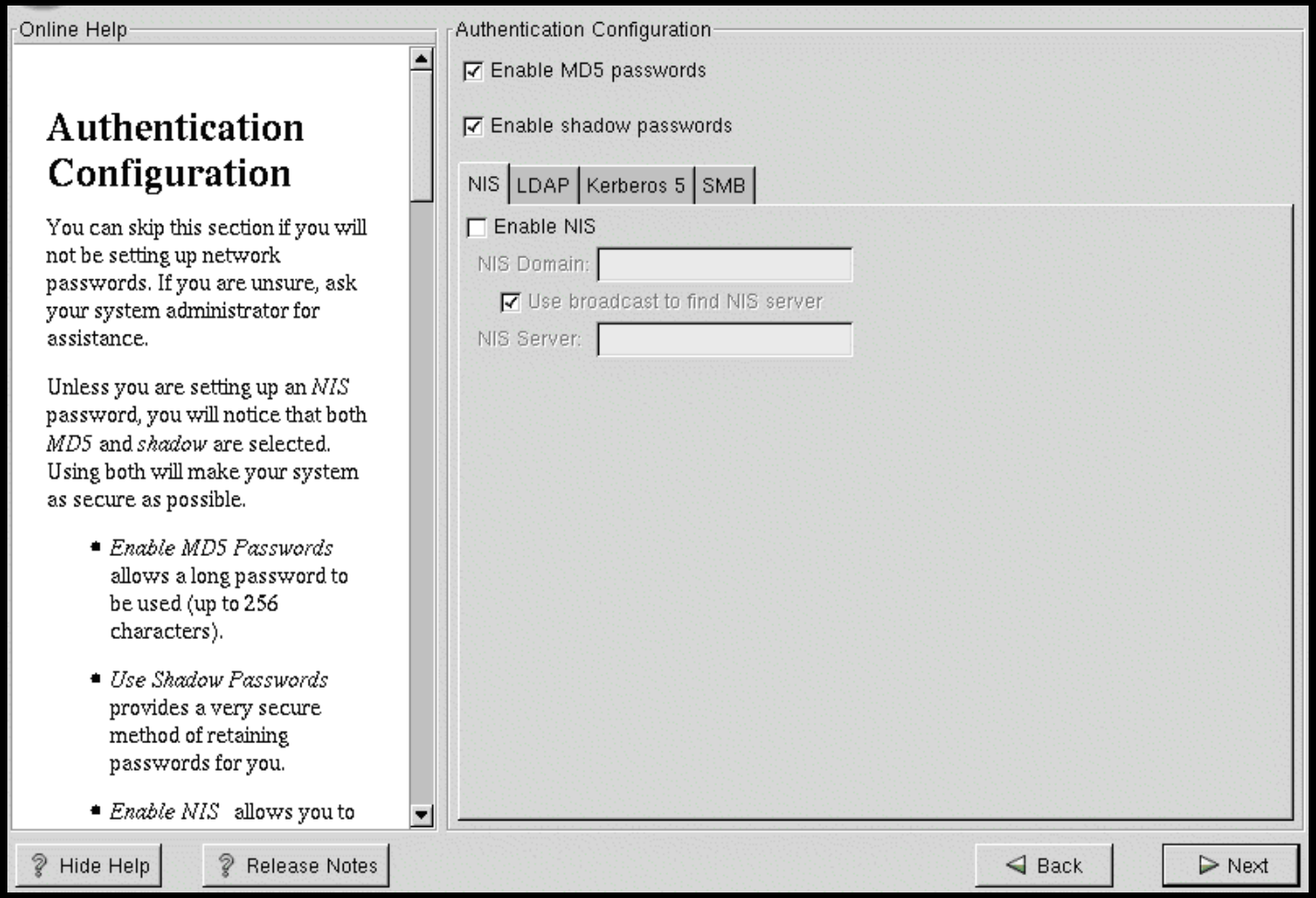
Quá trình cài đặt (tt)



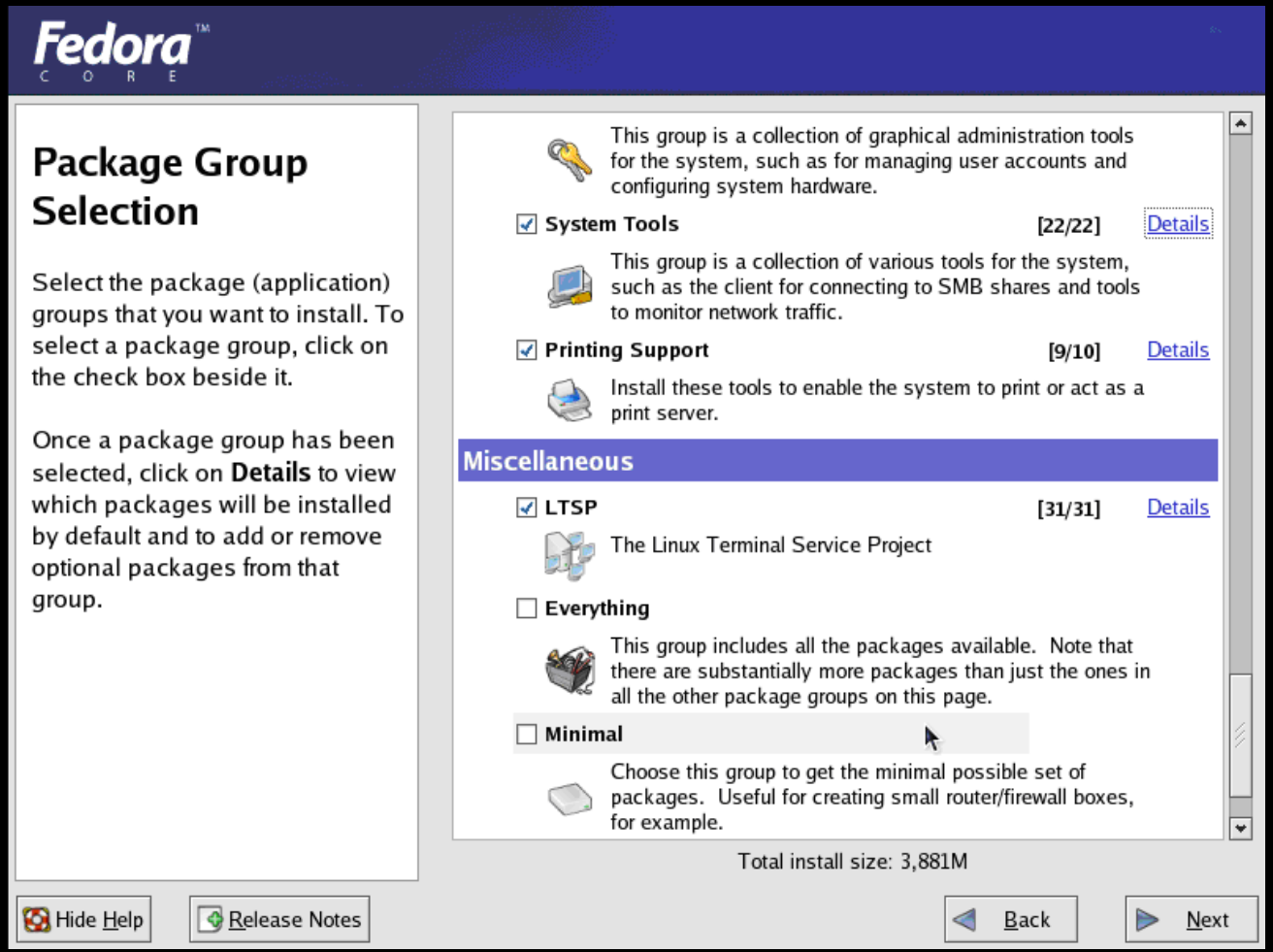
Quá trình cài đặt (tt)



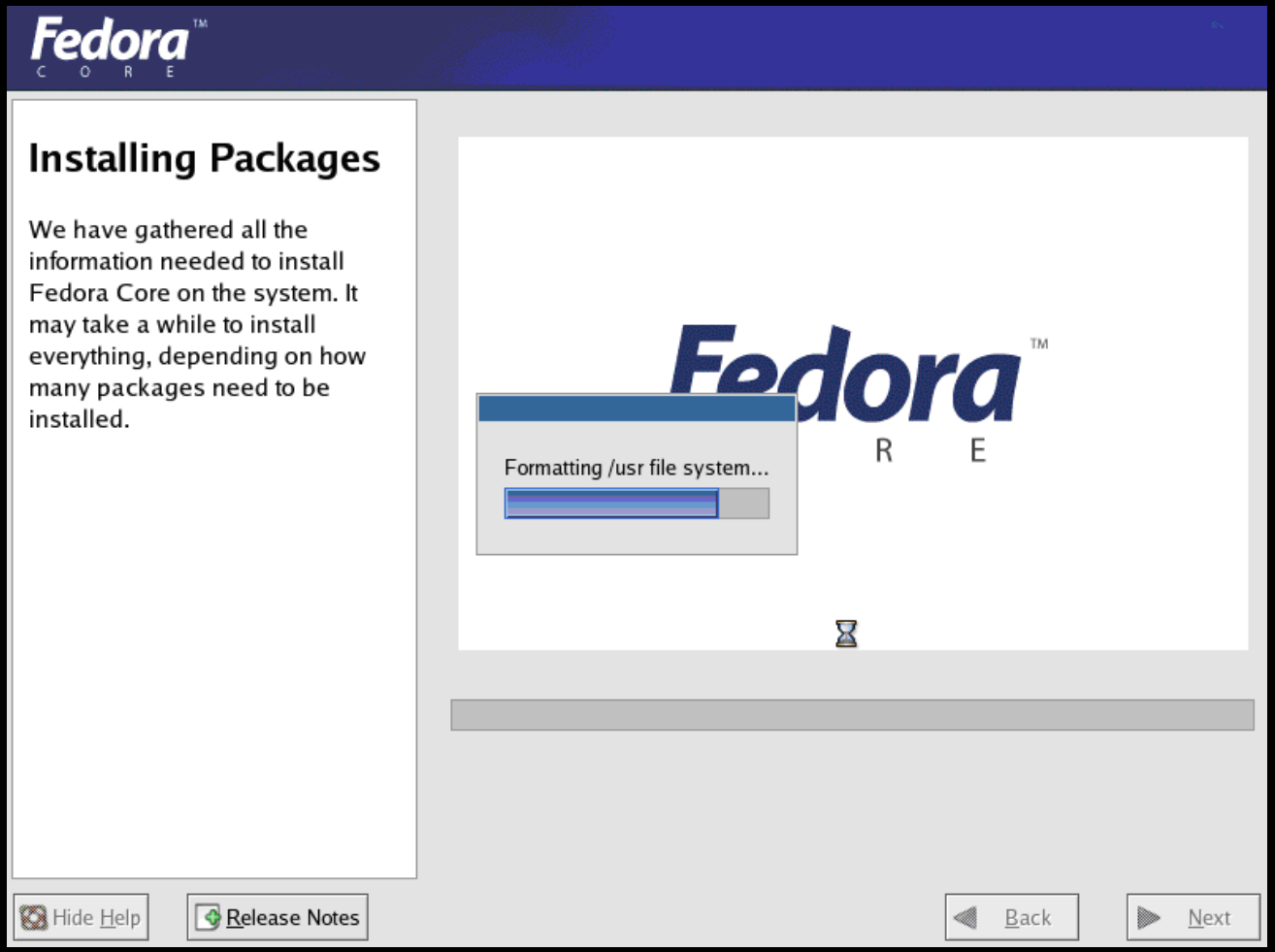
Quá trình cài đặt (tt)



Quá trình cài đặt (tt)



Quá trình cài đặt (tt)



4. Login và Logout.

Login:

Password:

[tênđăngnhập@tênmáy thưmục]dấunhắclệnh

$

#

exit logout

5. Cú pháp lệnh.

Command [options] [paramaters]

6. Những lệnh thông thường.

- who
- tty
- date
- cal
- finger
- chfn
- head
- tail
- w

Những lệnh thông thường (tt)

passwd

su

#su [-] [tên-user]

-

Man

#man [tên lệnh]

Hostname

7. Các mức hoạt động của hệ thống.

Init 6

Init 5

Init 4

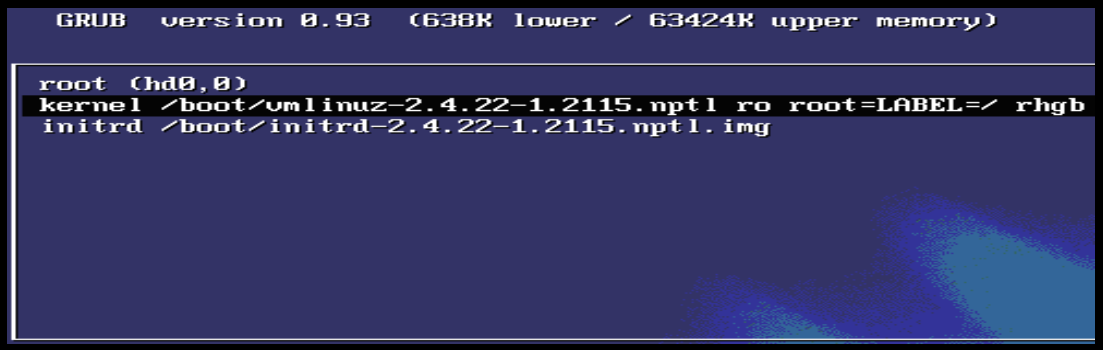
Init 3

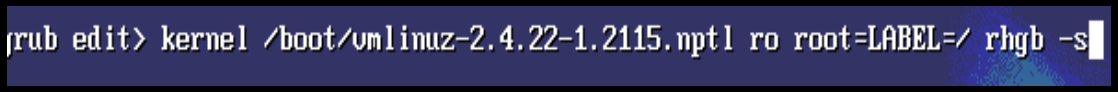
Init 2

Init 1

Init 0

8. Phục hồi mật khẩu cho user quản trị.





9. LILO & GRUB.

/sbin/grub-install <tên_ổ_đĩa>

lilo.conf.anaconda lilo.conf

lilo

GRUB là một boot manager

/etc/grub/grub.conf

LILO & GRUB (tt)

LILO & GRUB (tt)

GRUB

password --md5 <Password>

10. KHỞI ĐỘNG HỆ THỐNG.

CHƯƠNG 3

HỆ THỐNG TẬP TIN

Nội dung

Nội dung (tt)

1. Filesystem là gì?

2. Khái niệm về thiết bị.

Block device

Character device

Khái niệm về thiết bị (tt)

3. Partition.

fdisk

Partition (tt)


```
# fdisk /dev/hda
Command (m for help) : n
e extended
p primary partition (1-4)
p
Partition number (1-4):4
First cylinder (523-525, default 523); <ENTER>
Using default value 523
Last cylinder or +size or +sizeM or +sizeK (523-525,
   default 525):<ENTER>
Using default value 525
```

Partition (tt)


```
Command (m for help) : p
Disk /dev/hda: 255 heads, 63 sectors, 525 cylinders
Units = cylinders of 16065 * 512 bytes
Device    Boot  Start   End   Blocks   Id   System
/dev/hda1   *      1    261  2096451    6   FAT16
/dev/hda2          262   503  1943865   83   Linux
/dev/hda3          504   522   152617+  82   Linux swap
/dev/hda4          523   525    24097+  83   Linux
```

4. Định dạng partition.

mkfs –t <fstype> <filesystem>

mkfs –t ext2 /dev/hda1

5. Những khái niệm cơ bản về filesystem.

- Superblock
- Inode
- Storageblock

Những khái niệm ... (tt)

Superblock

Inode

Những khái niệm ... (tt)

Storageblock

- Datablock của tập tin

- Datablock của thư mục

Những khái niệm ... (tt)

- Tập tin dữ liệu :

- Thư mục :

- Tập tin thiết bị :

Những khái niệm ... (tt)

- Link (Liên kết)

ln [-s] <source> <destination>

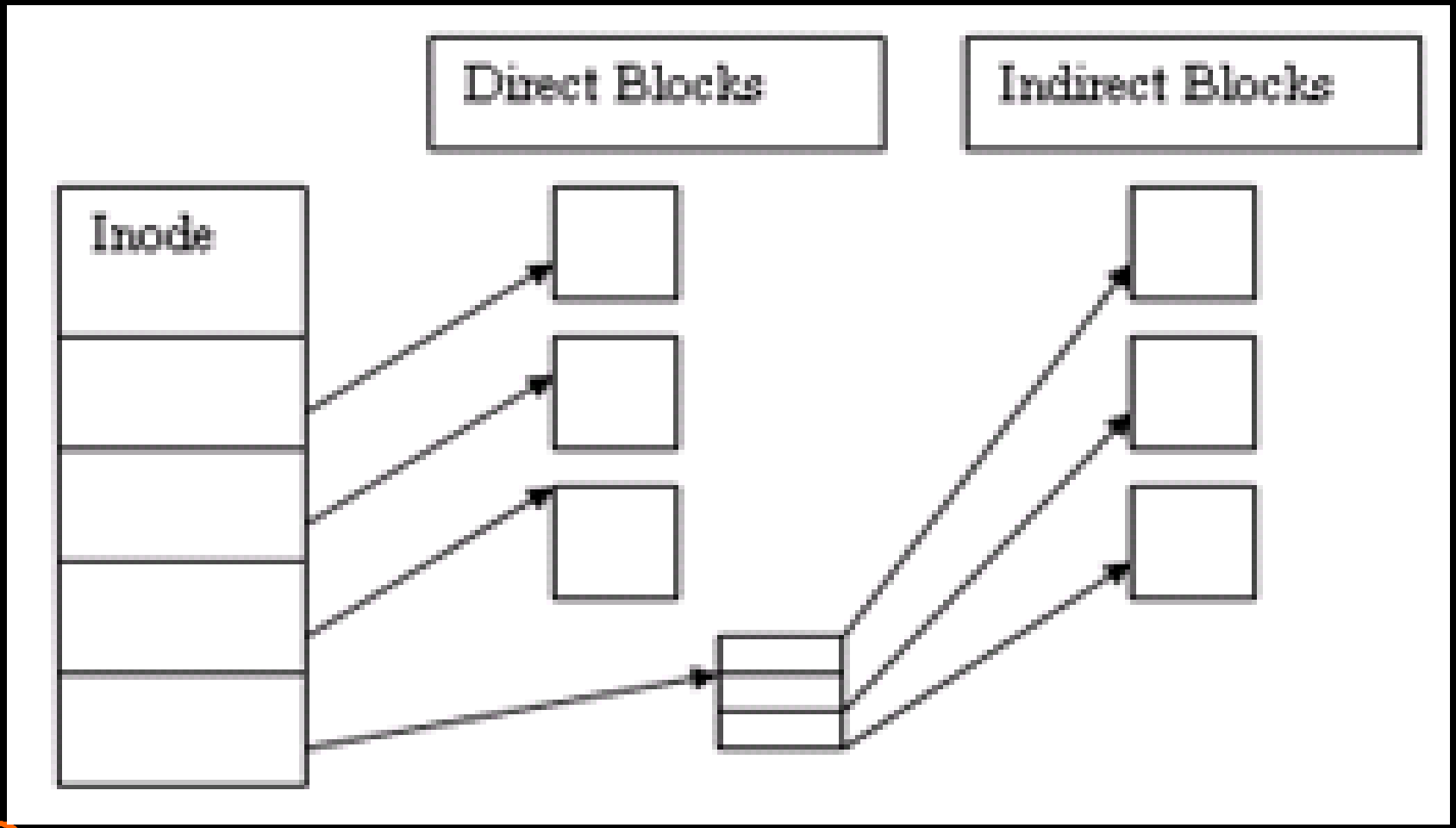
+ Hard Link

+ Symbolic Link

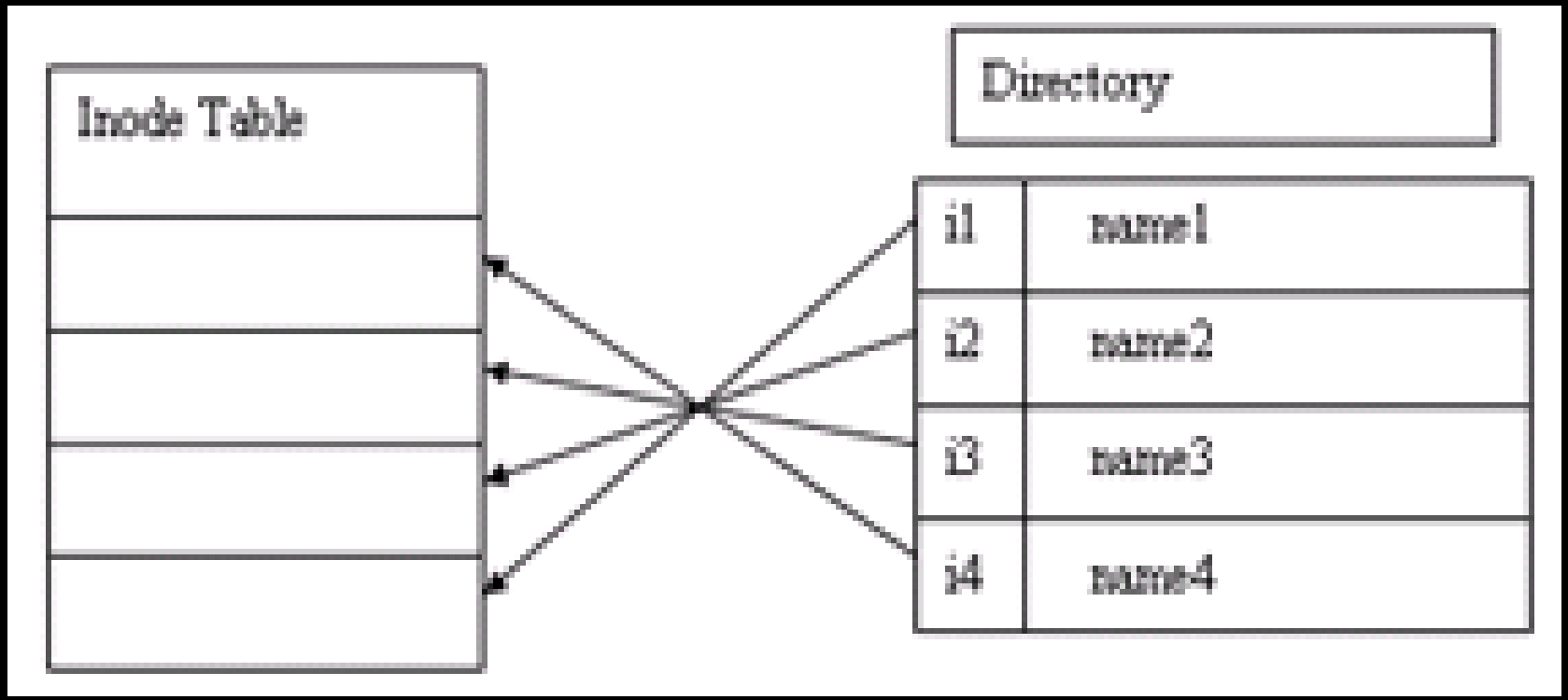
–s

ln

Những khái niệm ... (tt)



Những khái niệm ... (tt)



6. Những filesystem có sẵn trong Linux.

7. Sửa filesystem.

fsck <option> <partition>

fsck –V –a /

```
Parallelizing fsck version 1.23
e2fsck1.23, 15-August-2001 for ext2fs0.5b, 95/08/09
/dev/hda2 is mounted
Warning!!! Running e2fsck on a mounted filesystem
   may cause
SEVERE filesystem damage.
Do you want to continue(y/n)?
```

Sửa filesystem (tt).

-A /etc/fstab

-V

-t loai-fs

-a

-l

-r

-s

8. Mount filesystem.

mount –t <device_name> <mount_point>

-f
-v
-w
-r
-t loai-fs
-a
-o remount <fs>

Mount filesystem (tt)

umount <device_name> <mount_point>

umount -a

umount -t loai-fs

Lưu ý

Mount filesystem (tt)

cột 1

cột 2 none

cột 3

cột 4

cột 5 dump

cột 6 fsck

9. Di chuyển filesystem.


```
mkfs  –t  ext2  /dev/hda4

mkdir  /mnt/newpartition
mount  /dev/hda4  /mnt/newpartition
```

Di chuyển filesystem (tt)


```
cp  –a  /home/*  /mnt/newpartiton
```

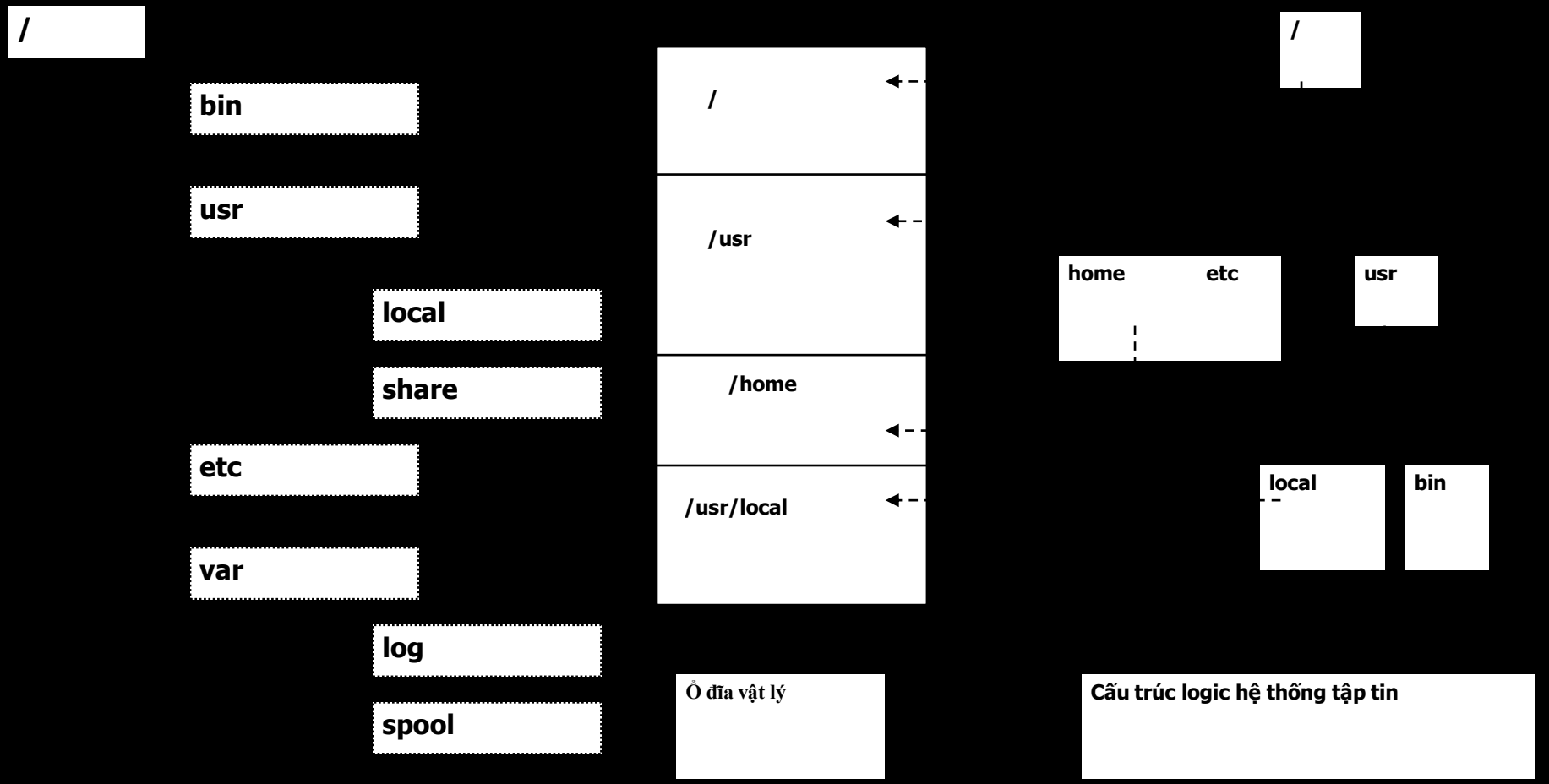
```
umount  /mnt/newpartition
mount  /dev/hda4  /home
```

10. Tập hợp thông tin về filesystem

df

df <option>

11. Cấu trúc cây thư mục.



12. Các thao tác trên tập tin và thư mục.

- pwd

  pwd
- cd

  cd [directory]
- ls

  ls [option] [directory]
- mkdir

  mkdir <directory>

Các thao tác ... (tt)

- rmdir

  rmdir <directory>

- cat

  cat <filename1> [filename2]

  cat

  > >> >

  >>

  CTRL-d

- more

  more <filename>

Các thao tác ... (tt)

- cp

  cp <source> <destination>

- mv

  mv <source> <destination>

- rm

  rm [option] <filename/directory>

- find

  find [path-list] [expression]

Các thao tác ... (tt)

-name <file>
-size n<bck>
-user uname uname

- grep

  grep [expression] [filename]

- touch

  touch <option> <filename>

- dd

  dd if=<file> of=<device>

13. Các tập tin chuẩn trong Linux.

- command < file
- command > file
- command1 | command2 | …

14. Lưu trữ tập tin và thư mục.

gzip/gunzip

gzip/gunzip [option] <filename>

tar

tar [option] <destination> <source>

CHƯƠNG 4

NHỮNG LỆNH VÀ TIỆN ÍCH

Nội dung

1. Trình soạn thảo vi.

vi <filename>

Chuyển chế độ lệnh sang chế độ soạn thảo:

Chuyển chế độ soạn thảo sang chế độ lệnh:

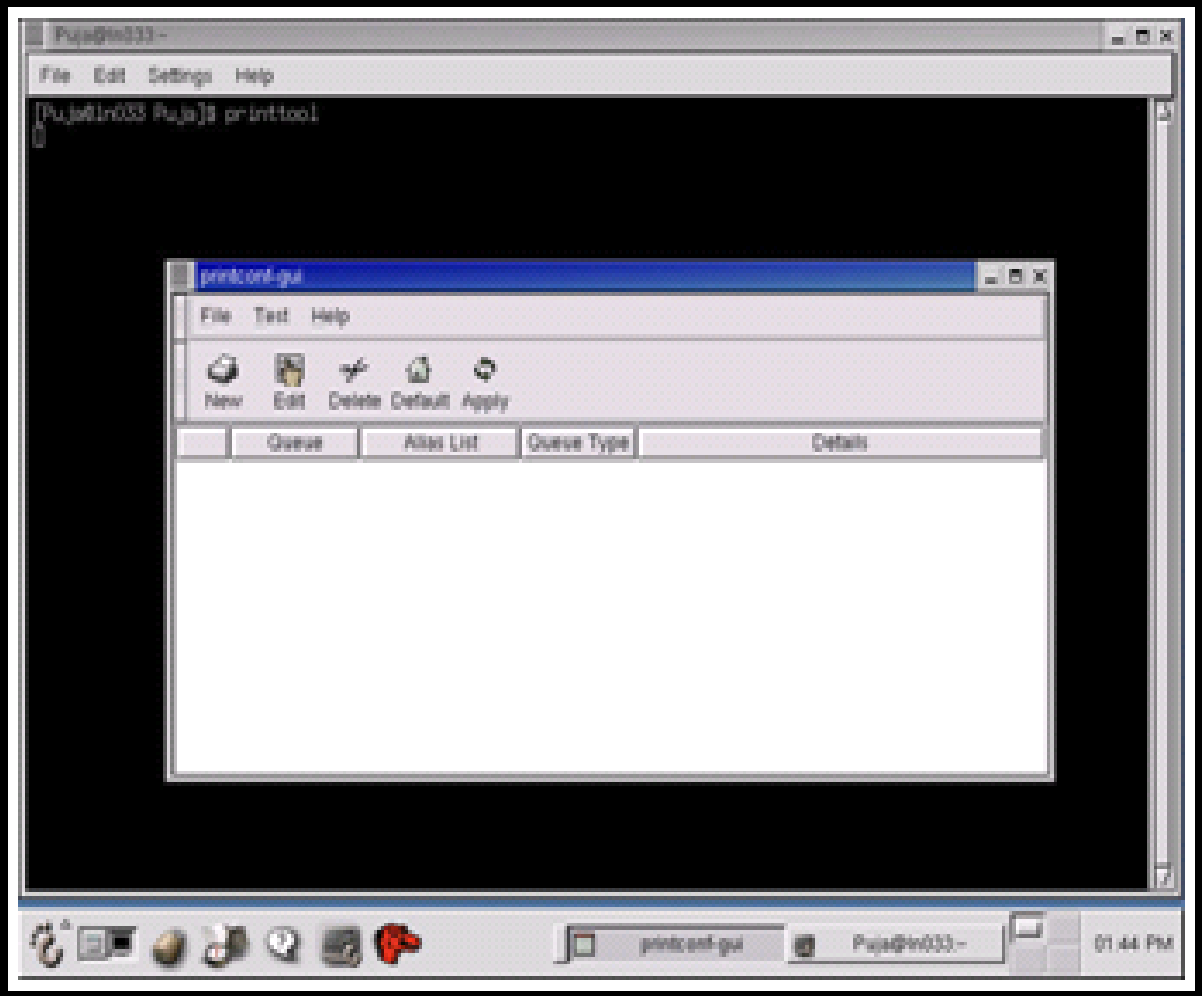
2. Email trong Linux.

- #mail
- #mail <user>

3. Dịch vụ in ấn.

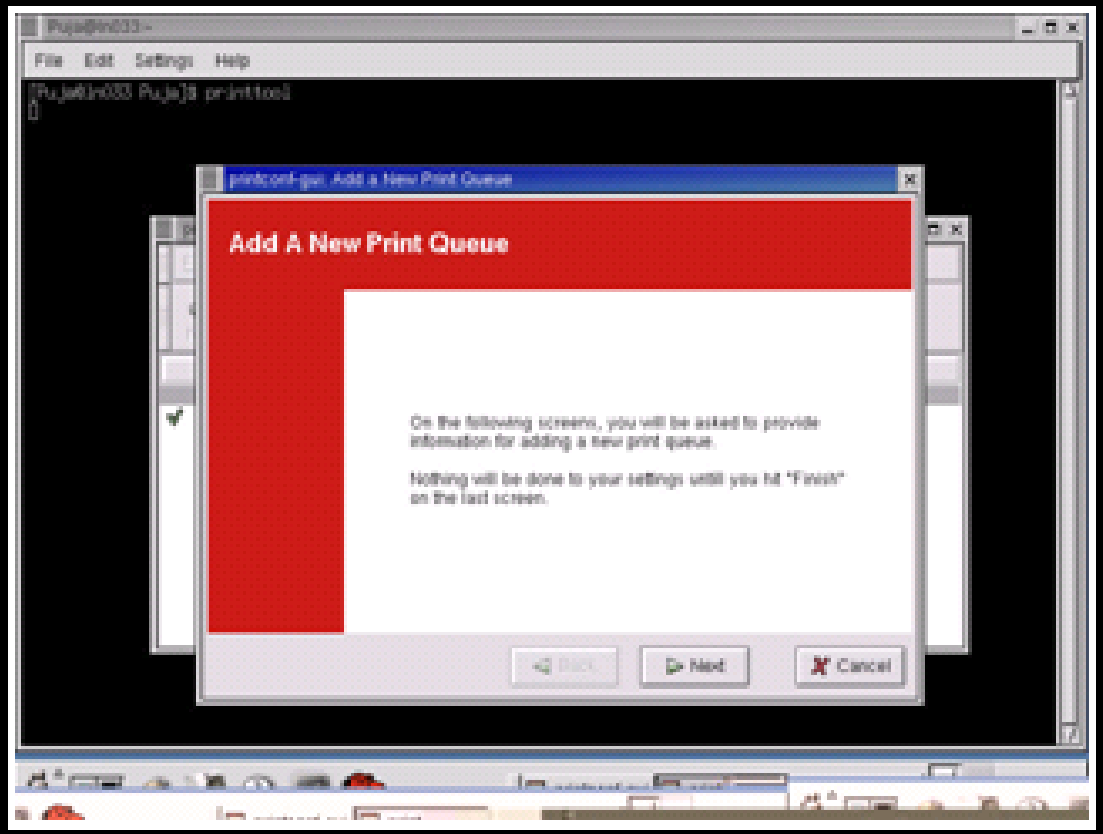
-
- printtool

Dịch vụ in ấn (tt)



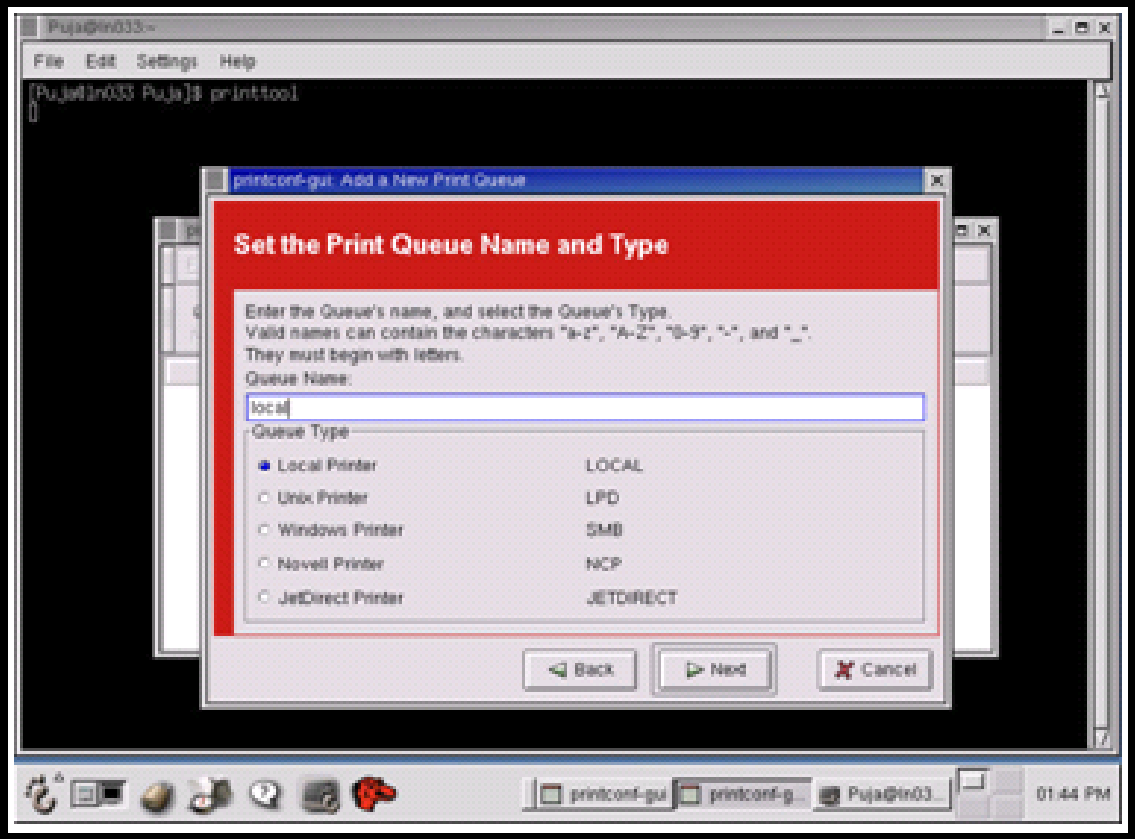
Dịch vụ in ấn (tt)

- New



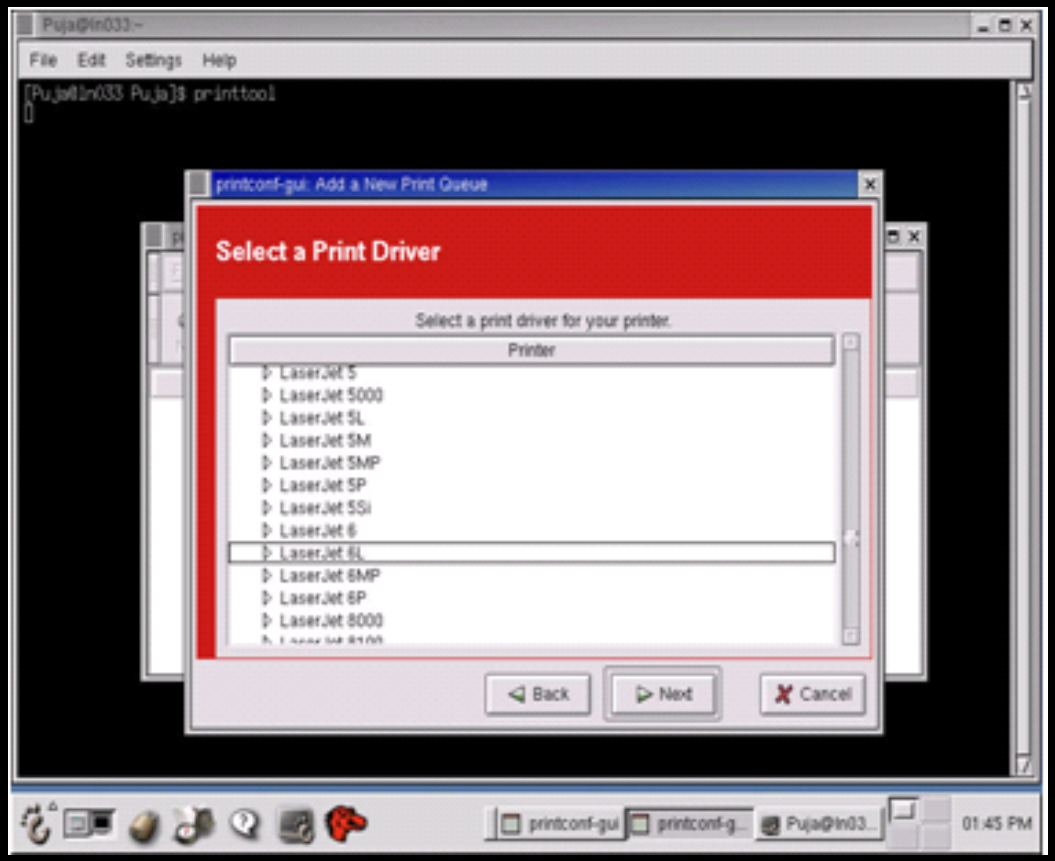
Dịch vụ in ấn (tt)

- Next



Dịch vụ in ấn (tt)

- Next



Dịch vụ in ấn (tt)

- Finish
- lpd

4. Những công cụ in ấn.

- Lpr
- Lpq
- Lprm
- Lpc

5. Một số các tiện ích khác.

setup

fdisk

iptraf

lynx

mc

CHƯƠNG 5

QUẢN LÝ USER, GROUP VÀ BẢO MẬT

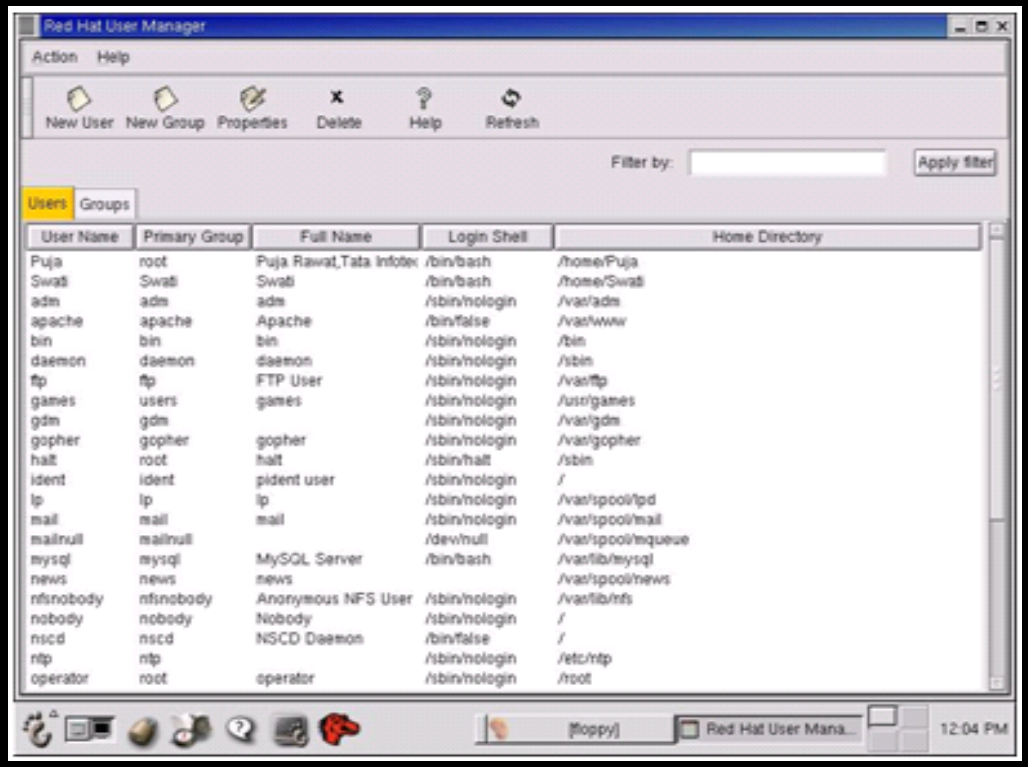
Nội dung

1. User.

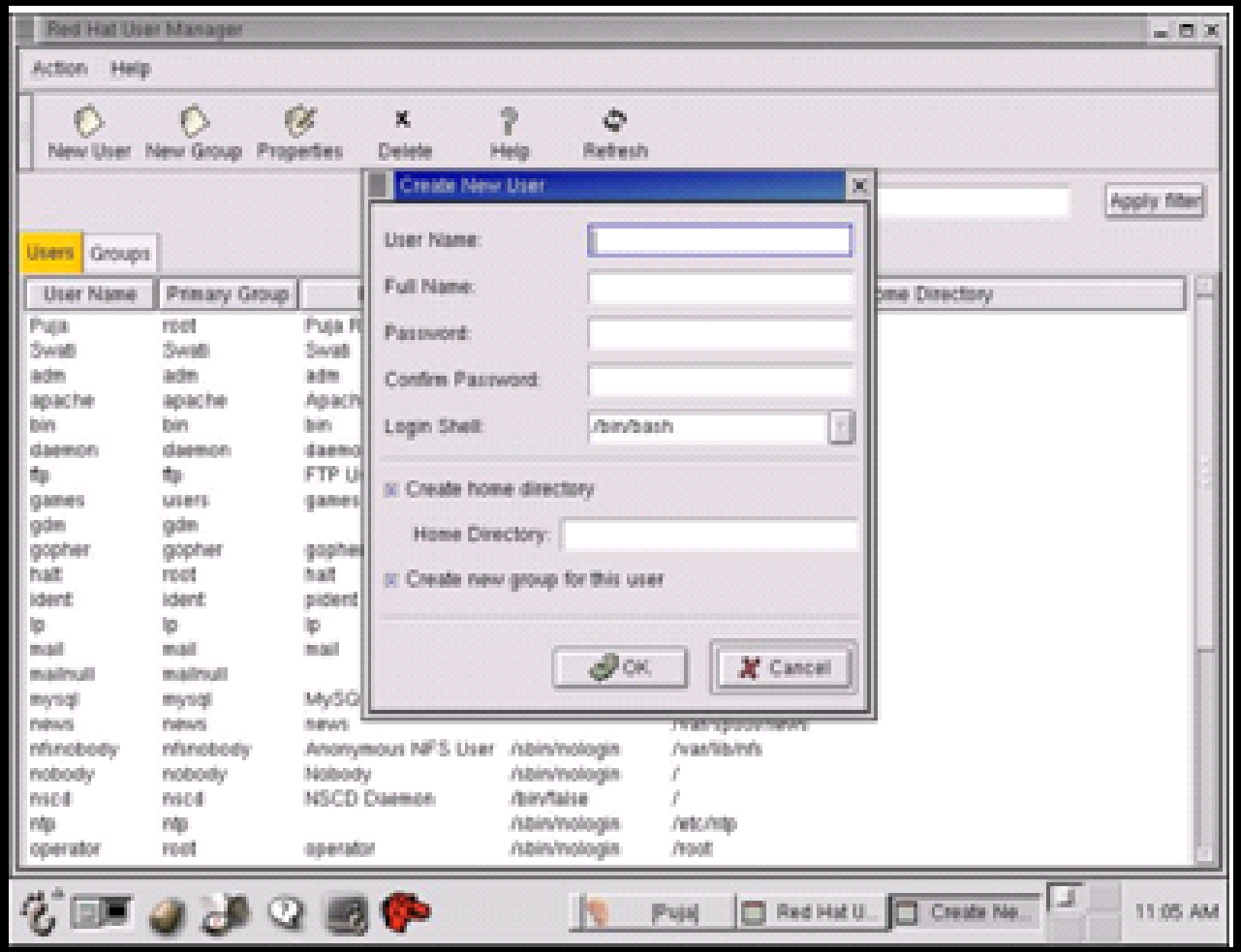
2. Group.

3. Các cách quản lý user và group.

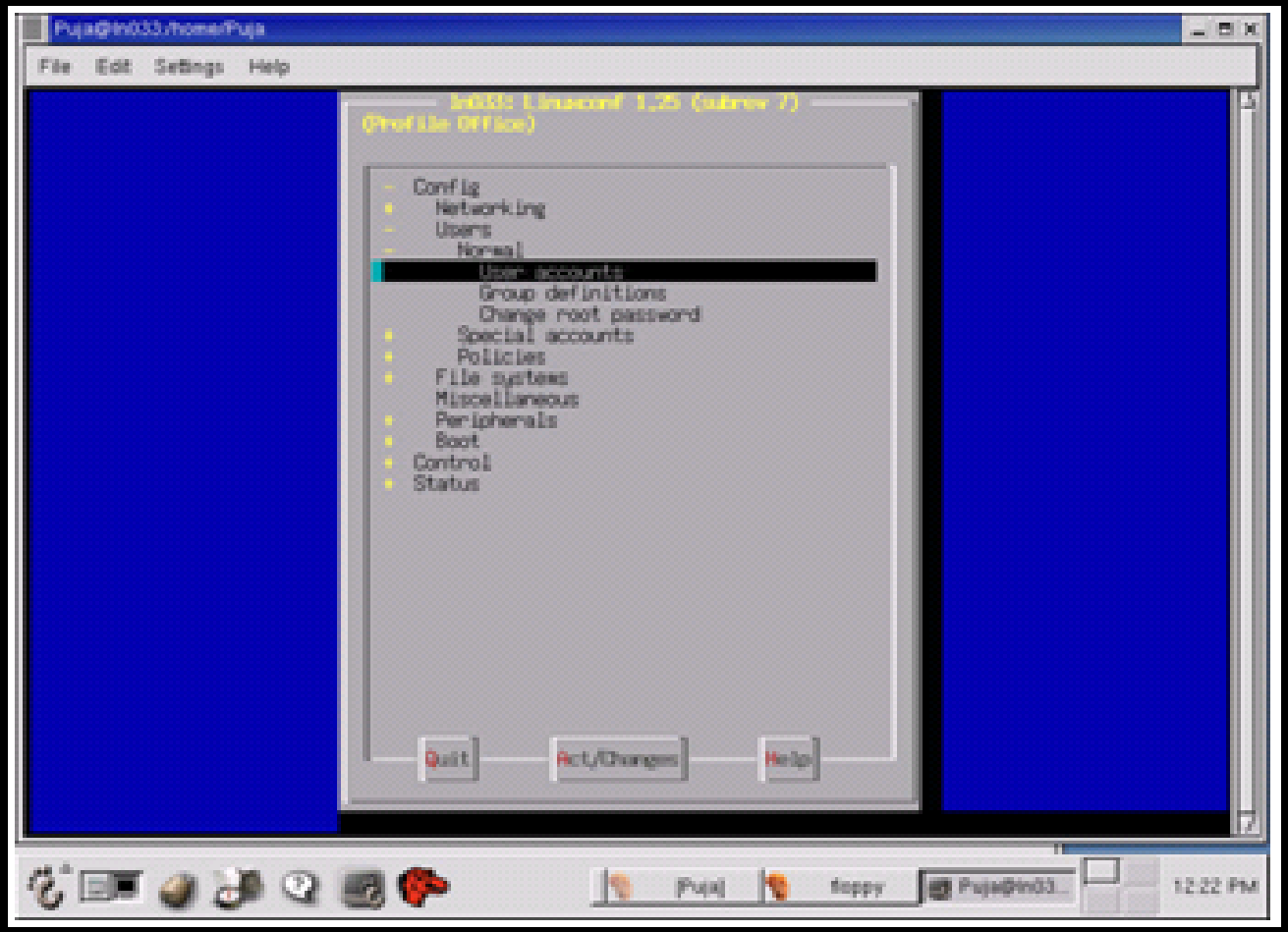
4. Tạo user bằng công cụ User Manager.



Tạo user bằng công cụ (tt)

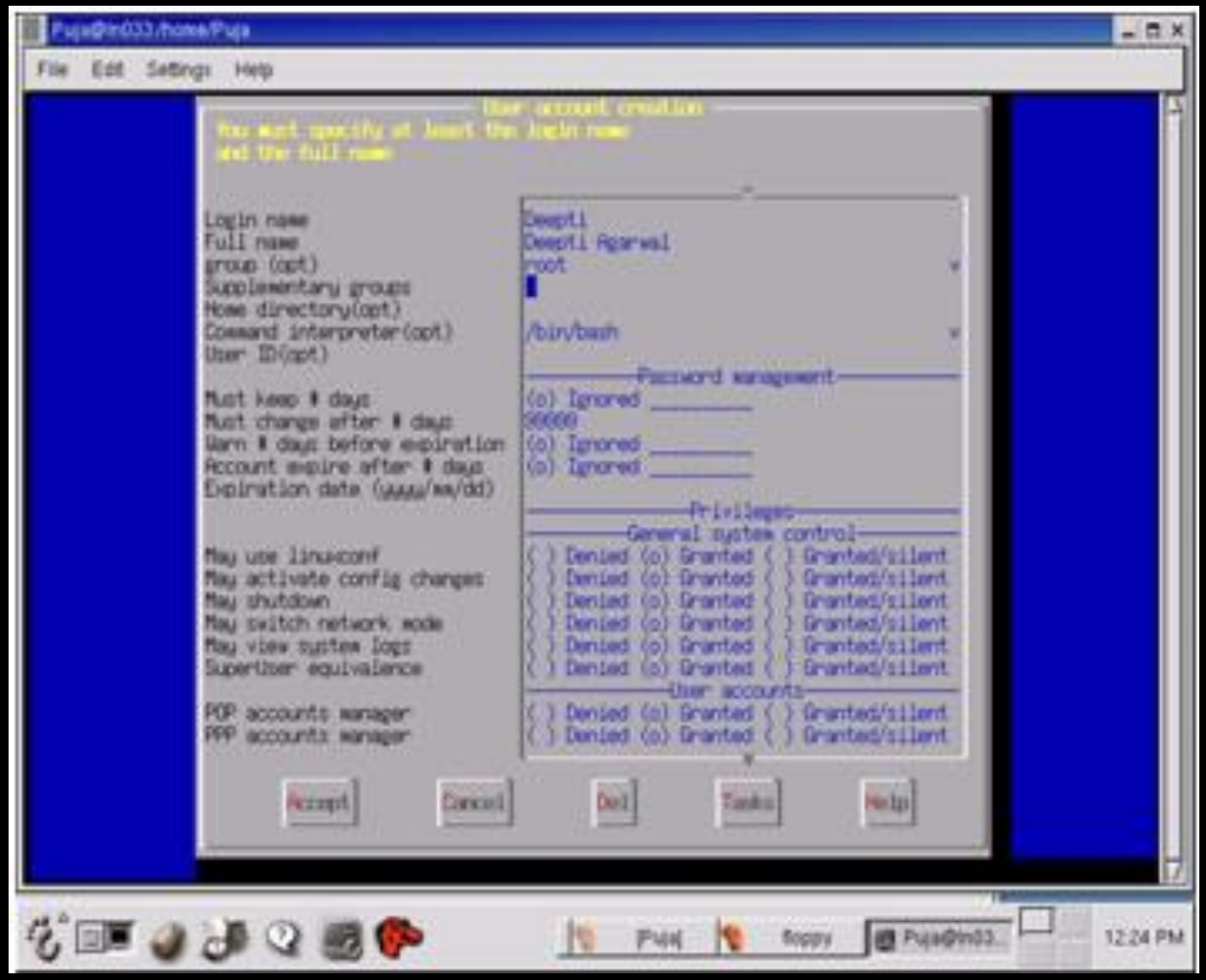


5. Tạo user với công cụ linuxconf.



Tạo user với công cụ linuxconf (tt)

Tạo user với công cụ linuxconf (tt)



6. Tập lệnh quản lý user và group.

useradd [options] <username>

-
-
-
-

Tập lệnh quản lý user và group (tt)

usermod [options] <username>

Tập lệnh quản lý user và group (tt)

userdel [option] <username>

passwd –l <username> passwd –u

usermod –L <username> usermod –U

x *

Tập lệnh quản lý user và group (tt)

groupadd <groupname>

groupdel <groupname>

Tập lệnh quản lý user và group (tt)

id <option> <username>

groups <username>

7. Những file lưu thông tin user và group.

/etc/passwd

/etc/group

/etc/shadow

8. Quyền hạn.

Quyền hạn (tt).

-rw-r--r-- 1 fido user 163 Dec 7 14 : 31 myfile

r w - r - - r - -

Tổ hợp 3 quyền trên có giá trị từ 0 đến 7.

9. Các lệnh liên quan đến quyền hạn.

chmod <specification> <file>

Các lệnh liên quan đến quyền hạn (tt).

chown <owner> <filename>

chgrp <group> <filename>

Các lệnh liên quan đến quyền hạn (tt).