HỆ ĐIỀU HÀNH UNIX

I/ Tổng quan về hệ điều hành UNIX

1. Lịch sử

Năm 1969 Ken Thompson đã lần đầu tiên cài đặt hệ điều hành UNIX. Mục tiêu khởi đầu là cung cấp một môi trường máy tính hoá để mô phỏng trò chơi không gian. Năm 1973 Ritchie và Thompson đã viết lại hệ điều hành bằng ngôn ngữ C, khác hẳn với các hệ điều hành truyền thống ghi bằng ngôn ngữ máy, do đó UNIX rất dễ cài đặt trên các hệ máy khác. Từ đó đã khai sinh ra hệ điều hành UNIX. Năm 1974 hệ thống UNIX đã được cài đặt trên các máy DEC PDP-11 ở hơn 100 trường đại học. Mục tiêu chủ yếu là cung cấp môi trường cho các lập trình viên chuyên nghiệp.

Ngày nay, 20 năm đã qua, có hàng trăm ngàn hệ thống UNIX cài đặt trên khắp thế giới. Hầu hết các hãng sản xuất máy đều có một phiên bản cho UNIX.

Tuy nhiên hiện nay để chuẩn hoá hệ điều hành UNIX, người ta quy ước các tập lệnh chuẩn và gọi là UNIX System V Release 4. Trên máy PC hiện nay phổ biến hai hệ điều hành là SCO UNIX và SUN Solaris.

2. Các đặc điểm cơ bản

Hệ điều hành UNIX có một số đặc điểm sau:

— Đa chương

— Nhiều người sử dụng

— Bảo mật

— Độc lập phần cứng

— Kết nối mở

— Dùng chung thiết bị

— Tổ chức tập tin phân cấp

a/ Nhiều người sử dụng:

Nhiều người sử dụng có thể sử dụng máy tính có cài UNIX tại một thời điểm. Ví dụ:

UNIX Server:

— User A: dùng Oracle

— User B: chương trình biên dịch

— User C: gửi thư

Hệ điều hành UNIX quản lý những người sử dụng theo cấu trúc phân cấp, người sử dụng có thể giao tiếp với nhau theo các nhóm. Người sử dụng cao nhất (super user) có thể can thiệp đến các người sử dụng khác nếu cần.

b/ Đa chương

Tại một thời điểm một người sử dụng có thể thực hiện đồng thời nhiều tác vụ. Với hệ điều hành đơn chương như MS-DOS một lệnh thực hiện sẽ chiếm toàn bộ thời gian CPU xử lý, bạn chỉ có thể thực hiện lệnh kế khi lệnh trước đó đã được thực hiện xong. Còn trong hệ điều hành UNIX bạn có thể đặt lệnh chạy ở chế độ nền (background) đồng thời khi đó có thể thực hiện các lệnh kế.

c/ Tổ chức tập tin phân cấp

Các tập tin của UNIX được tổ chức theo dạng cây có chung thư mục gốc được biểu diễn bởi ký tự /.

Bên trong thư mục có thể là các thư mục con hay các tập tin . UNIX có 3 loại tập tin:

- Tập tin bình thường (ordinary file): là một tập tin chứa các dữ liệu ASCII hay nhị phân.

- Tập tin thư mục (directory file): chứa danh sách các phần tử (thư mục, tập tin, thiết bị) có thể truy xuất tới.

- Tập tin đặc biệt (special file): là các tập tin liên quan đến các thiết bị phần cứng và truyền thông.

Ví dụ:

- bàn phím là một tập tin nhập.

- màn hình là một tập tin xuất.

/

UMUNIX - bin - users - dev

user1

user2

(n ổ đĩa)

Đối với Unix toàn bộ hệ thống tập tin chỉ có một root. Có thể sử dụng lệnh mount để kết nối các ổ đĩa trong một hệ thống tập tin duy nhất.

Ổ 1 : / LIB
BIN
USR

Ổ 2: /- ETC
ORACLE

mount => /- LIB
BIN
USR- ETC

- ORACLE

d/ Độc lập phần cứng

Vì hệ điều hành UNIX 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.

e/ 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.

f/ Bảo mật

Unix cung cấp rất nhiều cơ chế bảo mật khác. Trong đó mỗi người sử dụng có một số quyền trên các tập tin nhất định và chỉ được phép chạy một số chương trình nhất định. Ngoài ra cơ chế mã hoá và giải mã cũng là một phần của hệ điều hành.

g/ Kết nối mở

Unix cung cấp các thiết bị mạng qua Ethernet, Modem, X25. Với nhiều thủ tục truyền thông khác nhau UUCP (Unix-to-Unix Copy), TCP/IP, và các ứng dụng E-mail,FTP, NFS (Network File System).

3. Các thành phần chính của hệ điều hành UNIX:

- ✓ Windows & Graphic User Interface
- ✓ Shell
- ✓ Lệnh và tiện ích
- ✓ Các bộ điều khiển thiết bị
- ✓ Kernel

a/ Kernel:

Là thành phần chủ yếu hay trái tim của hệ điều hành. Nó chiếm khoảng 500KB --> 2MB tùy theo tính chất phức tạp của hệ thống. Nhiệm vụ chủ yếu của Kernel là:

⇨ Quản lý tài nguyên: như bộ nhớ, v.v...

⇨ Quản lý hệ thống tập tin: có thể là các tập tin, thư mục cục bộ hay từ xa.

⇨ Quản lý các quá trình thường trú (deamon).

⇨ Quản lý bộ nhớ ảo: để thực thi nhiều quá trình đồng thời trong khí số lượng bộ nhớ có hạn, Unix tổ chức một vùng trên đĩa như là một vùng bộ nhớ (bộ nhớ ảo). Kernel phải "swap" các quá trình giữa bộ nhớ --> bộ nhớ ảo

RAM

Kernel Vùng làm việc Swap

Đĩa cứng

Swap

bộ nhớ ảo

⇨ Quản lý quá trình :

Như đã biết vì Unix là một hệ điều hành đa chương do đó việc quản lý các quá trình đồng thời rất phức tạp. Nó phải quản lý việc khởi tạo và kết thúc các quá trình cũng như các tranh chấp có thể xảy ra.

⇨ Quản lý các bộ điều khiển thiết bị.

⇨ Quản lý mạng: bao gồm nhiều thiết bị phần cứng khác và các thủ tục khác.

⇨ Quản lý việc khởi động và dừng máy.

b/ Bộ điều khiển thiết bị:

UNIX thể hiện các thiết bị vật lý như các tập tin đặc biệt. Mộ tập tin đặc biệt sẽ có 1 điểm vào trong thư mục và có 1 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.

c/ Lệnh và tiện ích:

Các lệnh và tiện ích của Unix rất đa dạng.

Một lệnh UNIX 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:

a/ 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

b/ Trình báo màn hình:

echo	hiển thị dòng ký tự hay biến
setcolor	đặt màu nền và chữ của màn hình

c/ 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

d/ 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

e/ 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

f/ Quản lý quá trình:

kill	hủy bỏ một quá trình
ps	trình bày tình trạng của các quá trình
sleep	ngưng hoạt động một thời gian

g/ 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

h/ Kiểm soát in:

cancel	ngưng in
lp	in tài liệu ra máy in
lpstat	trạng thái của hàng chờ in

d/ Shell:

Là bộ xử lý lệnh của người sử dụng, nó cho phép người sử dụng tạo các lệnh rất phức tạp từ các lệnh đơn giản. Chúng ta có thể coi shell như một ngôn ngữ lập trình cấp cao. Các chức năng chính của shell là:

UNIX Shell:

- ✓ Kiểm soát I/O và đổi hướng
- ✓ Các biến môi trường
- ✓ Thực hiện lệnh
- ✓ Thư viện lệnh nội tại
- ✓ Tên tập tin mở rộng
- ✓ Ngôn ngữ lập trình và môi trường

Thực hiện lệnh: Lệnh có thể được thực hiện ở chế độ tương tác với người sử dụng hay chế độ nền (background).

Thư viện lệnh nội tại: Các lệnh nội trú trong shell.

Ngôn ngữ lập trình và môi trường: Cho phép tạo các tập tin shell-script và các cấu trúc điều khiển như Do, While, Until, If, Case.

Tên tập tin mở rộng: Cho phép biên dịch tên tập tin ở dạng ?, *.

Các biến môi trường: Cho phép đặt các biến môi trường. Ví dụ: PATH=/USR/BIN

Kiểm soát I/O và đổi hướng: Shell định nghĩa các thiết bị xuất/ nhập chuẩn và cho phép ta đổ hướng thiết bị xuất/ nhập của các quá trình.

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 : 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.

e/ Windows và Graphic User Interface:

Giao tiếp đồ hoạ và cửa sổ là một khả năng rất mạnh của hệ điều hành UNIX, nó cho phép hệ điều hành giao tiếp thân thiện hơn với người sử dụng. Hiện nay UNIX cài đặt XWINDOW (X11) là một môi trường quản lý đồ hoạ lý tưởng. Trong Sun Solaris thì sử dụng với tên gọi là OpenWin.

4. Tóm lại:

Đứng về phía người sử dụng ta có thể hình dung hệ điều hành UNIX như sau:

Người sử dụng - lệnh Unix - biên dịch Shell - Kernel - Máy tính.

5. UNIX và MS-DOS

Phần này nhằm khám phá sự tương tự và khác biệt giữa UNIX và MS-DOS để giúp cho những người đã làm quen với MS-DOS có thể dễ dàng học UNIX.

5.1/ Các đặc điểm chung cung cấp bởi UNIX và MS-DOS

Bộ biên dịch lệnh

UNIX và MS-DOS cả hai đều có bộ biên dịch lệnh tách biệt với phần nhân của hệ điều hành. Ưu điểm chính của việc tách biệt với nhân của hệ điều hành là sự độc lập này cho phép bạn có thể chọn các loại shell khác nhau để sử dụng. Cả hai môi trường DOS và UNIX đều có một số bộ biên dịch lệnh khác nhau.

Trong UNIX shell chuẩn được là sh, trong MS-DOS nó được gọi là Command.com. Trong cả hai hệ thống bạn có thể thực hiện một bản sao mới của shell bằng cách đánh tên của nó như lệnh. Một shell mới sẽ được chạy như một quá trình của shell hiện hành.

Lập trình Shell

Trong MS-DOS có các tập tin bó cho phép bạn tổ hợp các lệnh thường sử dụng vào 1 lệnh mới. Đây là một dạng lập trình. Sức mạnh lập trình của shell trong UNIX lớn hơn nhiều so với đặc điểm tập tin bó của MS-DOS. Một chương trình viết bởi shell UNIX gọi là shell-script. Một vài lệnh cho phép trong tập tin bó của MS-DOS thì lại không cho phép đưa vào từ bàn phím. Ngược lại, UNIX không có sự phân biệt giữa lệnh trong tập tin shell-script và lệnh đưa vào từ bàn phím.

Các biến môi trường

Giống như MS-DOS, shell UNIX cung cấp các biến môi trường ở dạng TÊN=GIÁTRỊ. Các dòng ký tự này có thể sử dụng để chứa các tham số và chọn lựa để cho chương trình sử dụng. Một vài biến môi trường được sử dụng bởi tự bản thân shell như PATH. Ví dụ:

```
SET PATH = %PATH%;D:\NEW\DIR (DOS)
path=$path,:/new/dir  (UNIX)
```

Các cấu trúc kiểm soát chương trình

Các tập tin bó MS-DOS sử dụng cấu trúc điều khiển chương trình chỉ trong một dòng. Các cấu trúc điều khiển trong UNIX là không giới hạn, bạn có thể ghi một tập tin shell giống như một chương trình. Ví dụ: viết tập tin bó để sắp thứ tự và in 1 tập hợp các tập tin.

Trong DOS:

```
FOR %F IN (*.WK) DO SPRT %F
```

Với SPRT.BAT có nội dung:

```
SORT %1 >\TMP\%1
PRINT \TMP\%1
```

Trong UNIX:

```
for F in *.wk
do
sort $F >/tmp/$F
lp /tmp/$F
done
```

Xuất nhập chuẩn và đổi hướng

Cả hai hệ thống đều chứa các bảng trong bộ nhớ để quản lý các tập tin mở. Trên DOS là file handle, dưới UNIX là file description. Các chương trình đọc/ ghi tập tin sử dụng file handle hay file description để báo với hệ điều hành các tập tin sử dụng. Tất cả các chương trình khi chạy đều có 3 tập tin đã mở sẵn : nhập chuẩn (0), xuất chuẩn (1), và tập tin sai (2).

Cả 2 hệ điều hành UNIX và DOS đều cung cấp khả năng đổi hướng. Có nghĩa là khả năng thay đổi các thiết bị xuất nhập chuẩn. Ưu điểm của đổi hướng là chương trình trở thành độc lập thiết bị.

Cả DOS và UNIX đều cung cấp cơ chế *đường ống* (xuất của 1 chương trình trở thành nhập của chương trình khác). Tuy nhiên vì MS-DOS không phải là một hệ thống đa chương nên không thể thực hiện 2 chương trình cùng một lúc, nó chứa toàn bộ dữ liệu xuất của chương trình đầu vào một tập tin tạm mà sau đó sẽ được đọc bởi chương trình thứ hai. UNIX ngược lại, nó có khả năng thực hiện 2 chương trình đồng thời, do đó dữ liệu xuất của chương trình đầu có thể trực tiếp đưa vào chương trình thứ hai không qua tập tin tạm.

Ngữ pháp của đổi hướng là giống nhau ở cả hai hệ thống: < đổi hướng nhập, > đổi hướng xuất. Cả UNIX và DOS cùng một ngữ pháp cho đường ống (ký hiệu |).

Ví dụ:

Trong DOS:

```
DIR | SORT > PRN
```

Trong UNIX:

```
ls | sort > /dev/lp
```

Tác động nạp hệ thống

Khi nạp MS-DOS, nó lấy tham số cấu hình từ một tập tin gọi là CONFIG.SYS và sau đó thực hiện các lệnh trong AUTOEXEC.BAT. Khi nạp UNIX nó lấy tham số cấu hình trong tập tin /etc/inittab (xác định các chương trình sẽ chạy ở các chế độ khác nhau hoạt động đơn hay đa chương. Chức năng cơ bản của /etc/inittab là định nghĩa các cổng có terminal nối vào và khởi động các terminal này), sau đó thực hiện các lệnh trong tập tin /etc/rc (đây là một tập tin shell script nhằm mục đích khởi động hệ thống tự động). Ngoài ra trong UNIX để khởi động môi trường cho mỗi người sử dụng nó còn nạp vào tập tin /etc/.profile

Cấu trúc tập tin

UNIX và DOS cả hai đều có hệ thống tập tin phân cấp. Một sự khác biệt quan trọng giữa hai hệ thống là thư mục của UNIX chỉ chứa tên tập tin và con trỏ đến dữ liệu của tập tin. Do đó UNIX cho phép một tập tin có thể có bí danh. Do đó nếu ở DOS bạn phải giữ 2 -> 3 bản sao thì ở UNIX chỉ giữ các con trỏ đến tập tin. Một sự khác biệt khác giữa hai hệ thống là đường dẫn của DOS chỉ hướng đến thư mục, tập tin trên đĩa, trong khi đó đường dẫn của UNIX có thể hướng đến thư mục, tập tin, thiết bị hay đường ống.

MS-DOS phân biệt các ổ đĩa - thực sự hay luận lý - mỗi ổ đĩa có một thư mục gốc. Trong khi đó UNIX chỉ cho phép 1 thư mục gốc cho toàn bộ hệ thống. Để nối kết các thư mục trên các ổ đĩa khác nhau, UNIX sử dụng ***mount***. Ví dụ:

```
Ổ 1                    Ổ 2
/--bin        mount    /--bin              /--bin
---lib        ---->    ---lib       ==>    ---lib
---usr                 ---spool            ---usr    ---bin
                                                     ---lib
                                                     ---spool
```

Với cách thức này thậm chí ta có thể mount cả các thư mục ở máy từ xa.

Các bộ điều khiển có khả năng nạp và nối kết

Cả hai UNIX và DOS cung cấp khả năng thêm các thiết bị mới bằng cách cung cấp các bộ điều khiển thiết bị điều khiển các yêu cầu xuất/ nhập đến thiết bị. UNIX định nghĩa tất cả các bộ điều khiển thiết bị trong thư mục /dev. UNIX hướng đến các thiết bị như 1 tập tin đặc biệt.

Các thuộc tính tập tin

UNIX có tất cả các thuộc tính tập tin như DOS. Thuộc tính tập tin trong UNIX gọi là ***file mode*** mà có thể thay đổi bởi lệnh chmod. Chế độ của tập tin bao gồm những bit cho phép chia làm 3 nhóm, mỗi nhóm 3 bit mà cung cấp các chủ quyền đọc, ghi, thực hiện cho 3 lớp người sử dụng: sở hữu, nhóm, chung.

TSR và Daemon

DOS cung cấp loại quá trình nền gọi là các chương trình thường trú. Các chương trình thường trú không phải là một quá trình độc lập như các quá trình trong UNIX. Trong UNIX bạn có thể thực hiện đồng thời hai hay nhiều quá trình. Để thực hiện một quá trình nền bạn có thể chạy một lệnh theo sau bởi ký hiệu &. Có nghĩa là sau khi lệnh bắt đầu chạy, shell sẽ trở lại ngay dấu nhắc, sau đó bạn có thể thực hiện lệnh kế, khi đó cả hai lệnh nền và lệnh mới thực hiện đồng thời. Nếu lệnh nền tách rời khỏi terminal của bạn thì lệnh nền trở thành deamon.

5.2/ Các đặc điểm của UNIX mà DOS không có

Hoạt động nhiều người sử dụng

UNIX là một hệ điều hành đa chương. Nó cung cấp sự phân biệt giữa người sử dụng này với người sử dụng khác. Mỗi người sử dụng được định danh bởi tên và chỉ số. Hệ thống sẽ yêu cầu bạn đưa tên mỗi khi bạn thâm nhập vào hệ thống. Tất cả các quá trình mà bạn khởi động, tất cả các tập tin mà bạn tạo là được đánh dấu với chỉ số của bạn. UNIX cung cấp một hệ thống chủ quyền tập tin để bảo vệ các tập tin được tạo bởi người sử dụng với các mức truy xuất khác nhau. UNIX cung cấp mỗi người sử dụng 1 tập hợp các thư mục, như vậy bạn có thể tạo và quản lý các tập tin mà không ảnh hưởng đến những người sử dụng khác. Các đặc điểm này không có trong DOS bởi vì DOS là hệ điều hành hướng đến 1 người sử dụng.

Đa chương

UNIX là một hệ điều hành đa chương. Bất cứ người sử dụng nào cũng có thể chạy nhiều chương trình đồng thời bởi sử dụng đặc điểm lệnh *nền*. Một chương trình có thể tạo một quá trình mới điều khiển, thông tin và đợi để chúng hoàn thành trước khi tiếp tục.

Đa chương được cài đặt bên trong nhân của UNIX. Tất cả các quá trình được quản lý bởi bộ định thời biểu quá trình theo độ ưu tiên. UNIX cung cấp một vài dạng thông tin giữa các quá trình. Nó cho phép tạo, kiểm soát và hủy bỏ quá trình.

UNIX cài đặt thủ tục "swap" mà cho phép các quá trình có độ ưu tiên thấp có thể loại bỏ tạm khỏi bộ nhớ sau đó được đưa trở lại sau khi các quá trình có độ ưu tiên cao hơn đã được thực hiện. Một vùng trên đĩa được dành riêng để "swap". Các quá trình trong UNIX được quản lý theo cơ chế phân cấp. Quá trình gốc gọi là quá trình init, quá trình này chạy tự động khi khởi động hệ điều hành, và có trách nhiệm tạo tất cả các quá trình thêm vào. Quan hệ cha - con tồn tại giữa 1 quá trình với bất cứ quá trình nào tạo nó. Đường ống có thể sử dụng để quá trình liên lạc với các quá trình cha hay con của nó. Khi các quá trình kết thúc, nó sẽ thông báo cho quá trình cha, một quá trình con thừa kế các biến môi trường và các tập tin mở của quá trình cha. Các quá trình UNIX có thể chia xẻ mã, khi nhiều người sử dụng chạy cùng một chương trình ở cùng một thời gian, chỉ một bản sao của chương trình là được đưa vào bộ nhớ.

Các đặc điểm này không được cung cấp bởi DOS. Bởi vì DOS là 1 hệ điều hành đơn chương.

Bảo vệ

UNIX cung cấp cơ cấu bảo vệ trong một vài lĩnh vực. Chủ quyền tập tin, kiểm soát sự truy xuất tập tin, thư mục, thiết bị. Sơ đồ mật khẩu ngăn ngừa những người không chủ quyền thâm nhập hệ thống. Ngoài ra nó cho phép mã hoá và giải mã các tập tin hay các khối dữ liệu.

Các nối kết

UNIX cung cấp đặc điểm gọi là nối kết, có nghĩa là 1 tập tin có thể được thể hiện bởi một vài điểm vào trong thư mục. Đặc điểm này không được cung cấp trong DOS.

Độc lập thiết bị

DOS chỉ cung cấp cho họ Intel, UNIX có thể cài đặt cho rất nhiều hệ phần cứng khác nhau từ máy tính lớn đến các máy tính cá nhân, bởi vì UNIX được viết bằng ngôn ngữ cấp cao (C).

5.3/ Các lệnh tương quan giữa DOS và UNIX

lệnh DOS	*tác vụ*	*lệnh UNIX*	*lưu ý*
cd	đổi thư mục	cd	
md	tạo thư mục	mkdir	
rd	xoá thư mục	rmdir	
copy	sao chép tập tin	cp copy tar	sao chép tập tin sao chép thư mục sao chép tập tin hay thư mục lên đĩa mềm
ren	đổi tên tập tin	mv	
type	liệt kê nội dung tập tin	cat more	
xcopy	sao chép thư mục	copy	
del	xoá tập tin	rm	
dir	liệt kê danh sách tập tin và thư mục	ls	
edit	soạn thảo văn bản	vi	
find	tìm một đoạn văn trong tập tin	grep	
print	in tập tin	lp, copy	lp tên tập tin &

Chú ý: khi đưa vào lệnh UNIX phân biệt giữa chữ lớn và chữ nhỏ.

II/ Thâm nhập hệ thống:

Bắt đầu và kết thúc phiên làm việc:

Khi bắt đầu làm việc trên hệ thống bạn phải login. Việc login báo cho hệ thống biết bạn là ai và các chủ quyền làm việc của bạn, khi kết thúc bạn phải logout. Khi đó không có một ai khác có thể truy xuất tập tin của bạn nếu không được phép. Trong một hệ thống có nhiều người sử dụng, mỗi người có một tên và một mật khẩu duy nhất.

login

Khi hệ thống đã sẵn sàng hoạt động ta sẽ thấy trên màn hình hiện lên:

login:

Bạn phải đưa vào tên của bạn và nhấn Enter. Kế đó là:

password:

Bạn hãy đưa vào mật khẩu của bạn. Mật khẩu phải dài ít nhất 6 ký tự và tối đa 13 ký tự, có thể thay đổi bởi lệnh passwd. Sau đó sẽ hiện lên dấu $ lúc này bạn có thể thực hiện lệnh được.

logout

Có thể kết thúc phiên làm việc bởi lệnh

exit (C-Shell)

logout (Bourne-Shell)

III/ Hệ thống tập tin

1. Khái niệm cơ bản

UNIX có hai dạng tổ chức tập tin. Thứ nhất là dạng vật lý (tập tin được tìm thấy trên đĩa cứng). Thứ hai là luận lý (tập tin thuộc sở hữu của ai). Với tổ chức vật lý, UNIX sử dụng một tập hợp các bảng gọi là "i-nodes" (các nút thông tin). Với tổ chức luận lý nó sử dụng hệ thống thư mục cây phân cấp (trong UNIX một thư mục là một tập tin chứa danh sách của tất cả các tập tin và thư mục con của thư mục đó). Các phần tử của thư mục có dạng:

idnode	tên tập tin

Mỗi idnode chứa các thông tin liên quan đến một tập tin bao gồm:

i-node	loại	chủ quyền	nối kết	UID	GID	Thời gian	Kích thước	Địa chỉ khối dữ liệu đầu

Loại : chỉ thị loại tập tin có thể là thư mục, tập tin, tập tin đặc biệt.

Chủ quyền : bao gồm 3 nhóm, mỗi nhóm 3 bit để kiểm soát quyền truy xuất và thực thi của người sở hữu, nhóm, chung.

Nối kết : trong UNIX, một tập tin vật lý (inode) có thể được nối kết với một hay nhiều tập tin luận lý (tên tập tin). Nối kết chỉ ra số tập tin luận lý nối kết với tập tin vật lý.

UID (user identification) : định danh của người sở hữu

GID (group identification) : định danh của nhóm sở hữu

UNIX tổ chức hệ thống tập tin bao gồm chỉ một thư mục gốc (/) mà từ đó các thư mục con của nó được gắn vào một cách trực tiếp hay gián tiếp. Có một vài thư mục con chuẩn /bin, /usr, /etc, v.v... Mỗi thư mục này lại chứa các tập tin hay thư mục con.

Bạn có thể sử dụng đường dẫn đầy đủ để xác định một tập tin, ví dụ: /usr/NVA/chuong1. Bạn cũng có thể sử dụng chỉ tên tập tin nếu tập tin được chứa trong thư mục hiện hành. Thường khi bạn login, thư mục hiện hành sẽ được đặt đến thư mục "home". Đây là thư mục được thiết lập bởi người quản trị hệ thống dành cho bạn, nó thường là trùng với tên login của bạn.

Những người sử dụng khi truy xuất các tập tin trong UNIX phải bị kiểm soát bởi "chủ quyền" (vùng chủ quyền trong inode). UNIX đoán nhận 3 lớp chủ quyền, chủ quyền đọc (r), chủ quyền ghi (w), chủ quyền thực hiện (x). Nếu tập tin là thư mục thì các chủ quyền trở thành chủ quyền trình bày nội dung của thư mục (r), thêm hay loại bỏ tập tin khỏi thư mục (w), và tìm tập tin hay thư mục con trong thư mục (x).

UNIX cũng đoán nhận 3 lớp người sử dụng dựa trên định danh (ID) liên hệ với mỗi tập tin. Mỗi tập tin có định danh người sử dụng và định danh nhóm người sử dụng. Mặc định, người tạo tập tin là người sở hữu và nhóm sở hữu là nhóm của người tạo tập tin. Dĩ nhiên ta có thể thay đổi bởi lệnh chown và chgrp.

Ba chủ quyền đối với tập tin và ba chủ quyền đối với người sử dụng tạo nên 9 chủ quyền đối với một tập tin. Sử dụng lệnh ls -l sẽ hiển thị các chủ quyền này cho từng tập tin. Ví dụ:

```
-rwxr-xr--
drwxr-xr--
```

Ký tự đầu xác định loại tập tin (d: thư mục) 3 ký tự kế xác định chủ quyền của người sở hữu, 3 ký tự kế xác định chủ quyền của nhóm, 3 ký tự cuối là chủ quyền của một người sử dụng bất kỳ. Dấu - chỉ thị là không có chủ quyền.

Bạn cũng có thể sử dụng số bát phân để thể hiện chủ quyền.

✓ **Số**	**Ý nghĩa**
✓ 04	chủ quyền đọc
✓ 02	chủ quyền ghi
✓ 01	chủ quyền thực hiện

Ví dụ: cho phép đọc ghi sẽ có số bát phân là 06

UNIX sử dụng lệnh ***mount*** để nhận hệ thống tập tin. "Mount" là quá trình thâm nhập một hệ thống tập tin hay một thư mục phân cấp của một hệ thống tập tin từ đĩa hay băng. Đặc biệt đối với hệ điều hành Sun Solaris bạn còn có thể truy xuất hệ thống tập tin hay thư mục của máy từ xa thông qua môi trường mạng.

Khi máy tính khởi động nó tự động chạy chương trình mount và gài những tập tin cần thiết để hoạt động hoặc cục bộ hoặc từ xa. Bạn có thể sử dụng lệnh mount để truy xuất thêm các tập tin hay thư

mục. Khi máy tính cài một thư mục phân cấp nó cài thư mục vào một điểm mount. Điểm mount được cấp phát ở hệ thống tập tin cục bộ. Thường điểm mount là một thư mục rỗng. Ví dụ:

Hệ thống tập tin trên đĩa của bạn:

Điểm mount ==>

từ đĩa hay máy từ xa:

Sau khi mount hệ thống tập tin trở thành

2. Các lệnh thao tác trên thư mục

Sử dụng lệnh cd [thư mục] để thay đổi thư mục làm việc hiện hành.

Ví dụ:

- chuyển đến thư mục /usr/include

```
$cd /usr/include
```

- chuyển trở lại thư mục "home"

```
$cd
```

- chuyển đến một thư mục con của thư mục hiện hành

```
$cd ccs
```

- chuyển đến thư mục cha

```
$cd..
```

Sử dụng lệnh ls để trình bày nội dung của thư mục

```
ls [choïn löïa] [thö muïc]
```

Ví dụ:

- danh sách các tập tin trong thư mục hiện hành

```
$ls
passwd
hosts
```

- danh sách các tập tin sắp theo thứ tự trong các lệnh

```
$ls -c
```

```
hosts passwd
inittab
```

- danh sách tập tin với loại

```
$ls -CF
bin/ chmod*
etv/ temp
```

/ chỉ thị bin là thư mục, * chỉ thị chmod là tập tin thực thi.

- danh sách các tập tin ở thư mục khác

```
$ls /bin
```

- danh sách một số tập tin trong thư mục

```
$ls -x [pP]*
prigfoot
Pasks
```

- danh sách tập tin với kích thước

```
$ls -s
total 28
4 passwd
1 hosts
```

- danh sách tập tin với các thông tin chi tiết

```
total 28
drwxr-xr-x 2 NVA adm 98 Sep 13 10:04 NVAS
```

chủ quyền nối kết tên người sở hữu tên nhóm sở hữu kích thước thời gian tên tập tin

Thường nếu thư mục có quá nhiều tập tin ta có thể sử dụng đường ống để có thể trình bày danh sách tập tin theo từng trang màn hình. Ví dụ:

```
$ls -C | more
```

Sử dụng mkdir để tạo thư mục

mkdir [-mp] [thö muïc]

-m mode: đặt chủ quyền của thư mục theo số bát phân

-p : tạo các thư mục gián tiếp nếu tên thư mục gián tiếp trong đường dẫn là không tồn tại

Ví dụ:

- tạo thư mục con trong thư mục hiện hành

```
mkdir tam1
```

- tạo thư mục con sử dụng đường dẫn tương đối

```
mkdir tam1/chung
```

- tạo thư mục sử dụng đường dẫn đầy đủ

```
mkdir /usr/tam1/dung
```

- tạo các thư mục gián tiếp

```
mkdir -p /usr/tam2/duc
```

thư mục tam2 không tồn tại do đó tạo cả thư mục tam2 và duc.

Sử dụng lệnh pwd trình bày tên của thư mục hiện hành

```
$pwd
/usr/tam1/dung
```

Sử dụng rmdir để xoá thư mục

```
rmdir [-ps] thö muïc
```

-p : trình bày lần lượt các thư mục bị xoá

-s : xoá thông báo được tạo bởi -p

Ví dụ:

- xoá 1 thư mục rỗng: rmdir dung

- xoá 1 thư mục không rỗng: rmdir tam2/*

3. Các thao tác với tập tin

Tên tập tin trong UNIX có thể dài 256 ký tự, ngoại trừ các ký tự đặc biệt sau: ! “ ‘ ; / $ < > ( ) [ ] . { }. Ngoài ra ta cũng có thể sử dụng các ký tự sau:

* đại diện cho một, nhiều hoặc không ký tự nào.

? đại diện cho một ký tự đơn

[...] đại diện cho một dãy ký tự có thứ tự trong bảng Alphabet. Ví dụ: liệt kê tất cả các thư mục bắt đầu bằng chữ c, d, e: lc [cde]*

Liệt kê tập tin

```
ls [choïn löïa] thö muïc
```

- Liệt kê danh sách các tập tin

```
ls /thö muïc
```

- Hiển thị các tập tin ẩn

```
ls -a
lc -a
```

- Liệt kê thông tin về tập tin

```
ls -l
```

Xem nội dung 1 tập tin và nối kết các tập tin

```
cat [choïn löïa] [taäp tin]
```

- Xem

```
cat hosts
195.127.50.1 sunfibi
```

- Tổ hợp hai tập tin thành một

```
cat part1 part2 > report
```

Ta cũng có thể sử dụng lệnh cat để tạo 1 tập tin từ bàn phím.

Ví dụ:

```
cat > thu
```

đưa vào nội dung tập tin <=> copy con trên DOS

^d

hay có thể sử dụng ***more*** hoặc ***pg*** để xem từng trang màn hình:

more [teân taäp tin]

Vd: more thu

Đổi tên tập tin

mv teân_cuõ teân_môùi

mv teân thö_muïc

chuyển tập tin có *tên* vào *thư_mục*

Ví dụ:

- Thay đổi tên của tập tin ở thư mục hiện hành:

$mv a.out test

- Chuyển tập tin sang thư mục tam1

mv test tam1

tập tin mới sẽ là tam1/test

- Chuyển một vài tập tin sang thư mục khác

mv test1 test2 tam2

Sử dụng lệnh ln để gán thêm 1 tên mới cho 1 tập tin

ln teân1 teân2

ln teân thö_muïc

Ví dụ:

- Tạo tên thứ hai cho 1 tập tin

$ln test thu

$ls -l t*

3271 test

3271 thu --> caû hai taäp tin naøy coù cuøng 1 i-nodes

$pwd

/usr/dung

$ln /usr/phan/*.c

tất cả các tập tin trong thư mục /usr/phan mà thích ứng với *.c sẽ được nối với thư mục hiện hành.

Xoá tập tin

rm [-fri] taäp tin

-f : xoá các tập tin mà không hỏi, thậm chí chủ quyền ghi là không cho phép

-r : chấp nhận thư mục như đối số. Toàn bộ nội dung của thư mục bị xoá (kể cả các thư mục con) sau đó bản thân thư mục cũng bị xoá.

-i : trước khi xoá tập tin sẽ hỏi

Ví dụ:

- Xoá tập tin thu:

$rm thu

- Xoá tập tin có hỏi

```
$rm -i t*
test : ? y
thu : ? n
$rm -r tam
```

---> xoá tất cả các thư mục con của thư mục tam và bản thân thư mục tam.

Sao chép tập tin

```
cp taäp_tin_nguoàn taäp_tin_ñích
```

ví dụ:

```
cp /etc/passwd  /usr/dung/passwdold
```

Tìm kiếm 1 tập tin

```
find ñöôøng_daãn bieåu_thöùc_tìm_kieám
```

* Biểu thức tìm kiếm :

- name : tên tập tin --> tìm theo tên tập tin
- user : tên user --> tìm theo tên người sử dụng
- print : nếu tìm không thấy thì trình bày đường dẫn

v.v...

Ví dụ:

- Tìm tập tin thu :

```
$find -name thu -print
/usr/tam/thu
```

Tìm kiếm chuỗi văn bản bên trong tập tin

Sử dụng lệnh ***grep*** để tìm kiếm một chuỗi văn bản bên trong các tập tin được chỉ định

```
grep vaên_baûn caùc_taäp_tin
```

nếu chuỗi văn bản dài hơn 1 ký tự thì phải để trong hai dấu nháy.

Ví dụ:

- Tìm chuỗi ký tự "ngan hang" trong tập tin thu

```
$grep "ngan hang" thu
ngan hang cong thuong
```

4. Bảo vệ tập tin và thư mục

Thay đổi nhóm truy xuất của tập tin

```
chgrp nhoùm_caùc_taäp_tin
```

chỉ có superuser hay người sở hữu mới được quyền thay đổi

Ví dụ:

```
$ls -a test
-rwx--x--x 1 bin bin 13023 Jun 21 94 test
$chgrp data test
```

```
$ls -a test
-rwx--x--x 1 bin data 13023 Jun 21 94 test
```

Thay đổi người sở hữu tập tin

chown tên_ người_sử_dụng các_tập_tin

chỉ có superuser hay người sở hữu mới được quyền thay đổi

Ví dụ:

```
$chown dung test
$ls -a test
-rwx--x--x 1 dung data 13023 Jun 21 94 test
```

Đặc quyền truy xuất ngầm định

umask chủ_quyền (3 số bát phân)

Sau khi đặt ***umask*** tất cả các tập tin và thư mục tạo sẽ nhận **chủ quyền**.

Ví dụ:

```
$umask 177
$cat test
...
^D
$ls -l test
-rw------ 1 dung adm 1 Mar 11 10:11 test
```

Thay đổi chủ quyền của tập tin

Nếu bạn là người sở hữu hay superuser, bạn có thể sử dụng lệnh chmode để thay đổi chủ quyền của tập tin. Có hai cách:

chmode chuû quyeàn taäp tin

- Chủ quyền tuyệt đối: sử dụng các số bát phân, mỗi số đại diện cho chủ quyền của một lớp người sử dụng:

2 cho phép đọc

4 cho phép ghi

1 cho phép thực hiện

Ví dụ:

```
$chmode 754 test
$ls -l test
-rwxr-xr--
```

- Dùng ký hiệu tượng trưng:

chmode loại toán_tử [quyền] tên_tập_tin

Loại :

u : người sử dụng

g : nhóm

o : chung

a : tất cả

Toán tử :

+ : thêm quyền

- : bớt quyền

= : gán giá trị khác

Quyền :

r : đọc

w : ghi

x : thực hiện

s : đặt UID hay GID

Ví dụ:

```
$chmode g +w test
$ls -l test
-rwxrwxr-- ..... ..... test
```

5. Soạn thảo tập tin sử dụng vi

vi là một chương trình soạn thảo văn bản chuẩn trong UNIX. vi có hai chế độ:

- Chế độ lệnh: cho phép chèn, xoá, thay thế.

- Chế độ soạn thảo: cho phép soạn thảo văb bản

Để vào trình soạn thảo vi ta đánh:

```
vi teân_taäp_tin
```

Khởi đầu vi đặt ở chế độ lệnh; để vào chế độ soạn thảo đánh <i>, thoát khỏi chế độ này đánh ESC.

Thoát khỏi vi nhấn: X

Một số hàm lệnh của vi

vi tập tin --> bắt đầu dòng 1

vi +n tập tin --> bắt đầu ở dòng n

vi +/pattern --> bắt đầu ở pattern

vi -r tập tin --> phục hồi tập tin sau khi hệ thống treo

Di chuyển con trỏ

h	sang trái một khoảng trắng
e	sang phải một khoảng trắng
<space>	- nt -
w	sang phải 1 từ
b	sang trái 1 từ
k	lên một dòng
j	xuống một dòng
<return>	- nt -

)	cuối câu
(	đầu câu
}	đầu đoạn văn
{	cuối đoạn văn
^-w	đến ký tự đầu tiên chèn vào
^-u	cuốn lên 1/2 màn hình
^-d	kéo xuống 1/2 màn hình
^-z	kéo xuống 1 màn hình
^-b	kéo lên 1 màn hình

Chèn văn bản

i	trước dấu con trỏ
I	trước ký tự đầu tiên trên dòng
a	sau dấu con trỏ
A	sau ký tự đầu tiên trên dòng
o	dưới dòng hiện tại
O	trên dòng hiện tại
r	thay thế 1 ký tự hiện hành
R	thay thế cho đến khi nhấn <ESC>

Lệnh xoá

dw	1 từ
do	đến đầu dòng
d$	cuối dòng
3dw	3 từ
dd	dòng hiện hành
5dd	5 dòng
X	1 ký tự

Lệnh thay thế

cw	thay thế 1 từ
3cw	thay thế 3 từ
cc	dòng hiện hành
5cc	5 dòng

Lệnh tìm kiếm

*/and	từ kế tiếp của and
*?and	từ kết thúc là and
*/nThe	tìm dòng kế bắt đầu bằng The
n	lặp lại lần dò tìm sau cùng

Lệnh tìm kiếm và thay thế

:s/text1/text2/g	thay text1 bởi text2
:1,$s/tập tin/thư mục	thay tập tin bằng thư mục từ hàng 1 đến cuối
:g/one/s//1/g	thay thế one bằng 1

Thoát khỏi chương trình vi

:w	ghi tập tin
:x	ghi và thoát
:g	thoát không ghi
:g!	dù tập tin có thay đổi thoát vẫn không ghi.

IV/ Xuất nhập chuẩn, đổi hướng và đường ống

Các chương trình nhận dữ liệu nhập và tạo xuất phải có các kênh liên lạc để chuyển các thông tin đó. Đôi khi công việc này được thực hiện tường minh bởi chương trình "mở" một tập tin cụ thể. Ví dụ: giả sử bạn đánh lệnh.

```
sort tam
```

Việc này gây ra chương trình sort mở tập tin tam như vậy là sort có thể đọc tập tin để nhập dữ liệu.

Đa số các lệnh UNIX (cat, grep, wc,...) tự động mở 3 tập tin được gọi "nhập chuẩn", "xuất chuẩn" và "sai chuẩn" để nhập, xuất, nhận thông báo sai tương ứng. Trong UNIX thiết bị được xử lý như tập tin, như vậy ba tập tin chuẩn có thể là các thiết bị. Nhập chuẩn là bàn phím, xuất chuẩn và sai chuẩn là màn hình. Dĩ nhiên bạn có thể sử dụng "đổi hướng" để thay đổi các thiết bị xuất nhập chuẩn.

Đổi hướng

Mặc dù một vài lệnh như sort đã có sẵn chọn lựa để xác định tập tin xuất và nhiều lệnh cho phép bạn xác định tập tin nhập trên dòng lệnh. UNIX cung cấp một cơ cấu tổng quát hơn và rất mạnh: "đổi hướng" - cho phép gán lại các tập tin chuẩn. Chương trình vẫn nối kết đến ba tập tin xuất/ nhập chuẩn nhưng định danh của tập tin có thể bị thay đổi. Ví dụ lệnh cat (sử dụng xuất chuẩn) chúng ta có thể sử dụng > để tạo xuất chuẩn là một tập tin.

```
cat tam1 tam2 > baocao
```

nối các tập tin tam1 và tam2 gửi kết quả đến xuất chuẩn. Ký hiệu > baocao ... là tập tin xuất chuẩn sẽ là tập tin baocao thay vì màn hình.

Chú ý: đổi hướng xuất chuẩn tạo tập tin xuất đến tập tin xuất đã tồn tại nó sẽ bị xoá trước khi sử dụng.

Đổi hướng nhập cũng giống như xuất ngoại trừ là ký hiệu < được sử dụng để gán lại nhập chuẩn. Ví dụ:

```
sort < tam
```

lệnh sort sẽ lấy dữ liệu nhập trong tập tin tam thay vì từ bàn phím.

Cái gì là sự khác biệt giữa :

```
sort tam vaø sort < tam
```

Trong sort tam, lệnh sort mở thực sự tập tin tam, như vậy nó có thể sử dụng tam thay cho nhập chuẩn. Như vậy 2 tập tin nhập đều được mở: tam và nhập chuẩn.

Với đổi hướng dĩ nhiên tập tin tam là nhập chuẩn và chỉ 1 tập tin nhập là được mở. Hơn nữa với đổi hướng, hệ điều hành mở tập tin tam thay vì lệnh sort mở.

Đổi hướng xuất chuẩn không ảnh hưởng đến sai chuẩn. Và thông báo sai vẫn gửi ra màn hình, giả sử tập tin tam2 không tồn tại, ta đánh lệnh:

```
cat tam tam2 > baocao
tam2 : No such file or directory
```

Chỉ một tập tin có thể sử dụng để đổi hướng xuất hay nhập chuẩn, bởi vì mỗi lệnh chỉ nối với 1 xuất và 1 nhập.

Đa đổi hướng

Ta có thể sử dụng cả đổi hướng xuất và nhập cho một lệnh.

Ví dụ:

```
tr "[a-z]" "[A-Z]" < act1 > bigact1
```

Lấy dữ liệu nhập từ tập tin act1 gửi nó vào lệnh tr và xuất đến tập tin bigact1. Cụ thể là lệnh tr đổi tất ký tự chữ nhỏ trong tập tin act1 thành chữ lớn và xuất vào tập tin bigact1.

Thứ tự đưa vào chỉ thị đổi hướng là không quan trọng, ví dụ:

```
tr "[a-z]" "[A-Z]" > bigact1 < act1
```

Đường ống

Đổi hướng nối kết lệnh với tập tin. Đặc điểm đường ống của UNIX nối kết 1 lệnh với 1 lệnh khác. Đặc biệt hơn nó tạo xuất chuẩn của 1 lệnh thành nhập chuẩn của 1 lệnh khác. Ký hiệu đường ống (|) được sử dụng để thiết lập đường ống. Ví dụ:

```
wc baocao* | sort -n
```

nhận xuất 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. Ví dụ:

```
wc baocao* | sort -n > rep-count
```

kết quả sẽ đưa ra tập tin rep-count.

Tổ hợp các tập tin với nhập chuẩn

Nhiều lệnh UNIX có thể sử dụng nhập chuẩn như một danh sách các tập tin nhập. Xem 3 ví dụ sau:

lp tam	lệnh mở tập tin tam
lp < tam	lệnh sử dụng nhập chuẩn
cat tam | lp	lệnh sử dụng .....

Tất cả đều in tập tin tam nhưng nó làm việc khác nhau. Lệnh đầu mở tam và bỏ qua nhập chuẩn. Lệnh thứ hai gây ra shell mở tam và coi nó là nhập chuẩn. Lệnh thứ ba coi xuất chuẩn của cat là nhập chuẩn của lệnh lp.

Một vài lệnh UNIX cho phép bạn sử dụng một danh sách các tập tin nhập bằng cách sử dụng ký hiệu (-) để thể hiện nhập chuẩn như 1 danh sách các tập tin. Ví dụ:

```
sort tam | lp tam1 - tam2
```

lp sẽ in đầu tiên là tam1 kế đó là kết quả sắp thứ tự của tập tin tam, cuối cùng là tập tin tam2.

Các dạng khác của đổi hướng

Đổi hướng thêm vào: >>

Sử dụng khi bạn muốn thêm dữ liệu vào cuối của một tập tin đã tồn tại.

Ví dụ:

```
cat baocao2 baocao3 >> baocao
```

thêm nội dung của baocao2 và baocao3 vào cuối của tập tin baocao.

Nếu tập tin đích không tồn tại thì nó sẽ được tạo.

Đổi hướng "Tập tin ở đây": <<

Giả sử bạn có thông báo chuẩn bạn có mail từ người sử dụng khác, bạn có thể thiết lập 1 shell script:

```
mail $* < message
```

message là tên tập tin chứa thông báo chuẩn. $* thể hiện các đối số của shell script.

Vấn đề với script này là lệnh và văn bản thông báo là hai tập tin riêng rẽ. Dĩ nhiên bạn có thể tránh vấn đề này bởi sử dụng << lấy chỗ của văn bản trong cùng tập tin lệnh. Văn bản như vậy được gọi là "Tài liệu ở đây".

script mới có dạng:

```
mail $* << stop
Please remember to pay your monthly file insurance
--Big Jake
stop
echo $* reminded on 'date' >> $HOME/fi.log
```

Các bộ mô tả tập tin và đổi hướng sai chuẩn: 2>

Đôi khi chúng ta muốn đổi hướng thông báo sai. Ví dụ khi chạy lệnh ở chế độ nền &.

Ví dụ:

```
spell baocao &
```

Chạy ở chế độ nền (giải phóng đầu cuối cho công việc khác) vẫn xuất các từ sai chính tả ra màn hình. Điều này gây rối cho các công việc khác.

Để tránh việc này, bạn phải sử dụng đổi hướng sai chuẩn với bộ mô tả tập tin (0: nhập chuẩn, 1: xuất chuẩn, 2: sai chuẩn).

Ví dụ:

```
spell baocao > baocaodung 2> baocaosai &
```

các từ sai sẽ được đưa ra tập tin baocaosai.

Tổ hợp xuất: 2>&1

Đôi khi bạn muốn đổi hướng cả hai xuất chuẩn và xuất sai đến cùng một tập tin. Việc này có thể làm với tổ hợp 2>&1.

Ví dụ:

```
spell baocao > baocaodung 2>&1 &
```

baocaodung sẽ chứa cả xuất và các thông báo sai.

Lệnh tee

Hoạt động đổi hướng và đường ống là đặc điểm của hệ điều hành UNIX. Tuy nhiên ta cũng có thể sử dụng 1 lệnh của UNIX để làm việc này. Đó là lệnh **tee**, nó sẽ giảm bớt các kết quả gián tiếp của chuỗi đường ống.

Ví dụ:

```
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.

V Thư điện tử

Thư điện tử (mail) là một tiện ích do UNIX cung cấp để gửi và nhận thông báo. Có hai lệnh mail và mailx.

1. mail

```
$mail
```

Thư điện tử cung cấp các khả năng cơ bản để gửi và nhận thông báo.Mail sẽ hiển thị thông báo theo thứ tự vào trước ra sau.Sau khi hiển thị mỗi thông báo mail sẽ hiện lên dấu ? để chờ lệnh của người sử dụng. Có thể có các lệnh sau :

- *newline* Hiển thị thông báo kế, nếu không còn thì thoát khỏi mail.
- \+ Giống như newline
- *p* In thông báo
- *s* [tập tin] Cất thông báo vào tập tin hoặc mbox
- *w*[tập tin] Giống như s nhưng không cất đầu thông báo
- *d* Xóa thông báo
- *q* Thoát khỏi mail
- *x* Thoát khỏi mail mà không thay đổi thông báo
- *!* lệnh Thực hiện lệnh Unix

Gởi thư : Đưa vào lệnh mail với địa chỉ của người sử dụng. Ví dụ :

```
$ mail dungø@fibi.hcm.vn.com
  <thoâng baùo>
  ^-D
```

Thông báo sẽ được gửi cho người sử dụng có tên là dung ở công ty fibi vùng hcm.vn.com. Có thể cùng một lúc gửi một thông báo cho nhiều người :

```
$ mail dung@fibi.hcm.vn.com trung@fibi,hanoi.vn.com
```

Nhận thư : Khi login vào hệ thống nếu có thư hệ thống sẽ thông báo " You have mail" khi đó có thể đánh $mail để xử lí mail.

2. mailx

```
$mailx
```

Mailx bao gồm các lệnh để truyền và nhận thư. Lệnh có dạng :

```
Leânh [danh saùch thoâng baùo] [caùc ñoái soá]
```

[danh sách thông báo] có thể là :

n	Thông báo số n
$	Thông báo cuối cùng
*	Tất cả thông báo
n-m	Từ thông báo n đến m
:d	Xóa tất cả thông báo
:n	Thông báo mới
:r	Thông báo chỉ đọc
:u	Thông báo chưa đọc
:o	Thông báo cũ

Lệnh

help	Hiển thị cú pháp của lệnh
ignore	Bỏ qua đầu tập tin
print	Hiển thị thông báo
Print	In thông báo
top	Hiển thị một vài dòng đầu của thông báo
repond	Trả lời một thông báo
copy	Chép thông báo lên tập tin
list	Hiển thi danh sách lệnh
save	Cất thông báo lên tập tin
quit	Thoát khỏi mailx

VI. UNIX Shell

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. Nó là mặc định trong đa số các hệ UNIX (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. Nó 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. Nó có thể gọi bởi ksh.

Phần này sẽ trình bày cả ba loại shell. Với Korn shell sẽ có chú thích riêng.

1. Các khái niệm cơ bản

Các lệnh

Lệnh đơn giản:

Là một chuỗi từ được phân cách bởi khoảng trắng. Từ đầu là lệnh Unix hoặc lệnh shell nội tại. Các từ khác hoặc là tên tập tin xử lí hoặc là các đối số làm thay đổi hành vi của lệnh.

Lệnh trong môi trường Unix thường xuyên trả lại một giá trị số gọi là trạng thái thoát khi shell kết thúc. Nếu lệnh hoàn thành trạng thái thoát bằng không. Nếu trạng thái thoát khác không có nghĩa là lệnh gặp diều kiện không bình thường hoặc là kết thúc bất thường. Với phát biểu if giá trị 0 tương ứng với diều kiện đúng.

Khi lệnh kết thúc bất thường nó trả về giá trị 128 + trạng thái, với trạng thái là giá trị gây ra kết thúc bất bình thường. Ví dụ nếu nhấn Del để kết thúc lệnh giá trị trả về là 130 (128+02).

Đường ống:

Bao gồm một hay nhiều lệnh đơn giản dược phân cách bởi kí hiệu (|). Xuất của mỗi lệnh được gửi đến nhập của lệnh kế. Mỗi lệnh được thực hiện như một quá trình riêng rẽ. Trạng thái thoát của đường ống là trạng thái thoát của lệnh cuối.

Chuỗi lệnh:

Bao gồm một hoặc nhiều lệnh đơn giản (hay đường ống) được phân cách bởi (;), (&), (&&), (||). Trong một danh sách lệnh các dấu phân cách được xử lí từ trái sang phải với độ ưu tiên:

&& và ||

; và &

Dấu (;) gây ra thực hiện tuần tự các lệnh trong danh sách lệnh. Shell đợi mỗi lệnh hoàn thành trước khi thực hiện lệnh kế.

Dấu (&) gây ra thực hiện bất đồng bộ các lệnh trong danh sách lệnh. Shell thực hiện lệnh mà không đợi lệnh trước hoàn thành.

Dấu (&&) gây ra lệnh sau được thực hiện chỉ khi lệnh trước trả về giá trị 0. Ví dụ :

```
mail -s && echo You have mail
```

Sẽ trình bày thông báo You have mail chỉ khi lệnh mail -s trả về giá trị không.

Dấu (||) gây ra lệnh sau được thực hiện chỉ khi lệnh trước trả về giá trị khác không Ví dụ :

```
mail -s ||  echo No mail
```

Sẽ trình bày thông báo No mail chỉ khi lệnh mail -s trả về giá trị khác không.

Nhóm lệnh

Có thể nhóm một chuỗi lệnh trong (). Nhập và xuất chuẩn có thể đổi hướng cho toàn nhóm. Ví dụ :

trong lệnh $ ls; who; date > status chỉ có xuất của lệnh date là được gửi tới tập tin status.

trong lệnh $ (ls; who; date) > status xuất của cả ba lệnh là được gửi tới tập tin status.

trong lệnh $ (ls; who; date) > status & cả ba lệnhđều chạy ở chế độ nền.

Sự thay thế lệnh

Xuất của một lệnh có thể sử dụng như nhập của lệnh khác hay gán biến với sự thay thế lệnh. Khi shell gặp lệnh nằm trong ‘ ‘, lệnh sẽ được thực hiện, xuất của nó trở thành nhập của lệnh khác hay gán cho biến. Ví dụ :

$ here=‘pwd’ thực hiện lệnh pwd trước sau đó gán cho biến here.

Chú thích: *sử dụng dấu #*

Biên dịch khoảng trắng:

Sử dụng biến phân cách vùng IFS để thay đổi kí hiệu phân cách khoảng trắng. Ví dụ :

```
$ IFS=:
$ ls:-l
```

Dấu nháy đơn

Shell không diễn dịch các dòng kí tự giữa hai dấu nháy đơn. Ví dụ :

```
$ echo `$TERM`
$TERM
```

Dấu nháy đôi

Shell không diễn dịch các dòng kí tự giữa hai dấu nháy đôi ngoại trừ các dấu ($,\,’). Ví dụ :

```
$ name= “Ng Van A”
$ echo $name
Ng Van A
$ echo “$name
Ng Van A
```

Dấu \ ngăn ngừa shell diễn dịch kí tự trực tiếp sau nó.

Tên tập tin:

Có thể là tên đầy đủ hay tên wild card. Trước khi thực hiện lệnh shell quét tên wild card. Có các wild card sau:

kí tự	ý nghĩa
*	tương ứng với mọi kí tự
?	tương ứng với một kí tự
[xyz]	tương ứng với ba kí tự x,y,z
[a-z]	tương ứng với các kí tự từ a đến z
[!a-z]	tương ứng với các kí tự khác a đến z

Dấu nhắc

Sử dụng biến PS1 để thay đổi dấu nhắc.

Sự thực thi:

Shell thực hiện lệnh như sau :

- ✓ Giải quyết các tên tập tin, dấu nháy, sự thay thế lệnh, sự thay thế biến, và đổi hướng xuất nhập.
- ✓ Thực hiện các lệnh nội tại.
- ✓ Tên lệnh được so sánh với tên của các hàm đã được định nghĩa. Nếu tương ứng các hàm sẽ được thực hiện
- ✓ Nếu lệnh không phải là lệnh nội tại hay hàm một quá trình mới được tạo và shell sẽ thực hiện lệnh bởi gọi hàm exec.

Shell sử dụng biến PATH để tìm lệnh. Giá trị mặc định là thư mục hiện hành kế đó là thư mục /bin sau đó là /usr/bin.

Quá trình nền:

Shell có thể chạy ở chế dặ nền hay tương tác. Khi lệnh chạy ở chế độ nền shell trở lại dấu nhắc ngay lập tức. Người sử dụng có thể đưa vào lệnh khác. Ví dụ :

```
$ sort large_file >file_out &
1222
```

Khi lệnh nền chạy ID-quá trình của nó sẽ dược trình bày. Cũng có thể sử dụng lệnh ps dể trình bày ID-quá trình. Sử dụng lệnh kill dể hủy bỏ quá trình. Ví dụ :

```
$ kill -2 1222
```

Nếu muốn lệnh chạy nền bỏ qua tín hiệu hang up thì có thể sử dụng

```
$ nohupsort database >db.out &
1245
```

Có thể sử dụng lệnh wait để đợi quá trình nền và nhận giá trị trả về của nó. Ví dụ :

```
$ wait 1245
```

Ngoài ra đối với ksh còn có cơ cấu kiểm soát công việc :

```
$ sort large > sort.out &
[1] 1923
```

[1] là chỉ số công việc.

Gọi lại lệnh:

Khi ksh thực hiện lệnh nó đưa văn bản lệnh vào tập tin $HOME/.history. Có thể đổi tên tập tin bởi biến HISTFILE. số lệnh có thể chứa là ở biến HISTSIZE mặc định là 128. Có thể sử dụng lệnh fc để trình bày HISTFILE. Ví dụ :

```
$ fc -l 2 12
```

danh sách các lệnh từ 2 đến 12

2. Xử lí quá trình

Hệ thống Unix dùng các quá trình như là phương tiện để chạy các chương trình.Mỗi lệnh mà người dùng nhập vào làm tạo ra một quá trình.Một quá trình là một chương trình đang chạy.Ngoài ra, quá trình còn kết hợp cả thông tin về người dùng và môi trường hệ thống.Mỗi quá trình có một

mã số nhận dạng (ID) riêng liên kết với nó và hệ điều hành sẽ dùng mã số này để phân biệt các quá trình.

Quá trình tạo ra một quá trình khác đực gọi là quá trình cha.Các quá trình được tạo ra bởi các quá trình cha được gọi là các quá trình con.Quá trình con nhận biết cha nó qua ID của quá trình cha.

Mỗi quá trình cũng có một ID nhóm quá trình liện kết với nó.Một nhóm quá trình bao gồm tất cả các quá trình có cùng ID nhóm quá trình.Mỗi vỏ đăng ký mỗi lúc chỉ có thể có một ID nhóm quá trình liên kết với nó.Quá trình này gọi là quá trình .Nếu một quá trình của người sử dụng không nằm trong ID của nhóm quá trình liên kết với vỏ đăng ký của người sử dụng thì quá trình đó là một quá trình nền.

Cùng được liên kết với mỗi quá trình là ID của người sử dụng thực và hiệu quả cũng như ID của nhóm thực và hiệu quả.Các ID của người sử dụng thực và quá trình thực được thiết lập bởi quá trình đăng ký và được tất cả các quá trình mà người sử dụng tạo ra thừa hưởng.ID của người sử dụng hiệu quả thường thì giống với ID của người sử dụng thực nhưng có thể được thay đổi bởi các chương trình có thể thiết lập ID người sử dụng (đó là su).Cũng vậy, các ID của nhóm thực và hiệu quả thường cũng giống nhau nhưng ID của nhóm hiệu quả có thể được thay đổi bởi các chương trình có thể thiết lập ID nhóm (ví dụ, mail).Các ID của nhóm hiệu quả và người sử dụng hiệu quả của một quá trình được dùng để xác định quyền truy nhập tập tin và thư mục của một quá trình (đọc, ghi, thực hiện hay tìm kiếm).

Các quá trình tạo bởi lệnh gọi hệ thống fork hay exec (một lệnh gọi hệ thống là một lời gọi của chương trình gửi đến hệ điều hành để yêu cầu các phục vụ hệ thống).

Lệnh gọi hệ thống fork :

Quá trình cha tạo ra một quá trình con bằng một lệnh gọi hệ thống fork.

hinh 1

Lệnh fork tạo ra một bản sao của quá trình gốc giống như quá trình cha ngoại trừ giá trị ID-quá trình của nó. Quá trình cha có thể tạm ngưng sự thực hiện của nó cho đến khi quá trình con kết thúc. Thông thường shell thực thi lệnh bằng cách tạo quá trình con sau đó đợi quá trình con hoàn thành trước khi tạo dấu nhắc kế. Quá trình con thừa kế các thông tin môi trường sau từ cha:

- ✓ ID-quá trình cha
- ✓ ID nhóm-quá trình
- ✓ Các tập tin mở
- ✓ Thư mục làm việc
- ✓ Mặt nạ tạo tập tin (umask)
- ✓ ID-user chạy
- ✓ ID-nhóm chạy
- ✓ Giới hạn kích thước tập tin tạo
- ✓ Giới hạn bộ nhớ tối đa
- ✓ Các tác động đặt signal
- ✓ Các biến

Khi quá trình hoàn thành, kiểm soát trở lại quá trình cha. Các tài nguyên được cấp phát sẽ được giải phóng.

Gọi hệ thống exec

Gọi hệ thống exec nạp chương trình vào vùng bộ nhớ của quá trình hiện hành. Không giống như fork, exec không tạo quá trình mới.

hinh 2

Quá trình login

Các sự kiện xảy ra khi login sẽ mô tả cách thức làm việc của fork và exec. Quá trình /etc/init (PID 1) chạy khi khởi động hệ thống và sẽ tiếp tục chạy cho đến khi ngưng. Quá trình /etc/init là một quá trình deamon.

/etc/init xem xét /etc/inittab tập tin này chứa các dòng lệnh phải thực hiện với các mức hệ thống khác nhau. Ví dụ như xuất login : hoặc đặt chế độ cho terminal

```
11:234:respawn:/etc/getty tty11 9600
```

respawn chỉ thị là mỗi khi getty chết nó sẽ được khởi động lại bởi /etc/init

hinh 3

Thực hiện lệnh đơn giản:

Khi đưa vào lệnh đơn giản shell fork một shell khác và sau đó exec lệnh. Shell cha đợi cho đến khi quá trình con hoàn thành

hinh 4

Thực hiện chuỗi lệnh:

Khi đưa vào chuỗi lệnh shell fork một shell con và sau đó gọi exec để thực hiện lệnh đầu tiên. Khi lệnh đầu hoàn thành shell fork shell khác và sau đó gọi lệnh exec để thực hiện lệnh thứ hai v.v...

hinh 5

Thực hiện nhóm lệnh:

Khi đưa vào nhóm lệnh shell fork một shell con để thực hiện các lệnh. Shell con này gọi exec để thực hiện lệnh đầu tiên. Khi lệnh đầu hoàn thành shell fork shell khác và sau đó gọi lệnh exec để thực hiện lệnh thứ hai vvv.

hình 6

Thực hiện lệnh đường ống:

Khi đoán nhận đường ống (|) shell khởi động mỗi quá trình trong dòng lệnh và thiết lập liên lạc giữa chúng thông qua một tập tin đường ống đặc biệt. Các lệnh trong đường ống được thực hiện từ phải sang trái.

hình 7

Thực hiện lệnh nền:

Khi đưa vào lệnh nền shell fork shell con` sau đó gọi lệnh exec để thực hiện lệnh. Kiểm soát trả lại quá trình cha ngay. Shell khi đó có thể thực hiện các lệnh khác.

hình 8

3. Lệnh nội tại

Các lệnh nội tại là một phần của shell. Có các lệnh sau :

. taäp tin

Shell hiện hành đọc và biên dịch lệnh trong tập tin

alias [teân[=giaù trò]...] (ksh)

Lệnh alias cho phép đổi tên các lệnh trong shell. Ví dụ :

```
$ alias dir="ls "
$ alias opt="-CF"
$dir opt
bin/   unit01.n     unit02.n
status *
```

bg %chỉ số công việc	đặt công việc ở chế độ nền
break	Thoát khỏi các cấu trúc for,while ...
cd [thư mục]	Chuyển thư mục
continue	Lặp các cấu trúc
echo [đối số]	Hiển thị nội dung đối số
exec [lệnh]	Gọi lệnh thực hiện
exit	Kết thúc shell
export	Đưa các biến môi trường thành các biến chung
fc	Hiển thị tập tin history
fg %chỉ số công việc	Đặt công việc ở chế độ tương tác
kill ID-quá trình	Xóa đi một quá trình
newgrp [nhóm]	Đổi nhóm của người sử dụng
read biến	Đọc dữ liệu từ bàn phím
pwd	Hiển thị thư mục hiện hành
return	Trả về hàm gọi
set	Đặt biến môi trường
test [biểu thức]	Kiểm tra biểu thức. Ví dụ :
-r tập tin	Đúng nếu tập tin tồn tại và có thể đọc vvv.
ulimit [kích thước]	Định kích thước tối đa tập tin tạo
*unalia*s tên	Xóa alias một tên

umask [mặt nạ]	Định mặt nạ chủ quyền tập tin	
Giá trị mặt nạ	Chủ quyền tập tin	Chủ quyền thư mục
000	-rw-rw-rw-	drwxrwxrwx
002	-rw-rw-r--	drwxrwxr-x
007	-rw-rw----	drwxrwx---
022	-rw-r--r--	drwxr-x---
027	-rw-r-----	drwxr-x---
077	-rw-------	drwx------
uset	Xóa các biến môi trường	
wait [PID]	Đợi quá trình PID hoàn thành sau đó báo cáo trạng thái kết thúc của nó.	

4. Biến và tham số shell

Các tham số shell được sử dụng chứa các giá trị của shell hiện hành hay các shell con để sử dụng sau đó. Có các loại tham số sau:

- ✓ Tham số vị trí
- ✓ Tham số đặc biệt
- ✓ Tham số được đặt tên (biến)

4.1 Tham số vị trí

Tham số vị trí là các giá trị truyền qua khi shell hay shell con được gọi. Có thể đặt các tham số vị trí như các đối số của shell script và các tham số này được đánh số 1,2 vvv. Tên của lệnh hiện hành được tham chiếu bởi $0. Các tham số khác sẽ là $1, $2 vvv.

4.2 Tham số đặc biệt

Tham số đặc biệt được đặt tự động khi shell được gọi.Chúng có một ý nghĩa đặc biệt không thể thay đổi.

Biến	ý nghĩa
$@	Tham số vị trí như “$1 $2 $3 $4...”
$@	Tham số vị trí như “$1”, “ $2”, “ $3”, “ $4” vvv.
$#	Chỉ số của tham số vị trí truyền đến shell
$?	Mã thoát của lệnh đồng bộ cuối cùng
$$	Chỉ số quá trình của shell hiện hành
$!	Chỉ số của lệnh nền cuối cùng được kéo bởi shell hiện hành

4.3 Tham số được đặt tên

Tham số được đặt tên được gọi là biến shell và phải bắt đầu với một chữ sau đó có thể là chữ hay số. Để đặt biến sử dụng dấu =. Ví dụ :

```
$ ten = “ Ng Van A”
$ echo $ten
```

Ng Van A

Dấu {} phải được sử dụng với tên biến theo sau bởi chữ hay số mà không phải là một phần của tên biến. Ví dụ :

```
$ filename=chapt
$ echo ${filename}0
chapt0
```

Các biến được đặt bởi shell(ksh) ;

ERRNO	giá trị sai của lần gọi hệ thống cuối
LINENO	Chỉ số dòng hiện hành trong shell script
OLDPWD	Giữ thư mục hiện hành cũ được đặt bởi lệnh cd
PPD	Chỉ số quá trình của shell cha
PWD	Thư mục hiện hành

Các biến được sử dụng bởi shell(ksh) ; Các biế này thường được đặt bởi quản trị hệ thống trong tập tin /etc/profile hay bởi người sử dụng trong tập tin .profile

CDPATH	Đường dẫn để tìm lệnh cd
HOME	Thư mục "home"
PATH	Đường dẫn để tìm lệnh
PS1	Dấu nhắc thứ nhất mặc định "$ "
PS2	Dấu nhắc thứ hai mặc định "> "
TERM	Loại Terminal

5. Cấu trúc lập trình

+ for

```
for bieán in töø
do
      danh saùch leänh
done
```

```
for bieán
do
      danh saùch leänh
done
```

Ví dụ :

```
$ ls unit??.n
unit01.n      unit02.n      unit04.n      unit12.n
$ for i in unit??.n
> do
> spell $i > ${i}_sp
> done
$ ls unit??*
```

. unit01.n unit02.n unit04.n unit12.n

unit01.n_sp unit02.n_sp unit04.n_sp unit12.n_sp

Hay ta có thể thu thập các tập tin xuất vào một tập tin

```
for i in unit??.n
do
    echo "$i:"
    spell $i
    echo "\n"
done > spell.out
```

+ while

```
while leänh
do
  danh saùch leänh
done
```

Thực hiện danh sách lệnh chừng nào trạng thái thoát của lệnh khác không. Nếu ngay lần đầu lệnh đã trả về giá trị khác không thì danh sách lệnh sẽ không được thực hiện. Ví dụ :

```
while who | grep NVA > /dev/null
do
    sleep 10
done
echo "NVA daõ log off " > /etc/w_NVA
```

+ until

```
until leänh
  do
    danh saùch leänh
done
```

Thực hiện danh sách lệnh chừng nào trạng thái thoát của lệnh bằng không. Nếu ngay lần đầu lệnh đã trả về giá trị bằng không thì danh sách lệnh sẽ không được thực hiện. Ví dụ :

```
until who | grep NVA > /dev/null
do
    sleep 10
done
echo "NVA daõ log on"
```

+ if

```
if leänh
then
   danh saùch leänh
fi
```

```
if leänh
then
    danh saùch leänh
else
   danh saùch leänh
fi
```

```
if leänh
then
    danh saùch leänh
elif leänh
then
   danh saùch leänh
fi
```

Nếu lệnh trở lại giá trị không danh sách lệnh sau then sẽ được thực hiện.

```
if who | grep NVA > /dev/null
then
  echo  “NVA ñaõ ø log on”
fi
```

```
if  [ -f status ]
then
    echo “ status laø taäp tin thöôøng”
else
  echo “ status khoâng phaûi laø taäp tin thöôøng”
fi
```

```
if [ -f status ]
then
   echo “ status laø taäp tin thöôøng”
elif [ -d status ]
then
  echo “ status laø thö muïc”
fi
```

+ case

```
case töø in
 pattern1)
      danh saùch leänh
     ;;
pattern2)
      danh saùch leänh
     ;;
```

```
esac
```

Lệnh case sẽ thực hiên danh sách lệnh liện hệ với pattern đầu tiên thích ứng với từ. Mỗi pattern được kết thúc bởi (;;). Ví dụ :

```
 echo " Ñöa vaøo leänh : \c"
read cmd
case $cmd in
   ls)
      /bin/ls ;;
  who)
     /bin/who;;
 date)
     /bin/date;;
  *)
      echo " Leänh sai";;
esac
```

+ Hàm

```
teân()
{
  caùc leänh
 }
function
{
  caùc leänh
 }
```

Các hàm được đọc vào shell hiện hành. Để gọi hàm chỉ cần gọi bằng tên (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.Hàm có thể truy xuất tập hợp biến của shell hiện hành. Ví dụ :

```
dir()
{
   /bin/ls -C
}
```

6. Shell script

Shell script là một tập tin chứa các lệnh nội tại,hàm và lệnh unix. Shell script thường sử dụng khi một chuỗi lệnh phải thực hiện nhiều hơn một lần. Để tạo shell script sử dụng một chương trình hiệu đính để đưa các lệnh muốn thực hiện vào tập tin.

Để thực hiện shell script có thể làm như sau :

Thay đổi chủ quyền tập tin đến chế độ thực hiện được. Sau đó chạy shell script như một lệnh bình thường.

hoặc sử dụng lệnh sh $ sh myprog

Tên của shell script không nên trùng với các lệnh UNIX,lệnh nội tại, hàm. Ví shell script thực hiện đầu tiên.

Để gỡ rối shell script sử dụng $ sở hữu -x myprog

Có thể sử dụng lệnh chấm (.) thực hiện chương trình trực tiếp shell script trong môi trường của nó. Ví dụ :

```
$ . .profile
```

Đọc và thực hiện các lệnh trong tập tin .profile

VII Sử dụng openwin

VIII Mạng UNIX

Mạng cho phép người sử dụng truy xuất và chia sẻ tài nguyên của các máy cách xa nhau. Các máy tính nối kết trên mạng được gọi là "host". Máy tại chỗ gọi là máy chủ cục bộ (local host). Tất cả các máy khác gọi là máy chủ từ xa (remote host). Mạng cục bộ (LAN) là tập hợp của các máy gần nhau (trong phạm vi 2km). Mạng rộng (WAN) là tập hợp của các máy có khoảng cách địa lí xa nhau (có thể hàng ngàn cây số). UNIX cung cấp cả hai khả năng nối kết mạng LAN và WAN, với hai giao thức :

UUCP (Unix-to Unix CoPy) - Nối kết giữa hai hệ UNIX với nhau.

TCP/IP (Transmission Control Protocol/ Internet Protocol) - Nối kết giữa các hệ máy chủ khác nhau. Đặc biệt nó là một giao thức chuân được sử dụng trong mạng Internet - Mạng WAN phổ biến nhất thế giới.

1. Giao thức TCP/IP

Giao thức TCP/IP được thiết lập bởi tổ chức Internet. TCP/IP là một tập các giao thức (FTP,SMTP,Telnet vvv) mà định nghĩa chuẩn cho các loại truyền thông khác nhau (Truyền tập tin, thư điện tử, mô phỏng terminal vvv). Hiện nay TCP/IP là giao thức được sử dụng phổ biến nhất.

Các giao thức TCP/IP được cài đặt như các ứng dụng Client/Server. Một "client" sử dụng để yêu cầu sự thực hiện một tác động bởi máy "server" từ xa. Máy "server" có trách nhiệm xử lí yêu cầu của máy "client". Dữ liệu truyền qua LAN và WAN theo từng khối (packets). Có rất nhiều loại LAN và WAN khác nhau tùy theo cách thức chúng truyền packets. LAN phổ biến nhất là Ethernet.

Một host mà có phần cứng cho cả LAN và WAN được gọi là "gateway" vì chúng có khả năng phân đường giữa LAN và WAN. Để cho các máy PC sử dụng các phần cứng LAN,WAN các bộ điều khiển thiết bị phải được cái đặt, Các bộ điều khiển cho LAN có thể là NDIS, ODI,ASI,PDS. Các bộ điều khiển cho WAN có thể là X.25,PPP và SLIP.

1.1 Các thành phần của TCP/IP

quá trình của người sử dụng quá trình của người sử dụng

TCP UDP

ICMP IP ARP RARP

giao tiếp phần cứng

TCP (Transmission Control Protocol) Giao thứ hướng đến nối kết : tin cậy, hai chiều, kiểm soát sự truyền nhận các dòng dữ liệu. Đa số các ứng dụng sử dụng TCP.
UDP (User Datagram Protocol) Giao thức không nối kết : không tin cậy, dòng dữ liệu truyền không cần kiểm tra nhận đươc hay không.

ICMP (Internet Control Message Protocol) Giao thức điều khiển sai và kiểm soát thông tin giữa "gateway" và các máy chủ.

IP (Internet Protocol) Giao thức cung cấp các dịch vụ chuyển gói (packet) để ch TCP, UDP và ICMP.

ARP Address Resolution Protocol Giao thức ánh xạ địa chì Internet vào địa chỉ phần cứng.

RARP Revese Address Resolution Protocol Giao thức ánh xạ địa chỉ phần cứng vào địa chỉ Internet.

1.2 Các khái niệm cơ bản

Địa chỉ IP (IP Address) : Chỉ số duy nhất để định danh mỗi máy trong mạng. Có dạng sau :

	Chỉ số mạng	Chỉ số mạng con	Chỉ số máy
Ví dụ :	192.127.	50.	1

Chỉ số mạng : Nhận từ Network Information Center(NIC), Không thể thay đổi

Chỉ số mạng con : Tùy chọn, chỉ số mạng con mà máy nối vào.

Chỉ số máy : Xác định duy nhất cho mỗi máy trên mạng. Do người quản trị hệ thống đặt khi tạo mạng hay thêm máy.

Địa chỉ IP được biểu diễn bằng 4 byte : 3 byte đầu dành cho chỉ số mạng và mạng con. Byte cuối dành cho địa chỉ máy. Mỗi byte cách nhau bởi (.). Ví dụ : 195.127.50.1

Mặt nạ mạng con (subnet mask) :Số sử dụng bởi phần mềm để phân cách địa chỉ mạng con và phần còn lại của địa chỉ IP. Ví dụ : 255.255.255.0 ba byte đầu biểu diễn chỉ số mạng và mạng con.

Địa chỉ chung (Broadcast Address): Được sử dụng để lan truyền thông báo đến tất cả các máy trong mạng. Ví dụ : 195.127.50.255. Khi sử dụng địa chỉ này để gửi thông báo thì tất cả các máy trong mạng 195.127.50 sẽ nhận được thông báo.

Tên vùng (domain name) : Tên dàng cho một nhóm máy được quản trị đống thời. Tên vùng có tính phân cấp, các thành phần được phân cách bởi dấu chấm (.). Mức vùng được tính từ phải sang trái. Mức vùng cao nhất có thể là :

.COM	lãnh vực thương mại
.EDU	lãnh vực giáo dục
.GOV	lãnh vực chính phủ

Tên vùng đầy đủ phải duy nhất trên toàn thế giới. Ví dụ: hcm.vn.com

Dịch vụ tên vùng (Domain Name Service) : dịch vụ tên của họ TCP/IP, cung cấp các thông tin về các máy cục bộ cũng như từ xa.

Tên máy (host name): Có thể là địa chỉ IP hay một chuỗi kí tự mà là tên hình thức của máy. Được định nghĩa trong tập tin /etc/host. Ví dụ : trong tập tin /etc/host có dòng

195.127.50.1 sunfibi

thì sunfibi là tên máy.

1.3 Các ứng dụng

Dựa trên TCP/IP có rất nhiều ứng dụng khác nhau :

Telnet : cho phép các terminal liên lạc với nhau theo từng kí tự. Nó cũng cho phép mô phỏng PC hnư một terminal của máy chủ.

Ftp (File Transfer Protocol): Cho phép truyền tập tin giữa các máy (mạng) cách xa nhau.

Dịch vụ tên vùng (DNS) : cung cấp các ánh xạ Tên-Địa chỉ.

Ngoài ra còn có các lệnh như rcp,rsh vvv. Dùng để thực hiện từ xa hay sao chép các tập tin ở các máy từ xa.

Các lệnh cơ bản trên (DOS,UNIX):

+ Kiểm soát đường truyền :

```
ping <host name> [time out]
```

timeout thời gian cồ gắng nối kết. Ví dụ :

```
$ping sunfibi
```

Ping gửi một ICMP tới hostname. Nếu host có đáp ứng thì sẽ có thông báo : sunfibi is alive. Nếu không : No answer from sunfibi.

+ Mô phỏng terminal :

Cho phép terminal nối kết với máy chủ từ xa. Có hai loại terminal có thể mô phỏng VT100 hay IBM3270. Ví dụ :

```
$rlogin sunfibi
login:
password:
```

+ Truyền tập tin :

Cho phép truyền và nhận tập tin với máy từ xa. Ví dụ :

```
$ftp sunfibi
login:
password:
>
```

Tại dấu nhắc này có thể đưa vào các lệnh hay sử dụng lệnh help để liệt kê các lệnh có thể sử dụng. Có một vài lệnh cơ bản như :

dir Hiển thị các thư mục của máy login

get path/filename (cục bộ) path/filename (từ xa) Truyền tập tin đến máy từ xa

put path/filename (từ xa) path/filename(cục bộ) Nhận tập tin từ máy từ xa

exit Thoát khỏi ftp

+ Sao chép tập tin từ xa :

Sao chép tập tin giữa các máy.Có dạng

rcp [choïn löïa] nguoàn ñích

chọn lựa : -p sao chép cả thời gian và chủ quyền của tập tin nguồn

-r sao chép cả các thư mục con.

nguồn và đích có dạng hostname:pathname. hostname có thể bỏ qua ở máy cục bộ. Ví dụ :

```
$rcp test sunfibi:/usr/dung hoaëc
$rcp sunfibi:/usr/dung/test /usr/tan/thu
```

+ Thực hiện lệnh từ xa:

Nối kết với host và thực hiện lệnh. Dạng :

rsh host [choïn löïa] [leänh]

chọn lựa : -lusername nối kết host với username.

-n hướng nhập đến /dev/null

Ví dụ :

```
$rsh sunfibi -lroot cat /test > /thu
```

chép tập tin /test ở máy sunfibi sang máy cục bộ.

+ In tập tin :

In nội dung tập tin ra máy in từ xa. Dạng :

lpr hostname filename

hostname là tên máy có gắn máy in.Ví dụ :

```
$lpr sunfibi test
```

Ngoài ra trong PC/TCP còn có các chương trình chạy trong WINDOW mà cho phép sử dụng các giao thức TCP/IP ở chế độ đồ họa.

Để có thể thực hiện được các ừng dụng của XWINDOW phải sử dụng chương trình mô phỏng X terminal (eXceed 4). Các bước thực hiện như sau :

- Nhấn hai lần con chuột trên icon eXceed trong Window để khởi động eXceed.
- Nhấn hai lần con chuột trên icon Telnet có dialog sau:

Connect

OK

Host : sunfibi Cancel

Help

Đưa vào địa chỉ IP hay tên host chứa XWINDOW server.Sau khi nối kết thàng công đưa vào

```
login:
password:
```

Kế đó là khởi động X WINDOW client:

```
$[path/]xterm -ls -displays client1:0 &
```

2. Giao thức UUCP

Mạng đã được đưa vào UNIX năm 1975 với UUCP dưới dạng "điểm - điểm" giữa hai hệ thống UNIX kết nối qua đường dây điện thoại. UUCP được sử dụng rộng rãi, ngày nay nó cung cấp cả khả năng trên mạng cục bộ. Giao thức UUCP có thể sử dụng để truyền tập tin, thực hiện từ xa, gửi thư tín điện tử. Lệnh UUCP đã được nâng cao, các tập tin truyền bị ngắt sẽ được truyền lại từ chỗ truyền sai chứ không phải từ đầu tập tin. Phần cứng của UUCP có ba loại khác nhau :

- ✓ Direct links: đường dây trực tiếp tối đa là 50 feet tốc độ có thể đạt được 19200 baud
- ✓ Telephone line: Sử dụng Automatic Call Unit (ACU) tự động quay số theo yêu cầu của UUCP.
- ✓ Networking: sử dụng môi trường mạng cục bộ

2.1 Các thành phần của UUCP:.

user mail news
UUCP UUX

work file data file executes file
UUCICO

|
Kernel
|
|
Kernel
|
UUCI
CO

uucico coõng cuù noỏi keỏt caực maựy tớnh xa cho pheựp truyeàn taọp tin hay thửùc hieọn leọnh
uucp sao cheựp moọt taọp tin tửứ moọt maựy ủeỏn maựy khaực, UUCP taùo taọp tin dửứ lieọu vaứ taọp tin laứm vieọc, xeỏp haứng caực coõng vieọc ủeồ truyeàn vaứ goùi uucico daemon ồ noỏi keỏt vụựi maựy tửứ xa.
*uu*x taùo caực taọp tin laứm vieọc(C.xxx) , dửứ lieọu(D.xxx) vaứ thửùc hieọn(X.xxx) caàn thieỏt ủeồ thửùc hieọn leọnh treõn moọt maựy tửứ xa.Taọp tin laứm vieọc chửựa cuứng moọt thoõng tin nhử taọp tin laứm vieọc taùo bụỷi uucp vaứ uuto. Taọp tin thửùc hieọn chửựa doứng leọnh seừ ửỷụùc thửùc hieọn treõn maựy tửứ xa vaứ danh saựch caực taọp tin dửứ lieọu. Caực taọp tin dửứ lieọu laứ caực taọp tin ửỷụùc ủoứi hoỷi ủeồ thửùc hieọn leọnh.

2.2 Các lệnh UUCP

Gọi UNIX (cu) :

Cho phép nối kết với một hệ UNIX từ xa. Sau khi nối kết có thể thực hiện các lệnh từ xa. Lệnh có dạng như sau :

cu [-s tốc độ][-l đường] [-h][-o][-e] hostname

cu [-s tốc độ][-l đường] [-t][-h][-o[-e] telnum

trong đó :

-s tốc độ : định nghĩa tốc độ truyền

-l đường : chọn cổng

-h :truyền theo chế độ half-duplex

-d : Hiển thị từng bước thực hiện

-o : kiểm tra chẵn

-e: kiểm tra lẻ

-t : quản lí terminal ở chế độ tự động

hostname: tên hệ thống từ xa

telnum: số điên thoại của máy từ xa

Sau khi nối đướng thành công,cu sẽ truyền thông tin đọc từ thiết bị nhập chuẩn cho hệ xa, nhận các thông tin của máy xa chuyển ra xuất chuẩn. Có thể sử dụng các lệnh sau :

. kết thúc đối thoại

! trở về hệ cục bộ

! cmd thực hiện lệnh tại chỗ

$ cmd thực hiện lệnh tại chỗ và gửi cho máy từ xa

%cd chuyển thư mục trên máy cục bộ

%take tập tin từ xa tập tin cục bộ sao chép tập tin từ máy từ xa

%put tập tin cục bộ tập tin từ xa sao chép tập tin đến máy từ xa

%debug Chế độ gỡ rối

Ví dụ :

cu -s 2400 -l tty01 sunfibihn hoặc

cu -s 2400 -l tty01 014234567

connect

<enter>

Wellcome to Sun Solaris

login:

password:

sunfibihn$~%

sunfibihn$[sunfibi]%cd /usr/dung

***+uucp** : sao chép tập tin giữa các hệ unix.*

Dạng lệnh :

uucp [chọn lựa][nguồn!]tập tin nguồn [đích!] tập tin đích

nguồn và đích có thể là các hệ từ xa, tập tin đích có thể là thư mục. Chọn lựa :

-C chép các tập tin vào thư mục spool của máy ở xa

-c không chép các tập tin vào thư mục spool của máy ở xa

-d tạo các thư mục nếu các thư mục không tồn tại

-f không tạo các thư mục nếu các thư mục khôn tồn tại

-m Gửi mail cho user khi sao chép hoàn thành

-n user gửi mail cho user báo các tập tin đã được gửi

Ví dụ :

```
$uucp -m test sunfibihn!~/test
```

*+ **uux (unix to unix execution):** Cho phép thực hiện lệnh trên một hệ và gửi kết quả đến hệ khác.*

Dạng lệnh :

uux [chọn lựa][sys!]lệnh

chọn lựa :

8 - nhập chuẩn

-L khởi đông uucico

-C chép các tập tin cục bộ vào thư mục spool để truyền

-c không chép các tập tin vào thư mục spoolđể truyền

Ví dụ :

```
$uux sunfibihn!ls > sunfibí!test
```

*+ **uustat** : Thông báo trạng thái uucp.*

Dạng lệnh :

uustat [choïn löïa]

chọn lựa : -a Hiển thị tất cả trạng thái

-u user Hiển thị trạng thái user

-s sys Hiển thị trạng thái của hệ