# *Chương I : Giới thiệu*

## 1. LỊCH SỬ RA ĐỜI CỦA LINUX

Vào năm 1991 tại Phần Lan, Linus B. Torvalds lúc đó là sinh viên ở trường Đại học tổng hợp Hensinki đã dùng một máy tính cá nhân có trang bị bộ xử lí 386 để nghiên cứu cách làm việc của nó . Do hệ điều hành MS-DOS không khai thác đầy đủ các đặc tính của bộ xử lí 386 , Linus đã sử dụng một hệ điều hành thương mại khác là Minix . Hệ điều hành Minix là hệ điều hành Unix cỡ nhỏ .

Do đối mặt với các hạn chế của hệ điều hành này , Linus bắt đầu viết lại một số một số của phần mềm để thêm chức năng và các điểm đặc trưng . Sau đó , ông thông báo kết quả của mình miễn phí bằng Internet dưới tên gọi Linux - chữ viết tắt của Linus và Unix . Phiên bản đầu tiên của Linux là 0.01 được tung ra vào tháng 8/1991 .

Các phiên bản đầu tiên có rất nhiều hạn chế . Tuy nhiên , sự kiện các mã nguồn được truyền bá rộng rãi đã giúp phát triển hệ điều hành rất nhanh . Nhiều năm qua , số lượng các công ty khai thác đã không ngừng tăng lên . Ngày nay , Linux được phát triển bởi nhiều người rải rác khắp nơi trên thế giới .

World Wide Web đóng một vai trò quan trọng do nó hỗ trợ mở rộng nhanh hệ điều hành . Thực tế chúng ta có thể tưởng tượng rằng một nhà khai thác cài đặt Linux trên máy của mình , anh ta phát hiện lỗi , sữa chữa nó và gởi file nguồn đến Linus . Một vài ngày sau đó (đôi khi chỉ vài phút sau ) phần chính yếu được cải tiến có thể sẽ được truyền trên mạng .

Mặc dù năm phiên bản đầu tiên của Linux tương đối không ổn định , nhưng phiên bản đầiu tiên được tuyên bố là ổn định (1.0 ) đã được công bố vào khoảng tháng 3/1994 . Số phiên bản đi kèm với kernel có một ý nghĩa đặc trưng bởi vì nó liên quan đến chu kì phát triển . Thực tế , quá trình phát triển Linux diễn ra theo một chuỗi hai giai đoạn :

- Giai đoạn phát triển : ở đây kernel không có độ tin cậy cao và tiến trình là bổ sung chức năng cho nó , tối ưu hóa nó và thử nghiệm các ý tưởng mới . Giai đoạn này đem lại sự gia tăng số lượng các phiên bản đánh số lẻ , chẳng hạn như 1.1, 1.3 , vv.... Đây là thời điểm mà lượng công việc tối đa được thực hiện trên kernel .
- Giai đoạn ổn định : ở giai đoạn này , mục đích là tạo ra một kernel càng ổn định càng tốt . Trong trường hợp này , chỉ cho phép thực hiện các hiệu chỉnh , sửa đổi nhỏ . Số phiên bản của các kernel được gọi là ổn định là các số chẵn , chẳng hạn 1.0 , 1.2 và mới nhất là 2.2.

Ngày nay , Linux hoàn toàn là một hệ điều hành Unix . Nó ổn định và liên tục phát triển . Nó không chỉ có khả năng phát triển trên các thiết bị ngoại vi mới nhất trên thị trường ( bộ nhớ flash quang , đĩa quang ... ) mà hiệu năng của nó còn có thể so sánh với một số hệ điều hành Unix thương mại và thậm chí còn có một số điểm ưu việt hơn . Sau cùng, mặc dù Linux đã có một khoảng thời gian bị giới hạn trong môi trường các trường đại học , bây giờ nó đang được tiếp nhận ở các hãng công nghiệp . Do công suất và độ linh hoat của hệ điều hành này và tính miễn phí của nó mà hiện nay nó đang thu hút một số lượng các công ty ngày càng gia tăng .

## 2. CÁC CHỨC NĂNG CỦA LINUX

Hệ điều hành Linux có rất nhiều chức năng và chúng khai thác khả năng của các hệ Unix hiện đại theo các cách sau :

- **v** Đa xử lí , các bộ đa xử lí : có thể thực hiện nhiều chương trình đồng thời bất kể sử dụng một hay nhiều bộ xử lí .
- **v** Đa nền .
- **v** Cho phép nhiều người sử dụng : giống như tất cả các hệ Unix , Linux cho phép nhiều người sử dụng cùng làm việc trên một máy ở cùng thời điểm .
- **v** Hỗ trợ truyền thông giao xử lí ( Pipes , IPC , Sockets ) .
- **v** Quản lí các thông điệp điều khiển khác nhau .
- **v** Hệ thống quản lí thiết bị đầu cuối tuân thủ theo tiêu chuẩn POSIX . Linux cũng giả các thiết bị đầu cuối cũng như điều khiển quá trình .
- **v** Hỗ trợ một dải rộng các thiết bị ngoịa vi , chẳng hạn như các cạc âm thanh , giao diện đồ hoạ , mạng , giao diện hệ máy tính nhỏ ....
- **v** Buffer cache : vùng bộ nhớ được dành để làm vùng đệm cho các đầu vào và đầu ra từ các quá trình khác nhau .
- **v** Hệ thống quản lí bộ nhớ trang yêu cầu . Một trang sẽ không được nạp chừng nào nó không thực sự cần thiết ở bộ nhớ .
- **v** Các thư viện động và dùng chung : Các thư viện động chỉ được tải khi chún thật sự cần thiết và mã của chúng được dùng chung nếu nhiều ứng dụng đang dùng chúng
- **v** Các hệ thống file có thể quản lí tốt và đồng đều các phân hoạch file Linux được sử dụng bởi filesystem làm các phân hoạch có các định dạng khác ( MS-DOS , ISO9660, vv. .. ).
- **v** Thiết bị của TCP/IP và các giao thức mạng khác .

Tóm lại , Linux là một hệ Unix đầy đủ và mạnh . Nó có thể được ứng dụng dễ dàng . Ngoài ra sự sử dụng công cộng rộng rãi đang trợ giúp nó phát triển một cách nhanh chóng .

## 3. GIỚI THIỆU CHUNG VỀ LINUX:

Linux là hệ điều hành gần giống Unix , có thêí hoạt động độc lập với phần cứng , đa nhiệm , đa người dùng , bảo mật cao , tổ chức tập tin phân cấp , tốc độ cao , đáng tin cậy , có khả năng làm Server cho mạng Internet . Linux có điểm khá nổi bật đó là Tính ổn định .Thật khó mà làm cho Linux bị ngưng trệ và tê liệt ! Đã có nơi thử nghiệm nhiều hệ thống chạy Linux liên tục hằng năm trời mà không phải khởi động lại . Ngoài ra , Linux có thể chạy trên các máy tính thế hệ cũ vốn không thể chạy Windows 9x , thậm chí
cả những máy 486 vứt trong kho .

Ta có thể chia Linux thành 2 phần :

- Linux Kernel :( Hệ lõi ) Xác lập nhiều chương trình cấp thấp và tương tác trực tiếp với CPU . Hệ lõi cung cấp 2 chức năng cho hệ điều hành :

Nó cung cấp một hệ giao tiếp chung cho các phần cứng khác nhau từng cạc âm thanh với các chương trình người dùng .

Nó xác lập rào chắn giữa 2 chương trình khác nhau , nếu một chương trình bị hỏng chương trình kia không bị nó làm ảnh hưởng . Đây chính là ưu điểm nhất của Linux so với DOS và Windows .

**- Linux Shell : (Hệ vỏ ) Dùng để cung cấp cho người dùng một hệ giao tiếp được thi** hành dễ dàng , giống như COMMAND.COM của DOS . Đồng thời nó cũng có nhiệm vụ bảo vệ hạt nhân của hệ điều hành khỏi tác động trực tiếp của người sử dụng bởi HĐH Linux được thiết kế để các Shell này độc lập với các thành phần chính của HĐH .

Khi bạn sử dụng một chương trình Shell và gõ một lệnh nào đó , Shell sẽ thông dịch và thực hiện lệnh này ngay sau đó . Nó sẽ đưa ra các thông báo , báo lỗi hoặc các thao tác tương ứng . Ngoài ra , người sử dụng có thể lập trình với Shell . Các chương trình này được gọi là Script ( ngôn ngữ kịch bản ) và chúng được thông dịch , thực hiện bởi các Shell .

Hiện nay có nhiều loại Shell như : sh (Bourn Shell ), bash (Bourn Again Shell ), tcsh , csh , pdksh (Public Domain Shell ) , zsh , ash và mc . Nhưng phổ biến là : sh (Bourn Shell ), csh (C Shell ) , ksh (Korn Shell ) .

Để có thể truy cập vào hệ thống , trước hết bạn phải có quyền truy cập , biết được mật khẩu ( Hệ điều hành Unix luôn có sự phân biệt chữ hoa và chữ thường , chữ hoa chỉ dùng để viết tên riêng và mật khẩu ) . Khi bạn dã truy cập được vào hệ thống , các tiện ích và ứng dụng có sẵn sẽ xuất hiện theo một trong hai cách sau :

* Sử dụng hệ thống thực đơn : tạo cho người dùng sự tiện lợi khi sử dụng và đồng thời cung cấp cho người quản trị hệ thống một phạm vi bảo mật to lớn . Người quản trị hệ thống có thể sắp xếp lại hệ thống thực đơn của bạn để bạn có thể truy cập đến những ứng dụng và dịch vụ cần thiết .

* Môi trường Shell : Nó đòi hỏi bạn phải thành thạo với các lệnh và cấu trúc của Linux . Nếu bạn không có hệ thống thực đơn , môi trường Linux Shell sẽ được kích hoạt ngay sau khi bạn đăng nhập hệ thống . Để biết được bạn đang ở đâu và xem nội dung thư mục , bạn hãy gõ dòng lệnh :

**$pwd** { print working directory }

Để chuyển đổi thư mục làm việc bạn sử dụng lệnh cd cùng với tên thư mục mà bạn muốn chuyển tới . Chẳng hạn bạn muốn chuyển đến thư mục research thì bạn phải gõ : $cd research

Để thoát khỏi hệ thống (loging out ) bạn gõ vào dòng lệnh :

**$exit**
**hoặc $logout**
**hoặc nhấn tổ hợp phím Ctrl - D**

**Sau khi thoát thoát khỏi hệ thống trên màn hình xuất hiện dòng thông báo :**

**Login :_**

Để chấm dứt hoạt động của hệ thống ( phải chắc chắn rằng bạn đã thoát khỏi tất cả các ứng dụng và đóng tất cả các tệp đã sử dụng ) bạn gõ vào dòng lệnh :

**% shut down**

Sau khi các quá trình trên đã hoàn tất , trên màn hình của bạn sẽ xuất hiện dòng chữ sau : System is down .

***Chương II :***

# Tìm hiểu & sử dụng Linux shell

## I- Các lệnh cơ bản trong Linux Shell :

1. Phép kết gán :

**Phép kết gán cho phép ánh xạ một phím cụ thể theo một hành động . Ví dụ , khi** chúng ta gõ phím ^A tại dấu nhắc hệ vỏ con trỏ sẽ nhảy đến đầu dòng. Để thuận lợi , nhiều hệ vỏ gán sẵn cho người dùng các phím gán sau :

| | |
|---|---|
| ^A | Dời con trỏ đến đầu dòng |
| ^C | Gửi một SIGINTR ( ngắt ) ra hệ vỏ |
| ^D | Hiển thị danh sách các tập tin |
| ^E | Dời đến cuối dòng |
| ^K | Triệt từ con trỏ đến cuối dòng |
| ^N | Dời xuống trong danh sách |
| ^P | Dời lên trong danh sách |
| ^U | Triệt nguyên cả dòng |
| ^I | Hoàn tất tập tin |

Trong trường hợp có nhiều tập tin bắt đầiu bằng các kí tự giống nhau , hệ vỏ sẽ đưa vào nhiều kí tự rồi phát tiếng kêu beep thông báo cho chúng ta biết có các khả năng khác . Ví dụ : >rmdir direct (^I)

>rmdir directory_I_want_to_

Đến đây chúng ta có thể gõ lệnh K hoặc D , ^D để có một danh sách các tập tin trong thư mục hiện hành bắt đầiu bằng các kí tự đó .

Đây chính là tính năng rất tiện dụng đối với các tập tin hay thư mục có tên dài .

2. Chức năng của một số kí tự :

| Kí tự | Chức năng |
|---|---|
| * ? [ ] | Kí tự đại diện hay theo mẫu . |
| & | Chạy ứng dụng ở chế độ nền , trả lại dấu nhắc hệ thống cho các tác vụ khác . |
| ; | Dấu phân cách nhiều lệnh trên một dòng lệnh . |
| \ | Tắt tác dụng của những kí tự đặc biệt như *, ?, [ , ], &, ; , >, <, \| |
| ',,,' | Khi tham số là nhóm từ (có khoảng trống ) . |
| "... " | Khi tham số có khoảng trống và các kí tự đặc biệt ngoại trừ kí tự $ và ' |
| > | 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 . |
| '...' | Dấu huyền dữ liệu xuất của một lệnh làm tham số . |
| $ | Sử dụng biến môi trường . |

3. Cấu trúc thư mục :

Linux tổ chức thư mục và tập tin theo cấu trúc cây giống như DOS và Windows . Về đường dẫn , ta có thêí dùng đường dẫn tương đối hoặc đường dẫn tuyệt đối như DOS . Nhưng thay vì dùng dấu " \ " để phân cách các cấp thư mục như trong DOS thì Linux lại dùng dấu " / " .

/etc Cấu hình hệ thống cục bộ theo 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 tạp tin chuẩn của C

/var/log Chứa các tập tin lưu trữ thông tin làm việc hiện hành của người dùng

/home Chứa các thư mục con của các User

/usr/local Các chương trình bổ sung không thuộc thành phần của một hệ thống . Thông thường ./usr/local có các thư mục con như sau :

/usr/local/bin /usr/local/lib

/usr/local/man /usr/local/include

/usr/src Vị trí của mã nguồn ( kể cả mã nguồn của HĐH Linux )

/usr/lib Chứa các tập tin thư viện của các chương trình người dùng

Trong Linux : " . " cho biết đó là thư mục hiện hành , " .. " chỉ thư mục cao hơn một cấp (thư mục mẹ ) . Nếu đường dẫn bắt đầu bằng "/ " thì hệ thống xem đó như là một tên đường dẫn đầy đủ (tuyệt đối ) . Đường dẫn bắt đầu bằng " ~ " là một đường dẫn tương đối. Những kí hiệu này có thể được sử dụng cùng với nhau . Ví dụ :

" ~/.." có nghĩa là thư mục mẹ của thư mục riêng.
"../.." để chỉ một thư mục cao hơn thư mục mẹ .
/home/user01# more ~/document/baocao
tương đương với
/home/user01# more home/user01/document/baocao

4.Cú pháp dòng lệnh :

Các lệnh trong Linux thường bắt đầu bằng tên lệnh (command) , sau đó làcờ (flag) và đối số (argument ) :

Command [ flag ] argument1 argument2 ....

Các cờ ( còn gọi là lựa chọn (option) ) trong DOS thường đứng sau " / ", trong khi
Linux lại dùng " - ". Ví dụ: trong DOS bạn gõ " dir /a /o:d " thì trong Linux bạn gõ là "ls - lac *" .*

Hầu hết các trường hợp nhiều đối 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 " .

*Các đối số phải cách nhau bởi dấu cách (space) hoặc Tab . Nếu trong đối số có* khoảng cách thì phải đặt nó trong cặp ngoặc kép ( xem thêm mục 5.11 )

5. Một số lệnh thường dùng:

**5.1. Tạo thư mục :**

Cú pháp : mkdir <dir1> <dir2> ... <dirN>
Trong đó <dir1> <dir2> ... <dirN> là các thư mục cần tạo .
Ví dụ : mkdir thuchanh tạo thư mục thuchanh
*mkdir thuchanh/baitap1 tạo thư mục baitap1 là thư mục con* của thư mục thuchanh

**5.2. Chuyển thư mục :**

Cú pháp : cd <directory>
*Dùng " . " để chuyển đến thư mục hiện hành , " .." để chuyển đến thư mục cha .*
*Ví dụ : cd /usr/local/bin*

**5.3. Xem thư mục hiện hành :**

Cú pháp : pwd

**5.4. Xoá thư mục rỗng :**

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

**5.5. Xoá tập tin hoặc thư mục :**

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

**5.6. Hiển thị thông tin về tập tin và thư mục :**

Cú pháp : ls <danh sách file> | <Danh sách thư mục> <tham số>
<tham số> : - F : dùng để hiển thị vài thông tin về tập tin . Sau tên file , hiển thị dấu sao (*) nếu là file thi hành , dấu (@) nếu là file liên kết , dấu ( / ) nếu là thư mục con , dấu chấm ( . ) nếu là file ẩn .

- l : (long) cho phép liệt kê kích thước tập tin , người tạo ra , các quyền người sử dụng ...

Để liệt kê nội dung của các thư mục con bạn có thể sử dụng cờ -R

**Để liệt kê các file ẩn bạn sử dụng cờ -a**

ví dụ : $ ls -lF

```
 total 75
drwxrwxr-x   2 user 12     user 12     1024 Apr 7 09:41 baitap/
drwxrwxr-x   2 user 12     user 12     1024 Apr 7 09:41 doc/
-rwxrwxr-x   1 user 12     user 12     71 Mar 31 10:39 hello*
-rw-rw-r--   1 user 12     user 12     126 Apr 7 09:26 baitho.txt
-rw-rw-r--   1 user 12     user 12     70 Apr 7 08:26 hello.c
$
```

**5.7. Di chuyển tập tin , thư mục :**

Cú pháp : mv <danh sách tập tin hoặc thư mục > <đích>

*<đích > : là tập tin hay thư mục*

Lệnh này có thể dùng để đổi tên tập tin hoặc thư mục (tương tự lệnh Rename của DOS).

**5.8. Sao chép tập tin , thư mục :**

Cú pháp : cp <nguồn > <đích >

*Lệnh này không tự động sao chép các thư mục con trừ khi bạn sử dụng cờ -R*

**5.9. Xoá các tập tin hoặc thư mục :**

Cú pháp : rm <file> |< thư mục>

*Nếu bạn dùng lệnh này kèm với cờ -i thì trước khi định xóa một file , máy sẽ hỏi* lại bạn có thực sự muốn xoá hay không .

Chú ý , lệnh rm * sẽ xoá mọi file trong thư mục hiện tại .

**5.10. Hiển thị nội dung các tập tin : Mỗi lần chỉ hiển thị đầy màn hình (24 dòng ) ,** muốn xem trang tiếp theo thì nhấn phím spacebar .

Cú pháp : more [-n] <danh sách các tập tin >

[-n] : chỉ định số dòng mỗi lần hiển thị là n dòng .

Ví dụ : $more baitho.txt //hiển thị nội dung tập tin baitho.txt

$more mbox // xem tất cả thư lưu trong hộp thư

$more -4 grocery.txt

\documentstyle[12pt] {article}

\input{psfig}

\input{/home/a_s/pehng/Teach/Mat466/std.top}

--More--(0%)

Dòng thông báo --More-- có nghĩa là bạn nhấn phím spacebar để xem phần tiếp theo, nhấn phím q nếu muốn kết thúc .

Nếu bạn muốn bỏ qua n dòng đầu tiên thì bạn sử dụng cờ +n

Ví dụ : $more +40 grocery.txt

**5.11. Tìm kiếm một chuỗi kí tự :**

*Cú pháp : grep <chuỗi cần tìm> <tên file>*

*Nếu tìm thấy thì trả về các dòng có chứa chuỗi cần tìm .*

*Ví dụ : grep New York // tìm từ " New" trong file " York"*

grep "New York" // tìm chuỗi "New York" trong đầu vào chuẩn ( standard input)

Chuỗi kí tự hay biểu thức cần tìm có thể kết hợp với các kí tự đặc biệt sau :

| Kí tự | Tác dụng |
|---|---|
| . | Thay thế cho một kí tự .Ví dụ : b.d sẽ tương xứng với bod and bad |
| *** và [ ]** | **Xem mục 6.** |
| **/** | **Tắt tác dụng của các kí tự đặc biệt .Ví dụ : /* sẽ tìm dấu * , // sẽ tìm** dấu / |
| **^** | **Ví dụ : ^704 sẽ tìm có mã vùng (bắt đầu ) là 704** |
| **{ }** | **Ví dụ : g\{3,4} sẽ tìm bất cứ dòng nào có chứa ggg hoặc gggg** |

Ví dụ : lệnh grep '408.[0-9]\{3\}.[0-9]\{4\}' sẽ tìm mọi số điện thoại có mã vùng là 408; chẳng hạn như : 408-555-1212, 408.555.1212, *408.234.7890 ....*

**5.12. Tìm kiếm một tập tin :**

Cú pháp :

*- Tìm theo tên : $find đườngdẫn -name têntậptin -print*

- Tìm theo số i- node (i-num) của tập tin :

$find đườngdẫn -inum number -print

- Tìm theo tên người sở hữu :

$find đườngdẫn -user username -print

Để tránh các thông báo lỗi đưa ra màn hình , ta có thể đổi hướng đầu ra chuẩn (*standard error ) tới một tập tin rỗng ( / dev/null) :*

$find / -name têntậptin -print 2>/dev/null

Ví dụ :
```
$pwd
/home/user01
$find / -name ttyc2d1 -print 2>/dev/nul
/dev/ttyc2d1
```

**5.13. Hiển thị hướng dẫn sử dụng lệnh:**

Cú pháp : man <command>

*<command> là tên của một lệnh hoặc của một tài nguyên cần gọi giúp đỡ .*

Ví dụ : man ls //đưa ra giúp đỡ của lệnh ls

Bạn có thể dùng cú pháp sau để hiển thị nội dung hướng dẫn sử dụng lệnh :

Tên lệnh --help

**5.14. Nối các tập tin :**

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

Ví dụ :
```
$cat baitho.txt vanban.doc //hiển thị nội dung cả 2 tập tin
$cat baitho.txt vanban.doc > tonghop.doc //kết nối nội dung
```
cả 2 tập tin vào tập tin tonghop.doc

**5.15.Phản hồi lại các tham số đưa vào:**

Cú pháp : echo <arg1> <arg2> ... <argN>

Ví dụ :
```
$echo "hello"
hello
```

**5.16. Hiển thị tên máy tính đang làm việc : Linux cất thông tin về tên máy trong tập** tin /etc/hots

Cú pháp: hostname

*Ví dụ :* *$hostname*
Linux.edu

**5.17. Nén một tập tin : Tên tập tin đã nén giống như tên ban đầu và kèm theo đuôi .gz**

Cú pháp : gzip <filename>
*Vi dụ :* *gzip vban.txt* *//tên tập tin đã nén là:vban.txt.gz*

**5.18. Giải nén một tập tin :**

Cú pháp: gunzip <filename>

**5.19. Gọi hướng dẫn :**

**Cú pháp : man < command name >**
*Lệnh này hiển thị tài liệu Linux hoặc các trang giúp đỡ ( man pages) về lệnh* <command name>

**5.20. Bí danh của lệnh: (alias)**

Cú pháp : alias < tên lệnh mới > = < tên lệnh cũ >
*Lệnh này cho phép bạn đặt một bí danh cho một lệnh đã có , kể cả những lệnh* phức tạp .
Ví dụ : alias help=man // bây giờ bạn có thể dùng lệnh help cp hoặc *man cp để hiển thị trang giúp đỡ về lệnh cp*
*alias timedir="ls -art"*
*Dấu ngoặc kép là cần thiết bởi nếu không có nó thì shell sẽ kết thúc lệnh khi gặp* dấu cách (space bar) và khi đó cờ -art sẽ mất tác dụng .
Chú ý : việc đặt bí danh từ dòng lệnh chỉ có hiệu quả trong phiên làm việc hiện hành . Để có bí danh được kích hoạt mỗi khi đăng nhập (log on ) , hãy định nghĩa bí danh trong file .profile nếu bạn sử dụng Born shell , trong file *.login nếu bạn sử dụng C shell.*

**5.21. clear : Xoá màn hình .**

**5.22. date : Hiển thị ngày tháng hiện hành của hệ thống .**

**5.23. time : Hiển thị thời gian hiện hành của hệ thống .**

**5.24. useradd : Thêm người dùng vào mạng .**

**5.25. passwd : Đặt lại password người sử dụng .**

( Tiện ích mc trên Linux có giao diện làm việc giống như trình NC (Norton Commander) . Để khởi động mc , từ dấu nhắc lệnh gõ :
$mc

6. Các kí tự đại diện dùng trong câu lệnh :

**6.1. Dấu sao (*) : Đại diện cho một nhóm kí tự bất kì .**

**Ví dụ : cat sales* > allsales // kết nối mọi file có tên bắt đầu là sales vào** trong file có tên là allsales
*ls *rep** *// hiển thị mọi file mà tên của nó có chứa "rep"*
ls .*rep* // hiển thị mọi file ẩn mà tên của nó có chứa "rep"

**6.2. Dấu chấm hỏi (?) : Đại diện cho một kí tự bất kì .**

**Ví dụ : lp ??x // in ra các file mà tên có 3 kí tự , bắt đầu bằng 2 kí tự bất** kì, còn kí tự cuối là x

**6.3. Dấu ngoặc vuông ([ ]) : Chỉ phạm vi các kí tự được đại diện .**

**Ví dụ : ls job[123] // chỉ hiển thị các file :job1, job2, job3**
ls [A-Z]* // chỉ hiển thị các file có tên bắt dầu bằng chữ in hoa .

ls [A-Z,a-z] // chỉ hiển thị các file có tên bắt dầu bằng chữ cái.

7.Kết nối các tiến trình với các ống dẫn (pipes) :

**Việc kết nối liên tiếp các lệnh bằng việc sử dụng ống dẫn ( kí hiệu là | ) làm cho** đầu ra (output ) của chương trình hay lệnh ở phía trái của ống dẫn là đầu vào (input ) của chương trình hay lệnh ở phía phải của ống .

Ví dụ : sort allsales | lp // để sort (lựa chọn, sắp xếp , phân loại ) file tên là allsales rồi in nó .

*cat sales* | sort | lp // để in ra danh sách dữ liệu đã sắp xếp (sort) trong các file có tên bắt đầu là sales .*

**8. Định hướng lại đầu vào và đầu ra : (Redirecting Input and Output )**

**Sử dụng dấu nhỏ hơn (< ) để định hướng lại đầu vào vào trong một chương trình** hay một lệnh từ một file thay vì một thiết bị đầu cuối (terminal:bàn phím ...) . Giả sử bạn muốn gởi một file tên là info bằng e-mail đến một ai đó có địa chỉ là sarah . Thay vì bạn phải gõ lại nội dung của file cho lệnh *mail , bạn có thể sử dụng file info như là đầu vào của lệnh mail bằng cách nhập* vào dòng lệnh sau :

m*ail sarah < info*

Sử dụng dấu lớn hơn (>) để định hướng lại đầu ra của một chương trình hay một lệnh đến một file thay vì đến màn hình (terminal) (đầu ra được đặt trong một file) . Lệnh date hiển thị thời gian và ngày tháng hiện tại ra màn hình . Nếu bạn muốn lưu trữ thời gian và ngày tháng hiện tại vào một file tên là now thì bạn nhập dòng lệnh sau :

*date > now*

Chú ý : nếu tên file bên phía phải của dấu > đã tồn tại thì nó sẽ ghi đè. Hãy cẩn thận đừng để mất những thông tin hữu ích bởi cách này .

Nếu bạ muốn bổ sung hoặc kết nối thông tin vào một file đang tồn tại , hãy dùng hai dấu lớn hơn (>>) .

Ví dụ : date >> report // để bổ sung ngày tháng hiện tại vào file tên là report

sort < sales >> salesreport // dữ liệu trong file sales vừa được đưa vào lệnh sort vừa được bổ sung vào file salesreport .

**9. Biến môi trường của Shell :**

**Môi trường của Shell chứa một số biến được định nghĩa trước . Lệnh set cho phép** liệt kê danh sách các biến của môi trường .

Dưới đây là danh sách các biến môi trường thường có :

| | |
|---|---|
| HOME | chứa tên thư mục tiếp nhận |
| LOGNAME | tên người sử dụng |
| PATH | tên đường dẫn cho các lệnh |
| PS1 | dấu nhắc 1 |
| PS2 | dấu nhắc 2 |
| TERM | kiểu của thiết bị cuối(terminal: bàn phím hoặc màn hình) |
| FCEDIT EDITOR | chương trình soạn thảo nhật kí |
| PRID | số của tiến trình cha của Shell |
| PWD | thư mục hiện hành |
| SHELL | tên Shell đang dùng |

RANDOM số ngẫu nhiên
SECONDS thời gian làm việc tính theo giây

**10. Biến thay thế :**

**Các biến Shell được lưu trữ như một chuỗi . Khi 2 biến được đặt cùng nhau , các** chuỗi riêng của chúng được nối lại ( các biến được dùng kèm với dấu $ ở trước ).

Ví dụ : giả sử ta có 2 biến X=hello , Y=world
*$echo $X$Y //cho kết quả là helloworld*
$echo $X $Y // cũng cho kết quả là helloworld
*$echo $XY // cho kết quả là helloY*

**11. Sự thay thế kết quả của lệnh :**

**Cú pháp : command1 parameter `command2`**

Lệnh 2 (command2 ) được thi hành trước và kết quả của nó được xem như là một tham số của lệnh 1 (command1 ).

Chú ý : dấu " ` " là dấu nháy ngược (backquote ) , phím backquote nằm trên phím Tab trên bàn phím .

Ví dụ : $echo Today\`s date and time are `date` cho kết quả là :
Today`s date and time are Mon May 18 14:35:09 EST 1994

**12. Tìm hiểu Nhóm lệnh (Command Group) và Shell dưới (Subshell) :**

**Nếu bạn muốn đặt một hay nhiều lệnh trên một dòng lệnh trước khi nhấn Enter,** bạn có thể sử dụng cú pháp dưới đây . Shell sẽ thi hành tuần tự các lệnh .

command1; command2; command3

Ví dụ : $clear;ls // xoá màn hình và hiển thị thư mục

**Nhóm lệnh : Nếu bạn muốn định hướng lại đầu vào và đầu ra các lệnh như một nhóm ,** bạn có thể tạo một nhóm lệnh . Một nhóm lệnh được định nghĩa như là một số lệnh được bao bởi dấu ngoặc móc ({ }) . Lệnh sau định hướng lại đầu ra của cả 2 lệnh đến file tên là output-file :

{command-1;command-2} > output-file

Đầu ra của một nhóm lệnh có thể được "đặt ống "( can be piped) .Ví dụ :

{command-1;command-2} | command-3

Subshell : Khi bạn chạy một chuỗi các lệnh như một nhóm lệnh , các lệnh này chạy trong Shell hiện tại . Nếu một trong những lệnh đó thay đổi môi trường hoặc thay đổi thư mục thì khi nhóm lệnh chạy xong , những thay đổi đó sẽ bị ảnh hưởng . Để tránh vấn đề này , nên chạy nhóm lệnh trong subshell .

Subshell là một bắt chước (clone) của Shell hiện tại , nhưng bởi vì các tiến trình con không thể thay đổi môi trường của tiến trình cha , mọi lệnh chạy trong một *subshell không ảnh hưởng đến môi trường khi nhóm lệnh kết thúc .Để chạy* một nhóm lệnh trong subshell ,ta thay thế ngoặc móc bằng ngoặc đơn .Ví dụ ở phần trên trở thành :

(command-1;command-2) | command-3

Chỉ command-3 chạy trong shell hiện hành , còn đầu ra của subshell được đặt vào ống để thành đầu vào chuẩn của command-3.

**13. Soạn thảo lệnh :**

Soạn thảo lệnh có nghĩa là sau khi bạn đánh vào một lệnh và trước khi nhấn Enter

Bạn có thể soạn hoặc thay đổi các phần của lệnh mà không phải đánh lại phần lớn lệnh . Để biên soạn lệnh , bạn nhấn phím <Esc> để chuyển sang chế độ

soạn thảo và sau đó sử dụng bất kì lệnh di chuyển dòng nào của trình soạn thảo v*i* để sửa đổi lệnh . Bạn có thể sử dụng phím <backspace> , sử dụng các lệnh khác của vi như x để xoá một kí tự , r để thay thế một kí tự .....

**14. Xem lại lệnh và thi hành lại lệnh đã thực hiện : (Viewing Command History)**

**Đặc trưng này cho phép bạn xem lại các lệnh đã nhập vào trước đó và gọi lại** chúng . Khi bạn kết hợp đặc trưng này với việc soạn thảo lệnh , bạn có thể dễ dàng sửa lỗi trong những lệnh phức tạp và giải quyết hiệu quả với một số công việc lặp lại .

Lịch sử lệnh (the history command) hiển thị danh sách các lệnh cũ mà shell đã lưu lại . Các lệnh được đánh số . Chẳng hạn , để thi hành lệnh 10 , bạn hãy nhập vào :! 10 . Bạn có thể dùng các phím mũi tên để gọi lại các lệnh trước đó .

**15. Làm việc với kịch bản Shell (Shell Script) :**

Shell script là một tập hợp của một hoặc nhiều lệnh shell trong một file . Để thi hành các lệnh đó , bạn đánh vào tên của file . Shell scipt đem lại những thận lợi sau:

- bạn không phải đánh lại liên tiếp các lệnh .
- bạn xác định các bước để hoàn thành mục đích một lần .
- bạn đơn giản hoá các thao tác cho chính bạn và cho người khác .

*Các bước tạo một Shell script :*

1. Sử dụng một trình soạn thảo văn bản , chẳng hạn vi hoặc emacs , hãy đặt các lệnh shell vào trong một file văn bản hoặc file ASCII rồi dặt cho file đó một tên .
2. *Để tạo một file thi hành , ta dùng lệnh sau : chmod +x <tên file>*
3. Thử lệnh bằng cách gõ tên lệnh và nhấn Enter .

Bạn có thể kiểm tra một shell script và thấy mọi bước nó thực hiện bằng cách nhập vào dòng lệnh sau :

*sh -x script-name*

*Trong cú pháp trên , script-name là tên của script mà bạn đang xem xét . Lệnh sh -x rất hữu ích khi bạn đang thử dò tìm lỗi của một script .*

## II- Phân quyền sử dụng - Bảo vệ tập tin, thư mục :

**1. Phân quyền sử dụng :**

Linux đưa ra 3 loại phân quyền sử dụng đối với người sử dụng :

- Đọc (chỉ cho phép đọc ); Ghi (cho phép thêm hoặc huỷ) ; Thực hiện (thực hiện
các chương trình ứng dụng hoặc các tệp Shell script )

Linux cho phép bạn kiểm soát 3 loại quyền cơ bản này với 3 loại người sử dụng :

- Chủ sở hữu : là người đầu tiên tạo ra tệp này .
- Nhóm người dùng : những người dùng Linux có thể tham gia vào một nhóm làm việc nào đó hoặc không . Những người dùng trong cùng một nhóm cùng tham gia vào một dự án đó .
-Những loại người dùng khác : là những người không thuộc 2 loại trên .

2. Mô tả người sử dụng :

**Một người sử dụng được mô tả bằng 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 (Home directory)
- [ Tên chương trình cho chạy lúc bắt đầu phiên làm việc ]

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

3. Mô tả nhóm người sử dụng :

**Một nhóm người sử dụng là tập hợp của một số người sử dụng có thể dùng chung** các tập tin của nhau . Một nhóm người sử dụng được mô tả bằng các thông tin sau :

- groupname tên của nhóm
- password [mật khẩu]
- gid số của nhóm
- [danh sách những người khách ]

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

4. Bảo vệ các tập tin và thư mục :

**4.1. Các quyền thâm nhập tập tin :**

**Khi tập tin được tạo lập , các thông tin sau đây đồng thời được ghi lại :**

- gid của nhóm người tạo tập tin
- **uid 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 (quyền đọc ) , w ( quyền ghi ), x ( quyền thực thi ) , suid( set user-id) , sgid (set group-id)

đối với thư mục : r : quyền đọc nội dung thư mục
w: quyền tạo và xoá các tập tin trong thư mục
x : quyền qua lại (crossing) thư mục

**4.2. Lệnh ls -l hoặc ls -lF**

Lệnh này liệt kê danh sách các tập tin và các thuộc tính của chúng trong một thư
mục , qua đó ta có thể phát hiện loại tập tin , cách bảo vệ , người sở hữu và kích thước của chúng .

Ví dụ : $ls -l /bin

- rwxrwwxr -x 1 bin bin 16336 Mar 8 1998 cat
- rwxrwwxr -x 3 root bin 16124 Mar 8 1998 cp

trong đó : cột 1: loại tập tin và quyền thâm nhập . Dấu trừ '-' ở đầu có nghĩa là tập tin thường . Dấu trừ '-' trong dãy bít có nghĩa không có quyền truy cập tươn ứng bit đó . Để tiết kiệm chỗ người ta đặt bit n vào cùng một nơi với bit x và kí hiệu :

- s nếu x tồn tại (bit s: set uid hoặc set gid khi chạy tập tin )

- S nếu x không tồn tại

cột 2 : số liên kết ( link number)
cột 3 : tên người sở hữu (owner)
cột 4 : tên nhóm sở hữu
cột 5 : kích thước tập tin
cột 6,7,8 : ngày sửa đổi gần nhất
cột 9 : tên tập tin

4.3. Thay đổi quyền thâm nhập , lệnh chmod:

Lệnh chmod cho phép thay đổi quyền thâm nhập các tập tin và các danh mục . Có thể chạy lệnh theo 2 cách :

- Cho thông số tuyệt đối :

*Cú pháp :* *chmod mode filename*

Trong đó : thông số mode là một cơ số 8 (octal )

| rwx | r - x | r - - |
|---|---|---|
| 111 | 1 0 1 | 1 0 0 |
| 7 | 5 | 4 |

*chmod 754 tên tập tin*

- Dùng các kí hiệu tượng trưng :

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

Câu lệnh chmod được dùng để thiết lập mức đặc quyền của tập tin . Chỉ những người sở hữu tập tin này mới có thể thay đổi được mức đặc quyền đối với tập tin này . Trong đó :

u có nghĩa user
g có nghĩa group
o other
a all

Các toán tử : + thêm quyền
- bớt quyền
= gán giá trị khác

Quyền : r Cho phép đọc ghi
w Cho phép ghi
x Quyền thực thi tập tin
s Thiết lập suid hoặc guid

Ví dụ : $chmod g -w , o +r baitho.doc

Nghĩa là :+ Bớt quyền ghi tập tin (w) baitho.doc cho nhóm (g)
+ Thêm quyền đọc tập tin (r ) baitho.doc cho các người sử dụng khác $chmod a+r baocao.txt // tất cả người sử dụng có thể đọc được

$chmod +r baocao.txt // lệnh này tương đương lệnh trên
*$chmod og-x baocao.txt// không cho thực thi*
*$chmod u+rwx baocao.txt// cho phép người sở hữu đọc, viết và thực thi*
*$chmod o-rwr baocao.txt // không cho truy nhập tập tin*
*$chmod 777 * // đặt tất cả các quyền cho tất cả các đối* tượng sử dụng đối với toàn bộ tập tin trong thư mục hiện hành

**4.4. Thay đổi người hoặc nhóm sở hữu tập tin :**

**- Lệnh chown cho phép thay đổi người sở hữu .**
- Lệnh chgrp cho phép thay đổi nhóm sở hữu .

Ví dụ : $echo Hello > file1
*$chmod 700 file1*
*$ls -l file1*
*- rwx - - - - - - 1 user1 stagiar 6 Apr 5 14:06 file1*
*$cat file1*
*$chgrp animator file1*
*$ls -l file1*
*$cat file1*

***Chương III :***

# LẬP TRÌNH VỚI SHELL

Để lập trình với Shell , bạn phải biết về các biến và các cấu trúc điều khiển . Biến là một đối tượng mà tại bất cứ thời điểm nào bạn cũng có thể gán lại một giá trị khác cho chúng . Cấu trúc điều khiển cho phép bạn điều khiển sơ đồ thực thi của một script . Có 2 kiểu cấu trúc điều khiển : cấu trúc rẽ nhánh ( như cấu trúc if ... then ...else fi và cấu trúc case ) và cấu trúc lặp ( như vòng lặp for hoặc while ) .

**Sử dụng echo : Bạn có thể sử dụng lệnh echo để hiển thị những gì xảy ra trong một scipt**

. Lệnh echo hiển thị các đối số của nó ra màn hình . Bạn cũng có thể định hướng lại kết quả của echo đến một file .

Ví dụ : echo "Please stand by ..." sẽ hiển thị ra màn hình dòng chữ sau :

*Please stand by ...*

*Dòng lệnh sau sẽ đặt Please stand by ... vào trong file có tên là messg :*

*Echo " Please stand by ..." >messg*

**Sử dụng chú thích : Dấu "#" là bắt đầu một chú thích của Shell .Shell sẽ bỏ qua mọi kí** tự nằm sau dấu "#"cho đến cuối dòng .

**Tình trạng thoát : (Exit Status) Khi một lệnh Shell thi hành , nó có thể thành công** hoặc không . Shell luôn trả về tình trạng kết thúc của một lệnh , chương trình hoặc Shell script . Giá trị trả về được gọi là tình trạng thoát (exit

status) của lệnh và được đại diện bởi biến $? . Nếu $? có giá trị là 0 , lệnh thành công, còn ngược lại là không thành công . Bạn sẽ thấy giá trị của $? nếu nhập vào lệnh sau :

*grep "American Terms" custemers*
*echo $?*

## I. Sử dụng biến trong chương trình Shell :

Để sử dụng các biến , bạn phải biết cách đặt một giá trị vào một biến và cách truy cập giá trị cất trong biến đó . Có 4 cách đặt một giá trị vào một biến :

- **ü** Dùng phép gán trực tiếp
- **ü** Sử dụng lệnh read
- **ü** Sử dụng các tham số của dòng lệnh
- **ü** Thay thế đầu ra của một lệnh

### 1. Sử dụng phép gán trực tiếp :

*myemail=edsga@crty.com*

*Biểu thức trên đặy vào biến myemail giá trị là edsga@crty.com . Lưu ý rằng hai* phía của dấu bằng "=" không có khoảng trống . Nếu giá trị của biến có khoảng trống thì phải đặt nó trong dấu ngoặc kép . Ví dụ :

*myoffice="Room 21 , Suit C"*

Shell sẽ tìm lại được giá trị của biến bất cứ khi nào nó thấy dấu "$" ở trước tên biến . Ví dụ : echo "My e-mail ađress is $mymail"

*Giả sử bạn cần copy file tên là current vào thư mục /corporate/ìno/public/sales ,* bạn nhập vào lệnh sau :

cp current /corporate/ìno/public/sales

Bạn có thể gán tên thư mục dài đó vào một biến như lệnh sau :

corpsales=/corporate/ìno/public/sales

*rồi copy file current vào thư mục đó bằng lệnh sau :*

*cp current $corpsales*

### 2. Sử dụng lệnh read :

**Lệnh read tạm ngừng script và đợi người sử dụng nhập vào từ bàn phím rồi gán** cho tên biến . Khi phím Enter được ấn thí script được tiếp tục . Nếu nhấn " ^d" trong khi lệnh read đang đợi nhập thì script được kết thúc . Ví dụ sau nhắc người sử dụng nhập vào tên file được copy :

corpsales=/corporate/ìno/public/sales
*echo "Enter name of file to copy "*
*read filename*
*cp $filename $corpsales*

### 3. Sử dụng các tham số dòng lệnh :

**Các tham số dòng lệnh được phân cách bởi ít nhất một kí tự trống (Nếu trong đối** số có khoảng trống thì phải đặt nó trong cặp ngoặc kép ) . Tên lệnh và các đối số được gán cho các biến là $0, $1 ,....,$9 . Tên lệnh là $0 , đối số thứ nhất của lệnh là $1 , đối số thứ 2 của lệnh là $2 , ...., cứ thế cho đến $9 .

Ngoài ra : $# để chỉ số các tham số , $" để chỉ tất cả các tham số , $$ để lấy PID của Shell scrpit .

Ta xem xét Shell script ví dụ sau có tên là shovars (show variables ) :

```
# Name :    shovars
echo   $0
echo   $2  $4
echo   $3
```

Giả sử ta nhập vào dòng lệnh sau :

*shovars  -s  hello  "look at me "  bart*

*kết quả là :*

*shovars*
*hello  bart*
*look at me*

*Shell script sau sẽ xóa một file nhưng trước đó file đã được copy vào thư mục* /tmp :

```
# Name :       safrm
cp  $1  /tmp
rm  $1
```

Bạn có thể đại diện mọi tham số trên dòng lệnh bằng $*

### 4. Thay thế đầiu ra của một lệnh :

**Bạn có thể gán cho một biến kết quả một lệnh được thi hành .**

Ví dụ :  cwd='pwd'  // lưu tên thư mục hiện hành vào biến cwd

Chú ý là pwd đặt trong cặp dấu nháy ngược " ` " (dấu này nằm ngay phía trên phím Tab) chứ không phải dấu nháy đơn .

Ta xem một script đổi tên file sau :

```
# Tên :       stamp
#  Mục đích :đổi tên file bằng cách bổ sung vào tên file ngày hiện tại
t d=`+%m%d%y`
mv  $1  $1.$td
```

Nếu hôm nay là ngày Sep 02 , 2001 và thực hiện lệnh stamp  myfile thì myfile bị đổi thành myfile.090201.

## II . Các cấu trúc điều khiển :

### 1. Cấu trúc case :

**Cú pháp :**

```
case  word  in
     pattern)  statement(s);;
     pattern)  statement(s);;
      .....
esac
```

Nếu đối số word tương ứng với đối số pattern thì các lệnh statement phía sau nó sẽ được thi hành . Lưu ý là lệnh được kết thúc bằng 2 dấu " ; " và kết thúc cấu trúc case bằng từ khóa esac (viết ngược lại của từ case) .

Ví dụ :

```
# Name :  Menu
# Mục đích: cho phép người sử dụng in một file , xóa một file , hoặc
                    thoát chương trình
echo  "Please choose either P, D or Q to : "
echo  "[P]rint a file "
echo  "[D]elete a file "
```

```
echo  "[Q]uit"
read  response
case $response in
        P|p)  echo  "Name of file to print :"
              read  filename
              lp  $filename;;
        D|d)  echo  "Name of file to delete :"
              read  filename
              rm  $filename;;
        *)    echo  "leaving now"
esac
```

"P|p" có nghĩa là "P" hoặc "p" . " * " đại diện cho tất cả các kí tự không phải là D, d , P hoặc p .

Ta có thể lấy số các đối số của dòng lệnh bằng cách dùng "$#" .

**2. Cấu trúc if :**

Cú pháp :

```
if      command1
then    command2
else    command3
fi
```

Ví dụ :

```
# Tên : checkname
# Mục đích: xác nhận nếu người đó đang đăng nhập hệ thống
# Cách dùng : checkname  login_name
if
        who |  grep  $1 > /dev/null
then
        echo  "$1 đang đăng nhập hệ thống ."
else
        echo  "$1 không có ở đây ."
fi
```

**3. Cấu trúc lặp for :**

Với vòng lặp for , bạn đặc tả một tập hợp các file hoặc các giá trị để sử dụng với một số lệnh . Để copy mọi file có tên tận cùng là .txt đến thư mục textdir ,bạn sử dụng
vòng lặp for như sau :

```
for  i  in  *.txt
do
        cp  $i  textdir/$i
done
```

Khi thông dịch Shell cho phép biến i nhận tên của bất cứ file nào trong thư mục hiện hành mà có tên kết thúc bằng .txt . Bạn có thể sử dụng biến $i với bất cứ câu lệnh nào nằm giữa từ khóa do và done .

Script sau đây sẽ in một hoặc nhiều file , file nào in được thì bổ sung tên file đó vào file có tên là printed , còn file nào không in được thì bổ sung tên file đó vào file notprinted . Kí tự $* đại diện cho mọi tham số của dòng lệnh .

```
for  i  in  $*
do
        if  lp -t  $i  -dlasers  $i > /dev/null
```

```
            then
                Echo  $i >>printed
            else
                Echo  $i >>notprinted
            fi
        done
```

**4. Cấu trúc while :**

**Ta xem xét script sau :**

```
#Tên : checkmail
#Mục đích : thông báo cho người dùng nếu hộp thư của họ có thay đổi
#MAIL là biến đặc biệt cho biết hộp thư của người sử dụng
# lấy kích thước của hộp thư để so sánh
cp  $MAIL  omail
while  diff  omail  $MAIL > /dev/null      # lệnh diff để so sánh 2 file
do
        cp  $MAIL  omail
        sleep  30                  # ngừng 30 giây
done
echo  "New mail ! " | write  $LOGNAME
```

*Script trên sẽ dừng vòng lặp khi hộp thư có sự thay đổi ( tức là $MAIL khác* omail) và khi đó sẽ thông báo "New mail !" .

- ***Để thoát kỏi các vòng lặp for và while ta dùng lệnh break . Lệnh break(n) cho phép ra khỏi n mức của vòng lặp .***
- ***Lệnh continue cho phép bỏ qua các lệnh còn lại và bắt đầu chu trình mới .***

## III . Lệnh test :

### 1. Các lựa chọn để kiểm tra tệp :

| Lựa chọn | Ýi nghĩa |
|---|---|
| -f | Thành công nếu file tồn tại và là file hợp lệ |
| -d | Thành công nếu file là một thư mục |
| -r | Thành công nếu file tồn tại và có thể đọc |
| -s | Thành công nếu file tồn tại và không rỗng |
| -w | Thành công nếu filetồn tại và có thể ghi |
| -x | Thành công nếu file tồn tại và có thể thực thi |

Ví dụ : test -r abc # Thành công nếu file tồn tại và chỉ đọc

### 2. Các lựa chọn để kiểm tra số :

| **Lựa chọn** | **Ý nghĩa** |
|---|---|
| -eq | Bằng nhau (equal) |
| -ne | Không bằng nhau (Not equal ) |
| -ge | Lớn hơn hoặc bằng (Greater than or equal ) |
| -gt | Lớn hơn ( Greater than ) |
| -le | Nhỏ hơn hoặc bằng (Less than or equal ) |
| -lt | Nhỏ hơn (Less than ) |

Ví dụ : # Tên script: chao

```
hour=`date +%H`                 # lấy giờ
if  test  $hour   -lt  12     # nếu trước 12 giờ ...
then
      echo  "Good Morning, $LOGNAME"
else
      if  test  $hour   -lt  17       # nếu trước 17 giờ ...
      then
            echo  "Good Afternoon, $LOGNAME"
      else
            echo  "Good Everning, $LOGNAME"
      fi
fi
```

**3. Kiểm tra xâu kí tự :**

**Cú pháp :**

| | |
|---|---|
| ["str1"="str2"] | : đúng nếu str1=str2 |
| test "str1" != "str2" | : đúng nếu str1<str2 |
| test -z "$a" | : đúng nếu xâu trong biến a rỗng |
| test -n "$a" | : đúng nếu xâu trong biến a không rỗng |

**4. Kết hợp các điều kiện :**

**Các toán tử so sánh có thể tổ hợp với :**

| | |
|---|---|
| -a | : và |
| -o | : hoặc |
| ! | : đảo ( phủ định ) |
| \( ... ... )\ | : gộp |

Ví dụ : $test \( -r file1 -o, -r file2 \) -a -w file3

*Lệnh trên đúng nếu ((file1 tồn tại, chỉ đọc hoặc file2 tồn tại ,chỉ đọc ) và file3 tồn tại , có thể ghi ) .*

## IV . Các phép tính số học:

Let được dùng để thực hiện các phép tính số học

Các toán tử số học : +, -, *, /, %

Các toán tử so sánh : >=, >, <, <=, = = , !=

Các toán tử logic : ! , && , | |

```
Ví dụ :   $integer   I=10  j=2  k    # khai báo các biến I , j , k
          $let  "k=I+j"
          $echo  $k          # kết quả là :
          12
          $((I<j))
          $echo  $?          # kết quả là:
          1
```

## V . Xuất một biến đến một Shell mới :

Cú pháp : export Tên biến

Khi bạn tạo các biến Shell hoặc cho giá trị các biến đang tồn tại , chúng tồn tại trong quá trình Shell đang chạy . Một biến được đặt trong Shell lúc đăng nhập hệ thống là có giá trị đối với mọi đối số của dòng lệnh . Một biến

đặt trong một Shell chỉ có giá trị trong Shell đó . Giá trị đó không xuất hiện hoặc được đặt lại khi bạn thoát Shell đó .

Giả sử bạn viết một Shell script tên là whatday như sau :

```
echo  " Today  is  $today ."
today=Friday
echo  " Today is  $today "
```

*Bây giờ nhập vào 4 lệnh sau từ dòng lệnh :*

```
chmod  +x  whatday        # lệnh này dịch script trên thành file whatday
today=Thursday
whatday
echo  $today
```

*Các dòng sau sẽ xuất hiện trên màn hình :*

```
Today is .
Today is Friday .
Thursday
```

*Giá trị Thursday của biến today trong login Shell không có tác dụng trong script whatday . Khi script thi hành thì biến today vẫn chưa được định nghĩa (nên hiển* thị Today is . ) cho đến khi nó nhận giá trị là Friday. Khi script kết thúc , bạn trở về login Shell và biến today vẫn giữ giá trị là Thursday .

Nếu ta thêm lệnh xuất biến " export today " vào :

```
today=Thursday
export  today
whatday
echo  $today
```

*Thì các dòng sau sẽ xuất hiện trên màn hình :*

```
Today is Thursday .
Today is Friday .
Thursday
```

*Nếu bạn muốn thay đổi các biến môi trường thì phải đặt lại biến môi trường trong* file .profile và theo sau nó là lệnh xuất biến export . Ví dụ , dùng một trình soạn thảo văn bản đặt các dòng sau vào file .profile :

```
PS1="Xin chao $"          # đặt lại dấu nhắc $ thành Xin chào $
export  PS1
TERM=vt100                # đặt lại kiểu thiết bị cuối là vt100
export  TERM
```

*Chú ý : Những thay đổi do bạn tạo ra trong file .profile hoặc .login không nhận* được hiệu ứng cho đến khi bạn thoát khỏi hệ thống và đăng nhập trở lại .

*Chương IV :*

# Các loại Shell khác nhau

## I. Bourne Shell :

Bourne Shell được biết với tên sh là một trong những loại Shell đầu tiên và thông dụng nhất hiện nay . Để bắt đầu sử dụng Shell bạn hãy gõ lệnh :

$sh

Khi đó con trỏ lệnh sẽ chuyển sang dạng mới là một dấu đô la ($)

Khuôn dạng chung cho các lệnh trong Bourne Shell là :

$ command arg1  arg2 .... argn

Trong đó : arg1  arg2 .... argn là các tham số của lệnh .

Để liệt kê các biến hiện có của Bourne Shell bạn hãy sử dụng lệnh :

$ set

BẢNG LIỆT KÊ CÁC BIẾN

| Tên biến | Giá trị |
|---|---|
| HOME | / home / a-Function /gmeghab |
| LANG | C |
| PS1 | $ |
| PS2 | > |
| PWD | / bin |
| TZ | US / Eastern |
| ISF | = |
| PATH | (liệt kê tất cả các thư mục mà người dùng hiện thời có quyền truy nhập ) |
| SHELL | / bin / csh |
| TERM | Wyse50 |

| | |
|---|---|
| OLWMMENU | / home / a-s / gmeghab / .openwin |
| OPENWINHOME | / usr / openwin |
| USER | Gmeghab |

Bất kì biến nào trong danh sách trên đều có thể thay đổi được giá trị bằng cú pháp:

**$ variable = value**

Để xem giá trị của từng biến riêng biệt bạn sử dụng lệnh :

**$ echo $ <tên biến>**

Cũng giống những Shell khác ,trong Bourne Shell thì một Script là một tệp chứa chuỗi các lệnh thực hiện theo đúng thứ tự sắp xếp trong tệp . Bạn có thể sử dụng bất kì trình soạn thảo nào để soạn một tệp Script .

Ví dụ : Soạn thảo một tệp Script có tên là Morning :

*$cat Morning*
*date*
*users*
*who*

Tạo tập tin có tên File1

*$cat File1*
*aa*
*bb*
*OK*

Với Script , chúng ta cũng có thể thực hiện một chuỗi liên tiếp các Script bằng cách gõ chúng trên cùng một dòng lệnh và mã Script ngăn cách với nhau bằng một dấu chấm phẩy (;) .

Ví dụ : $morning ; afternoon ; evening

Để liệt kê tất cả các lệnh của Bourne Shell bạn hãy đọc các tài liệu có sẵn bằng cách gõ lệnh sau :

*$ man sh*

## II. Korn Shell :

Korn Shell cũng tương tự như Bourne Shell ,ngòai nó có thêm bốn chức vụ rất quan trọng mà Bourne Shell không có :

+ Theo dõi hoạt động của người dùng (history file) .
+ Quản lí các hoạt động của người dùng (jod control) .
+ Chức năng thao tác với các bí danh .
+ Trình soạn thảo lệnh (command editor) .

Để chạy Korn Shell bạn hãy chạy chương trình có tên : ksh .

Korn Shell cũng định nghĩa các biến cục bộ của riêng nó và một phần của các biến này có tên và chức năng giống như trong Bourne Shell , ngoài ra còn thêm một số biến sau :

| Tên biến | Giá trị |
|---|---|
| PS3 | Thông báo cho lệnh SELECT |
| PS4 | Thông báo cho lệnh TRACE |
| SECOND | Thời gian tính bằng giây để nạp Shell |
| TMOUT | Thời gian sử dụng Shell |
| PRID | ID của tiến trình Shell |

Các lệnh trong Korn Shell :

- Thay đổi các giá trị ngầm định của các biến cục bộ :

*$ variable = value*

- Xem chi tiết một lệnh nào đó :

*$ type command_name*

- Xem chi tiết về thời gian thực ,thời gian do người dùng định nghĩa và thời gian của hệ thống :

*$ time*

## III. Sử dụng C Shell :

C Shell được thiết kế để thay thế Bourne Shell , tên của chương trình là csh nó được xây dựng dựa trên ngôn ngữ lập trình C .

**a. Bí danh :**

Để sử dụng các lệnh của C Shell một cách nhanh nhất ,bạn gán cho mỗi lệnh này một bí danh nào đó . Để thực hiện như vậy bạn đánh lệnh :

% alias newcommandname oldcommandname options

Ví dụ : Để gán bí danh cho lệnh ls bạn làm như sau :

*% alias li ls-als*

Từ đây bạn có thể sử dụng lệnh ls-als bằng lệnh li .

Ngoài việc gán bí danh cho lệnh , bạn cũng có thể tiến hành gán bí danh cho các ứng dụng sẵn có .

Ví để gán bí danh cho trình ứng dụng matlab bạn làm như sau :

*% alias matlab/usr/bin/matlab*

**b. Xem các thông tin về các tiến trình :**

Lệnh whodo được sử dụng để liệt kê thông tin về các tiến trình trong hệ thống , đồng thời các tiến trình này do người nào sử dụng .

Lệnh %ps[ts] được sử dụng để xem thông tin về trạng thái của các tiến trình .

Với [ts] : tham số đi kèm .

Một số tham số tùy chọn của lệnh ps

| Tùy chọn | Ý nghĩa |
|---|---|
| -a | Hiển thị tất cả các tiến trình |
| -u process -id | Hiển thị tất cả các thông tin liên quan đến tiến trình hiện đang hoạt động với id đã cho ,các thông tin này bao gồm :id của người dùng ,id của tiến trình ,thời gian bắt đầu hoạt động , thời gian kết thúc , thời gian sử dụng CPU , tên lệnh đã gọi tiến trình . |
| -x terminal | Chỉ định thiết bị cuối được sử dụng để hiển thị thông tin . |
| -e | Hiển thị các thông tin liên quan đến các tiến trình bao gồm : id của tiến trình ,thiết bị cuối ,thời gian ,tên lệnh . |
| -f | Hiển thị tất cả các thông tin liên quan đến các tiến trình bao gồm : id của người dùng ,id của tiến trình ,thiết bị cuối . |

Nếu không có tham số đi kèm , lệnh ps sẽ xuất ra thông tin về tất cả các chương trình đang chạy .

| Tiêu đề | Ý nghĩa |
|---|---|
| PID | ID của tiến trình |

| | |
|---|---|
| TT | Thiết bị cuối điều khiển tiến trình |
| S | Tình trạng của tiến trình |
| TIME | Thời gian sử dụng CPU của tiến trình |
| COMMAND | Tên lệnh đã gọi tiến trình này |

**c. Các biến :**

Giống như tất cả các Shell khác , C Shell cũng có những biến riêng của bản thân nó ,với những biến này bạn có thể tiến hành khai hoặc gán giá trị . Để khai báo một biến trong C Shell bạn có thể sử dụng một trong ba lệnh sau : set ,@ setenv .

**Khai báo biến bằng lệnh set :**

Để khai báo hoặc gán giá trị cho một biến cục bộ bằng lệnh set bạn sử dụng cú pháp :

```
$set  username
$echo  $username
```

Để xóa biến vừa được thêm vào danh sách (username) bạn làm như sau :

```
$unset username
$ set
```

Để liệt kê tất cả các tên trong tệp thư mục hiện thời bạn làm như sau :

```
$ echo*
```

Bạn cũng có thể gán giá trị cho biến từ bàn phím bằng cách :

```
$ set newname = $<
```

Bạn cũng có thể kiểm tra kích thước (số phần tử của mảng ) bằng lệnh :

```
$ echo  $# variable
```

**Khai báo biến bằng lệnh @ :**

Lệnh @ khai báo các biến cục bộ , tuy nhiên lệnh này yêu cầu người dùng chỉ được khai báo và gán các biến có giá trị số .

```
% @ name
@: syntax error
% @ name = 5
% echo $name
5
```

Lệnh @ cũng cho phép bạn tính tóan các biểu thức số . Cú pháp để tính tóan biểu thức số với lệnh @ hòan toàn tương tự như cú pháp sử dụng trong ngôn ngữ C .

**Các biến của Shell và các biến môi trường :**

Các biến trong C Shell được phân biệt làm hai loại : biến của Shell và biến môi trường . Biến môi trường được hiển thị bằng các chữ cái hoa , biến của Shell được hiển thị bằng chữ cái thường . Các biến môi trường có thể sử dụng bởi các tiến trình con của C Shell trong khi các biến cục bộ thì không .

Các biến của C Shell được lưu trong hai tệp : .login và .cshrc . Để hiển thị giá trị của các biến lưu giữ trong hai tệp này bạn sử dụng lệnh cat :

```
$cat.cshrc
...
$ cat.login
...
```

Lệnh sentenv hiển thị tất cả các biến môi trường , những biến này được hệ thống hiển thị bằng các chữ hoa .

$sentenv

**Các biến của Shell thường được sử dụng nhất :**

| Tên biến | Mô tả |
|---|---|
| $argv | Biến này được thừa kế từ môi trường lập trình C , trong đó argv [0] chứa tên chương trình , argv [1] chứa tham số đầu tiên của dòng lệnh . |
| $cdpath | Biến này được chứa trong tệp .cshrc và chứa tên các thư mục , biến này tác động đến sự hoạt động của lệnh cd . |
| $cwd | Thư mục làm việc . |
| $history | Biến này quản lý kích thước của danh sách lưu trữ quá trình sử dụng (history list) . |
| $home | Chứa thư mục gốc ứng với từng người dùng (thư mục gốc này thường được tham chiếu bởi kí hiệu ~) . |
| $ignoreoff | Khi giá trị của biến này được đặt ta phải sử dụng ta phải sử dụng lệnh exit để chấm dứt việc sử dụng Shell thay vì sử dụng tổ hợp phím Ctrl+d . |
| $mail | Tệp lưu giữ hộp thư của người dùng . |
| $noclobber | Biến này được đặt để ta không thể ghi đè một cách vô tình lên một tệp có sẵn khi bạn định hướng lại đầu ra (output) . |
| $path | Biến này được lưu trong tệp .cshrc , nó chứa những thư mục mà ta hay sử dụng nhất , và khi ta gõ một lệnh nào đó ta không nhất thiết phải gõ đầy đủ tên và đường dẫn của thư mục . |
| $prompt | Biến này được lưu trong tệp .cshrc , chứa dấu nhắc mà người dùng sẽ nhìn thấy trên dòng lệnh . |
| $savehist | Số lượng các lệnh bạn đã sử dụng khi bạn chấm dứt việc sử dụng . |
| $status | Biến này chứa trạng thái kết thúc của lệnh được sử dụng gần đây nhất<br>Nếu lệnh này được thực hiện thành công thì giá trị này sẽ là 0 và ngược lại giá trị này là -1 . |
| $shell | Chứa thư mục của Shell . Với C Shell thì giá trị của biến này là /bin/csh . |

## IV. Lập trình với C Shell :

Người dùng có thể sử dụng ngôn ngữ lập trình của Shell không chỉ được sử dụng để thực hiện một chuỗi các lệnh . Giả sử bạn cần tạo một thư mục mới và sao chép tất cả các tệp ở một thư mục sẵn có vào thư mục mới này . Để thực hiện công việc đặt ra bạn cần phải sử dụng cơ chế lặp , khái niệm và cách sử dụng cơ chế này trong lập trình với C Shell được trình bày trong phần sau đây :

### a. Câu lệnh if :

Cú pháp : if (express) command

Ví dụ :

```
#!/ bin/ csh
set bad = 0
```

*If ($bad = = 0) echo "I am bad"*

Chương trình trên khai báo biến có tên bad như một biến cục bộ và giá trị đầu là 0 . Câu lệnh if kiểm tra giá trị của biến bad và đưa ra thông báo "I am bad" trên màn hình .

**b. Câu lệnh if-else :**

*Cú pháp :*

```
If (express)
        Commands
Else
        Commands
Endif
```

Ví dụ :

```
#!/ bin / csh
set mychoice = openwin
If ($mychoice = = openwin)
        #
        unset mychoice
        echo-n "Starting Open Windows ..."
        clear    # get rid of annoying cursor rectangle
        echo-n "Automatically logging out ..."
        #
else
        #
        unset mychoice
        echo-n "Starting Sun View ..."
        #default Sun View ...
        echo-n "Automatically logging out ..."
        #
endif
```

( Trong chương trình , mychoice là một biến cục bộ và đặt giá trị ban đầu của biến là openwin . Câu lệnh If-else kiểm tra giá trị của biến mychoice :

+ Nếu mychoice = openwin thì xóa biến mychoice và đưa ra thông báo :

"Starting Open Windows ..."
"Automaticlly logging out ..."

**c. Câu lệnh Switch :**

Cú pháp :

```
Switch (express)
        Case comparasion 1 :
                Commands
                Breaksw
        Case comparasion 2 :
                Commands
                Breaksw
        Default :
Endsw
```

Ví dụ :

```
#!/ bin / csh
set mychoice = openwin
```

```
Switch($mychoice)
        Case openwin :
        unset mychoice
        echo-n "Starting Open Windows ..."
        clear    # get rid of annoying cursor rectangle
        echo-n "Automatically logging out ..."
        breaksw
        #
        case sunview :
                unset mychoice
                echo-n "Starting Sun View ..."
                #default Sun View ...
                echo-n "Automatically logging out ..."
                #
        endsw
```

(Trong chương trình , mychoice là một biến cục bộ và đặt giá trị ban đầu của biến là openwin . Câu lệnh switch kiểm tra giá trị của biến Mychoice và cho kết quả tương tự như ở chương trình ví dụ của câu lệnh If-else :

+ Nếu mychoice = = openwin thì xóa biến mychoice và đưa ra thông báo:

"Starting Open Windows ..."
"Automaticlly logging out ..."

+ Ngược lại cũng xóa biến mychoice nhưng đưa ra thông báo :

"Starting Sun View ..."
"Automaticlly logging out ..."

**d. Vòng lặp foreach :**

-Vòng lặp này giống như vòng lặp for trong các ngôn ngữ lập trình khác .

Cú pháp :

```
Foreach variable (wordlist)
        Commands
End
```

Ýi nghĩa : Vòng lặp này duyệt tất cả các thành phần của một danh sách được chỉ định trong wortlist . Trong cú pháp trên thì wordlist luôn phải được đăt giữa hai dấu ngoặt đơn ( ) .

Ví dụ : Sử dụng vòng lặp foreach để đổi tên của nhiều tệp khác nhau sang tên mới :

```
#! /bin / csh
set tencu; set tenmoi
foreach tencu (*)
echo $tencu
        echo "nhap ten moi :"
        set tenmoi = $<
echo $tenmoi
mv $tencu $tenmoi
end
ls-als
```

+ Trong ví dụ trên , hai biến tencu và tenmoi là hai biến cục bộ nhưng không có giá trị khởi đầu .

+ Vòng lặp foreach gán lần lượt tất cả các giá trị trong wordlist cho biến tencu .Trong Shell cũng như trong ngôn ngữ C ,ký tự (*) biểu hiện cho tất cả các tệp trong thư mục cho trước ở đây là thư mục làm việc ,các tên tệp này được gán cho biến tencu theo thứ tự chữ cái trong mã ASCII .Tên mới của mỗi tệp đều được nhập vào từ bàn phím thông qua biến tenmoi .

+ Dòng lệnh : mv $tencu $tenmoi đổi tên tệp được chỉ ra trong biến tencu thành tên mới được chỉ định trong tenmoi .

**e.Vòng lặp while :**

- Vòng lặp while tương đương với vòng lặp foreach .

Cú pháp :

```
While (expr)
        Commands
....
End
```

**Ý nghĩa : Trong cấu trúc trên thì expr đặc trưng cho biểu thức sẽ được tính tóan** trước khi bắt đầu mỗi lần lặp . Nếu giá trị này là true thì những lệnh tiếp theo sẽ được thực hiện cho đến khi gặp câu lệnh end , nếu giá trị này là false thì vòng lặp sẽ kết thúc .

**f. Vòng lặp repeat :**

Cú pháp :

```
Repeat count command
```

**Ý nghĩa : Phạm vi sử dụng vòng lặp này tương đối hạn hẹp và thông thường thì** nó được sử dụng để thực hiện những thao tác đơn giản lặp đi lặp lại .

Trong đó :

Count : là biến với giá trị đã được xác định trước .

Command : là một lệnh do người lập trình chỉ định .

*Chương V :*

# Trình soạn thảo vi

*Nội dung : Giới thiệu trình soạn thảo vi , cung cấp một số kiến thức cơ sở để có thể* soạn thảo được văn bản hay chương trình .

## I. Khởi động vi :

1. Giới thiệu : Vi (video interactif) là chương trình sọan thảo văn bản theo trang màn hình :

- Màn hình được xem như một cửa sổ mở trên tập tin .
- Có khả năng di chuyển con trỏ đến bất kì vị trí nào trên màn hình .
- Cửa sổ có thể di chuyển tự do trên tập tin .

Phần lớn các phím dùng độc lập hoặc kết hợp với phím Shift và Ctrl để tạo ra các lệnh của vi . Khi một lệnh bị gõ sai , vi báo hiệu bằng cách nháy màn hình , kêu còi hay 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 .**

### 2. Khởi động vi :

Ta có thể gọi vi với tên tập tin văn bản :

$vi  tên_tập tin

Cửa sổ soạn thảo sẽ được mở tại đầu tập tin . Nếu tập tin chưa tồn tại , nó sẽ được tạo bởi lệnh ghi . Dòng cuối cùng trên màn hình được dùng cho những công việc sau :

- Vào các lệnh .
- Thống kê .
- Báo lỗi .

Khi ta chỉ muốn xem nội dung một tập tin đã có trên đĩa , dùng lệnh :

$view tên_tập tin

**Để thóat trình xem , nhấn phím ESCgõ  :q!  nhấn phím Enter**

**3.Thóat khỏi vi :**

Muốn ra khỏi vi và ghi lại nội dung tập tin , bạn nhấn 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 trước đó

:q!

Khi ở trong chế độ soạn thảo của vi , muốn làm việc với các lệnh SHELL , ta có thể làm như sau :

- Chạy một lệnh SHELL

:! Lệnh

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

:! sh
$ lệnh
$ Ctrl-D

## II. Soạn thảo văn bản :

### 1. Chèn văn bản :

- Chèn ký tự trên một dòng :

a <text> <ESC>

- Chèn ký tự vào sau vị trí con trỏ . Lệnh không được hiển thị trên màn hình .

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

i <ESC> Xen ký tự vào sau con trỏ .
A <ESC> Xen ký tự vào cuối dòng .
i <ESC> Xen ký tự vào cuối dòng .

- Chèn dòng :

o <ESC> Xen một dòng vào trước dòng chứa con trỏ .
o <ESC> Xen một dòng vào sau dòng chứa con trỏ .

### 2. Di chuyển con trỏ trong tập tin :

- Theo ký tự :

Sang trái : dùng phím trượt trái hoặc h hoặc backspace
Xuống dòng : dùng phím trượt xuống hoặc j hoặc linefeed
Sang trái : dùng phím trượt phải hoặc i hoặc espace
Lên dòng : dùng phím trượt lên hoặc k hoặc

- Theo dòng :

^ về đầu dòng
$ cuối dòng
Enter đầu dòng kế tiếp

- Đầu dòng trên

0(null) về đầu dòng vật lý (dòng bắt dầu bằng dấu cách hoặc Tab)

- Theo màn hình

H về đầu màn hình (Home)
M về giữa màn hình (Miđle)
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 :
  ( về đầu câu
  ) về cuối câu

Lưu ý kết thúc một câu là dấu .! hoặc ?

- Theo cửa sổ (window):
  z dòng hiện tại ở giữa cửa sổ .
  z<Enter> dòng hiện tại ở đầu cửa sổ
  ^D dòng hiện tại ở cuối cửa sổ
  ^U xuống nữa cửa sổ
  ^F xuống một cửa sổ (-2 dòng)
  ^B lên một cửa sổ (-2 dòng)

*Lưu ý :^là ký hiệu phím CTRL .*

-Theo số thứ tự dòng :

Để hiển thị số thứ tự các dòng soạn thảo :
: set nu

Xóa bỏ hiển thị trên :
: set nonu

:n<Enter> hoặc nG Chuyển con trỏ đến dòng thứ n
:s hoặc G Đến cuối dòng văn bản
:se list hiển thị các kí tự ẩn

- Tìm theo dãy kí tự :
  / kí hiệu chiều tìn xuôi
  ? kí hiệu chiều tìm ngược
  /string chuyển con trỏ đến dòng chứa dãy kí tự theo chiều xuôi
  ?string chuyển con trỏ đến dòng chứa dãy kí tự theo chiêu ngược
  // lặp lại tìm xuôi
  ?? lặp lại tìm ngược

**3. Xóa văn bản :**

- Xóa kí tự :
  x xóa kí tự taịo vị trí con trỏ
  3x xóa 3 kí tự
  x xóa kí tự trước vị trí con trỏ
- Xóa dòng văn bản :
  dd hoặc :d<CR> xóa dòng chứa con trỏ
  3dd xóa 3 dòng bắt đầu từ vị trí văn bản
  d$ hoặc D xóa đến cuối dòng
  dw xóa từ chứa con trỏ
  3dw xóa 3 từ
  d/string xóa đến khi hết dãy kí tự

**4. Thay thế văn bản :**

-Thay thế kí tự :
  rc thay thế kí tự đại diện bằng kí tự c
  R <ESC> thay thế số kí tự bằng dãy 'text"
- Thay thế dòng :
  S <ESC> xóa dòng hiện tại và thay thế nó bằng "text"
- Thay thế từ :

| | |
|---|---|
| cw <ESC> | thay thế một từ bằng "text" . Từ được thay thế tính từ vị trí con trỏ đến kí tự $ |
| c2w<ESC> | thay 2 từ |
| c hoặc c$ | thay thế đến cuối dòng |
| c/string | thay thế đến hết chuỗi |

**5. Xóa hoặc lặp lại tập lệnh :**

- Xóa lệnh

| | |
|---|---|
| u | xóa tác dụng của lệnh cuối cùng |
| U | xóa tất cả các thay đổi đã làm trên dòng hiện tại |

- Lặp lại lệnh

| | |
|---|---|
| . | lặp lại lệnh thay dổi văn bản |

**6. Xem trạng thái văn bản đang soạn thảo :**

^G Hiển thị tên , trạng thái ,số dòng , vị trí cursor và phần văn bản tính từ vị trí con trỏ đến cuối dòng văn bản .

**7. 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 riêngcho đến lần xóa sau . Lệnh p hoặc P cho phép lấy lại phần văn bản từ vùng đệm đó . Trước khi thực hiện lệnh này , dấu nháy phải được đặt vào vị trí cùng kiểu với phần văn bản có trong vùng đệm :

- kí tự
- từ
- dòng
- cuối dòng

**p sao phần văn bản xóa lần cuối vào sau đối tượng cùng kiểu**
**P sao phần văn bản xóa lần cuối vào trước đối tượng cùng kiểu**
*** cách khác để chuyển dòng :**

| | |
|---|---|
| :5, 10m 20 | chuyển các dòng từ 5 đến 10 tới sau dòng 20 |

+ Sao chép văn bản

Lệnh y (yank) cho phép sao chép phần văn bản ta muốn vào vùng đệm . Muốn sao phần văn bản từ vùng đệm ra, ta phải chuyển cursor đến 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 |

Cách khác dùng để sao chép dòng :

| | |
|---|---|
| :5, 8 t 25 | sao chép các dòng từ 5 tới 8 tới sau dòng 25 |

## III . 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 tập tin chứa một số lệnh đã chạy (của history):

$fc -nl n1 n2 > cmd

**cmd là một tập tin chứa các lệnh của history từ lệnh ngày đến lệnh n2 .**

## TÀI LIỆU THAM KHẢO

1. Special using editor Linux ( Chương 18 - Phần IV )

2. Nguyễn Tấn Khôi - " Hệ thống mạng Linux " , Đà Nẵng , 2000.

3. Đỗ Duy Việt & Nguyễn Hoàng Thanh Ly - "Linux Kernel "

---------- ***** ----------