Chương 03. File System_Phần 2

4. Thi hành đồng thời nhiều lệnh

- Linux cho phép nhập nhiều lệnh tại một thời điểm. Các lệnh sẽ thực hiện một cách tuần tự
- **Cú pháp**

  command1; command2; ...; commandN

5. Pipeline

- Linux cho phép thực hiện kết nối các tiến trình bằng cách cho kết quả xuất của một lệnh này là đối số nhập của một lệnh khác. Cơ chế này gọi là pipeline hay đường ống.
- **Cú pháp**

  command1 |command2 |...|commandN

- **Ví dụ:**

  cat myfile |head -10|tail -3

6. Lệnh head

- **Cú pháp**

  head [option] [file]...

- Xem phần nội dung đầu mỗi file. Nếu có nhiều file thì trong kết quả, đứng trước nội dung của một file là đầu đề có chứa tên tập tin. Nếu lệnh head không có đối số file thì head sẽ đọc nội dung từ thiết bị nhập chuẩn
- **Option**

  -n: số dòng đầu tiên của file cần hiển thị. Mặc định là 10.

  -q: không hiển thị phần đầu đề chứa tên file trong trường hợp mở nhiều file cùng lúc.

7. Lệnh more và less

more dùng để xem nội dung tập tin theo từng trang màn hình.

- **Cú pháp**

  more [options] [file]...

- **option**

  -num: xác định số dòng mỗi màn hình

  +linenum: Dòng bắt đầu hiển thị

  -s: Xóa bớt các dòng trắng (nếu có), chỉ để lại một dòng trắng giữa mỗi đoạn.

- Lệnh less tương tự như more nhưng cho phép dịch chuyển lên xuống bằng các phím mũi tên.

8. Chuyển hướng đầu vào/ đầu ra

- Mọi chương trình chạy trong shell đều mở ba tập tin (thiết bị):
  - Nhập chuẩn (standard input)
  - Xuất chuẩn (standard out put)
  - Lỗi chuẩn (standard error). Các tập tin này cung cấp ý nghĩa truyền thông giữa các chương trình và tồn tại trong suốt quá trình tiến trình hoạt động.

  - Thiết bị nhập chuẩn cung cấp một cách thức gửi dữ liệu cho một tiến trình. Mặc định dữ liệu nhập chuẩn được đọc từ bàn phím.
  - Thiết bị xuất chuẩn cung cấp một cách thức để chương trình gửi dữ liệu ra. Mặc nhiên thiết bị xuất chuẩn là màn hình.
  - Tập tin lỗi chuẩn là nơi chương trình ghi lại báo cáo về bất kỳ lỗi nào xảy ra trong quá trình thi hành. Mặc định tập tin này là màn hình.

- Một chương trình có thể được chỉ ra cho biết nơi nào thông tin sẽ được gửi vào và đâu là nơi xuất thông tin ra, bằng cách sử dụng chuyển hướng đầu vào/ đầu ra
  - Linux sử dụng ký tự nhỏ hơn **<** để chỉ hướng đầu vào.
  - Dấu lớn hơn **>** để chỉ hướng đầu ra.

- **Chuyển hướng đầu vào**

  < file

  trong một lệnh để chỉ dẫn cho shell đọc thông tin đầu vào từ một tập tin **file** thay thế cho việc nhập từ bàn phím.

- **Ví dụ:**

  cat < /etc/passwd

  cat > test1 < /etc/passwd

- **Chuyển hướng đầu ra**

  >file

  trong một lệnh cho phép shell chuyển đầu ra của một lệnh vào tập tin file thay thế cho việc xuất tập tin ra xuất chuẩn (màn hình). Nếu tập tin file đã có thì tập tin cũ sẽ bị ghi đè.

- **Ví dụ**

  head /etc/passwd > test2

  ls -li /etc > test3

- Sử dụng >> để bổ sung thêm nội dung đến tập tin đã có.

  >> file

  sẽ thông báo cho shell bổ sung thêm nội dung xuất của một lệnh vào cuối tập tin file. trường hợp tập tin file chưa có thì nó sẽ được tạo ra.

- **Ví dụ**

  ls -li /tmp > myls

  ls -ls /root >> myls

- **Chuyển hướng lỗi**

  Sự chuyển hướng tập tin lỗi chuẩn là khá phức tạp, phụ thuộc vào kiểu shell đang sử dụng. Trong bash chuyển hướng tập tin lỗi chuẩn bằng ký tự "**2>**".

- **Ví dụ**

```
a /etc 2> error1
ls -z /etc > test 2>error2
```


9. Cách sinh tên tập tin

- Hãy lấy ví dụ một lệnh có nhiều đối số như rm, cp:

  # rm file1 file2 file3

- Lệnh này có thể viết dưới dạng cô đọng như sau:

  # rm file?

- Ký tự dấu hỏi "**?**" ở đây được gọi là một siêu ký tự sinh tên file vì nó đại diện cho một ký tự bất kỳ.
- Các siêu ký tự sinh tên file (wildcards) là những ký tự sau: **‘ ? * [ ] – !**

  trong đó ký tự "**!**" chỉ có nghĩa với Bourne shell.

9.1- Siêu ký tự dấu hỏi "?“

Ký tự dấu hỏi sinh ra (thay cho) bất kỳ ký tự nào khác, ví dụ lệnh sau:

 # rm fi?e

sẽ có thể tương đương với lệnh:

 # rm file fine fire

9.2- Siêu ký tự dấu sao "*“

Siêu ký tự dấu sao "*" sẽ sinh ra 0, 1 hoặc nhiều ký tự bất kỳ, ví dụ lệnh:

 # cat file*

sẽ có thể tương đương với lệnh:

 # cat file file1 file_in file_out

9.3- Cặp siêu ký tự dấu ngoặc vuông "[" và "]“

Ta thường gặp cặp dấu ngoặc vuông "[" và "]" trong tên file. Ví dụ:

 # rm sa[cnN]h

Từ các ký tự ở trong cặp dấu ngoặc vuông, tức **cnN**, một ký tự bất kỳ sẽ được chọn, do đó lệnh trên tương đương với:

 # rm sach sanh saNh

(nếu những file đó tồn tại trong thư mục hiện hành).

9.4- Siêu ký tự dấu "-“

Từ ký tự đầu đến ký tự sau trong cặp dấu ngoặc vuông, một ký tự bất kỳ sẽ được chọn, ví dụ:

 # cat fi[b–m]h

tương đương với:

 # cat fibh fich filh

9.5- Siêu ký tự dấu "!“

- Siêu ký tự "**!**" chỉ có nghĩa với Bourne shell là phải lấy một ký tự khác những ký tự trong cặp dấu ngoặc vuông sau nó
- **ví dụ:**

  % rm fi[!fgtk]h

  tương đương với:

  % rm fiAh fich fioh

9.6. Trung hoà các siêu ký tự

Trung hoà nghĩa là làm cho shell không phải hiểu các ý nghĩa đặc biệt nói trên của các siêu ký tự nữa, bằng dấu chéo ngược và dấu nháy đơn.

- **1- Bằng ký tự dấu chéo ngược**

  Ví dụ có thể tạo ra một file có tên “**file***”:

  # cat > file*
- **2- Bằng ký tự dấu nháy đơn**

  Có thể trung hoà các siêu ký tự trong tên file, ví dụ trên tương đương với:

  # cat > ‘file*’

10. Lệnh wc

- **Ý nghĩa**: Đếm số dòng, số từ, số ký tự trong file
- **Cú pháp**

  wc [option] file

  10 120 540
- **Các lựa chọn**

  -c : số byte

  -m : số ký tự

  -l : số dòng

  -w : số từ
- **Ví dụ**:

  echo "Tong so File :" `ls | wc -l`

11.Lệnh fgrep

- **Ý nghĩa** : *hiển thị các dòng chứa "**chuỗi**" trong "**file**"*. "chuỗi" là tập ký tự, bao gồm các ký tự đặc biệt, được đặt trong cặp dấu nháy đơn hoặc kép.

  fgrep [Option] "chuỗi" file
- **Các Option**

  -n đánh số các dòng kết quả theo file gốc

  -v kết quả là những dòng không chứa "chuỗi"

  -i không phân biệt chữ hoa, chữ thường

12. Lệnh grep

- **Ý nghĩa** : *hiển thị các dòng chứa "**chuỗi**" trong "**file**"*. "**chuỗi**" là loại biểu thức regular.

  grep [tùy_chọn] chuỗi file

- **Biểu thức regular** : chuỗi ký tự, bao gồm cả các ký tự có ý nghĩa đặc biệt đối với lệnh **grep**. Các ký tự đó là :

  **^ $. * [ ] - **

Các ký tự đặc biệt đối với grep

12.1. ^: dấu mũ không phải là phím điều khiển <Ctrl> như trong ^D, nếu ở đầu một biểu thức regular thì những ký tự đi sau **^** sẽ coi như ở đầu dòng của "**file**":
 # grep '^in ra' file
ví dụ trên sẽ hiển thị những dòng có chuỗi ký tự "in ra" ở đầu dòng.

12.2. $**: nếu ở cuối một biểu thức regular thì những ký tự đi trước **$ sẽ coi như ở cuối dòng:
 # grep 'end$' file
ví dụ trên sẽ hiển thị những dòng kết thúc bằng chuỗi ký tự "end".

12.3. **“*”**: dấu sao thể hiện một chuỗi n ký tự (n là 0 hoặc nguyên dương) giống ký tự đi trước *:
 # grep ‘ ll*T’ file

ví dụ trên sẽ hiển thị những dòng chứa các chuỗi ký tự toàn **l** rồi đến **T.**

12.4. **“.”**: dấu chấm thể hiện một ký tự ASCII bất kỳ, trừ <RETURN>:

 # grep ‘ .*’ file

ví dụ trên sẽ hiển thị mọi dòng của “file", kể cả dòng trống.

12.5. **[ ]**: thể hiện một ký tự ASCII trong cặp ngoặc vuông, nhưng cần đặt giữa cặp dấu nháy như: **‘[xyzt]’**

12.6. **“-”**: dấu trừ giữa hai ký tự ASCII bên trong cặp ngoặc vuông, ví dụ [b-y], thể hiện một ký tự có trong khoảng đó, nhưng cũng cần đặt giữa cặp dấu nháy như **'[b-y]'.**

12.7. **“\”**: dấu chéo ngược sẽ bỏ ý nghĩa đặc biệt của ký tự đi sau nó và trả lại ý nghĩa gốc.

13. Lịch sử lệnh

- Một công cụ để tiết kiệm thời gian nhập lệnh là sử dụng lịch sử dòng lệnh. Bằng cách sử dụng các phím mũi tên lên xuống ta có thể tìm lại những lệnh trước đó ta đã nhập vào để thi hành.
- Theo mặc định có đến 1000 lệnh được lưu trữ và có thể xem trong file *bash_history* trong thư mục đăng nhập của người dùng.

14. Tạo tập tin liên kết ln

- **Cú pháp**

  ln [options] target [link_name]

  ln [options] target ...directory

- Lệnh **ln** thực hiện tạo một tập tin liên kết có tên là **link_name** tới tập tin **target**. Nếu không chỉ ra **link_name** thì một tập tin có tên trùng với tập tin **target** được tạo tại thư mục hiện hành.
- Trường hợp tạo liên kết cho nhiều tập tin **target** thì đối số cuối cùng phải là **directory**, thư mục này sẽ chứa các tập tin liên kết được tạo.

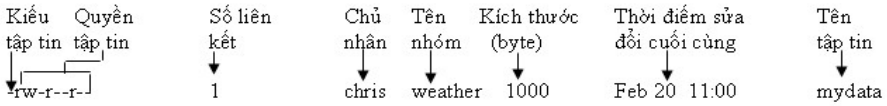
- **Option**
  - **-s**: Tạo symbolic link. Trong trường hợp mặc định (không có lựa chọn **-s**) một hard link sẽ được tạo ra.
- **Chú ý**
  - Không thể tạo được hard link cho một thư mục
  - Chỉ có thể tạo được hard link trên cùng một partition.
  - Có thể tạo được một symbolic link ngay cả khi tập tin target không tồn tại.

15. Các lệnh về quyền tập tin

15.1. Các khái niệm về quyền tập tin

- Mỗi tập tin và thư mục trong Linux có chứa một bộ các quyền. Bộ quyền này xác định người dùng nào có quyền truy nhập và mức độ truy nhập đến chúng.
- Một tập tin hay một thư mục có các quyền đọc (Read), ghi (Write) và thi hành (Execute). Linux phân thành ba loại đối tượng truy nhập đến tập tin và thư mục:
  - Chủ nhân (owner): người tạo ra tập tin/ thư mục
  - Nhóm chủ nhân (group)
  - Các người dùng khác (other)

- Mỗi đối tượng có một bộ quyền read, write, execute của riêng mình. Ba bộ quyền read, write, execute ứng với ba đối tượng truy nhập owner, group và other tạo thành một tổ hợp ba nhóm quyền (có 9 kiểu).
  - Nhóm quyền đầu tiên điều khiển sự truy nhập của chủ nhân đến tập tin/thư mục của họ.
  - Nhóm quyền thứ hai điều khiển sự truy nhập của nhóm chủ nhân đến tập tin/thư mục.
  - Nhóm quyền thứ ba điều khiển sự truy nhập của tất cả các người dùng khác đến tập tin/thư mục.



- Một quyền trống (không gán quyền) được đại diện bởi ký tự **-**. Quyền đọc được biểu diễn bằng **r**, quyền ghi được đại diện bằng **w**, và quyền thi hành được biểu diễn bằng **x**.

- **Sticky bit**
  - Các quyền truy nhập thư mục của Linux thể hiện một điều là nếu một người dùng có quyền ghi vào một thư mục thì người dùng đó có thể đổi tên hay xóa các tập tin trong thư mục đó, ngay cả khi các tập tin không thuộc về người dùng đó.
  - Phương pháp để giải quyết vấn đề trên là thiết lập ticky bit cho thư mục. Khi một thư mục được gán ticky bit thì các thao tác đổi tên hay xóa tập tin trong thư mục đó chỉ có hiệu lực đối với chủ nhân của tập tin, chủ nhân của thư mục và người quản trị hệ thống.

- Lệnh **ls** của Linux hiển thị sticky bit là chữ '**T**' hoặc '**t**' trong trường thi hành (execute) của nhóm other.

```
[root@localhost root]# ls -li|tail -5|head -1
  3998 drwxr--r-T    2 root     root        4096 Oct 12 21:11 test
[root@localhost root]# _
```

15.2. Lệnh chmod

Dùng để thay đổi hay gán quyền trên tập tin, thư mục

- **Cú pháp**

  chmod [option] mode ...file...

- **Option**

  -R: Thay đổi quyền cả các thư mục con và các tập tin trong chúng.

  -v: Hiển thị thông điệp hệ thống khi thực hiện xử lý mỗi tập tin.

- Lệnh **chmod** thay đổi quyền của mỗi tập tin được liệt kê ứng với ***mode***
- Có hai cách để chỉ ra giá trị của đối số ***mode***. Cách thứ nhất là sử dụng các ký tự quyền gọi là phương pháp tượng trưng (**symbolic method**). Cách thứ hai là sử dụng mặt nạ (**binary mask**) còn gọi là phương pháp tuyệt đối (**absolute method**).

- **Phương pháp tượng trưng**
  - Định dạng của ***mode*** là
    [ugoa][+ - =] [rwxt]
  - Sự kết hợp của **u**,**g**, **o** hay **a** chỉ ra đối tượng được thay đổi quyền.
  - Các phép tóan **+ - =** xác định thao tác thêm bớt quyền.
  - Các ký tự **r, w**, **x** và **t** chỉ ra quyền cần thay đổi.

Ý nghĩa các ký tự:

- **u**: chủ nhân tập tin/thư mục
- **g**: group của tập tin/thư mục
- **o**: những người dùng khác không trong group
- **a**: Tất cả các người dùng
- **+**:Thêm quyền truy nhập tập tin/thư mục
- **-**:Lọai bớt quyền truy nhập
- **=**: Chỉ gán quyền được chỉ ra cho đối tượng, tất cả các quyền khác hiện có sẽ bị lọai bỏ.
- **r**, **w**, **x**: quyền read, write, execute
- **t**: Thay đổi sticky bit

- **Sử dụng phương pháp tuyệt đối**

  Phương pháp này thay đổi tất cả các quyền một lần bằng cách chỉ ra một mặt nạ (binary mask) tham chiếu đến tất cả các quyền trong mỗi đối tượng. Mặt nạ này tuân thủ theo định dạng nhị phân 8 bit-mỗi nhóm quyền được đại diện bởi một ký số bát phân. Khi chuyển về số nhị phân mỗi ký số bát phân trở thành ba ký số nhị phân, mỗi ký số nhị phân sẽ đại diện cho một quyền truy nhập của đối tượng

- **Bảng các ký số nhị phân và các quyền được gán tương ứng**

Số bát phân	Số nhị phân	Quyền
0	000	---
1	001	--x
2	010	-w-
3	011	-wr
4	100	r—
5	101	r-x
6	110	rw-
7	111	rwx

16. Soạn Thảo Văn Bản vi

16.1. Giới thiệu

- Trình soạn thảo văn bản chuẩn của Linux là **vi**.
- **vi** chạy ở hai chế độ khác nhau
  - Ở chế độ câu lệnh, những gì nhập vào sẽ được hiểu như là câu lệnh cho vi. Lệnh có thể là lưu tập tin, thóat khỏi vi, chuyển con trỏ đến các vị trí khác nhau trong tập tin, chỉnh sửa, thay thế đọan văn bản …
  - Ở chế độ nhập văn bản (chế độ INSERT), những gì nhập vào sẽ là nội dung của tập tin đang sọan thảo hay đang chỉnh sửa

- Để chuyển từ chế độ chỉnh sửa sang chế độ lệnh: **đánh phím ESC**.
- Để chuyển từ chế độ lệnh sang chế độ sọan thảo: **đánh phím Ins hay một chữ cái bất kỳ.**
  - Chú ý dòng trạng thái cuối màn hình, nếu có –INSERT– hay –REPLACE– thì đang ở chế độ sọan thảo, nếu là những dấu hiệu khác thì đang ở chế độ lệnh.

16.2.Chạy vi

vi filename

- **Filename:** là tên tập tin cần tạo hoặc các tập tin cần chỉnh sửa.
- Nếu vi được khởi động mà không có tên tập tin thì **vi** sẽ khởi động với bộ đệm (buffer) rỗng

16.3. Các lệnh cơ bản của vi

- **:help** Mở hướng dẫn sử dụng vi
- **:w** [*file*] Ghi lại nội dung tập tin. Nếu tập tin đang soạn thảo chưa có tên thì phải chỉ tận tập tin cần ghi là *file*
- **:q** Thóat khỏi vi;
- **:q!** Thóat khỏi vi và không ghi lại các thay đổi trên tập tin;
- **:qa!** Thóat khỏi vi mà không ghi lại các thay đổi trên tất cả tập tin đang mở.
- **:wq** Thực hiện ghi lại tập tin trước khi thóat.

- **:next** Chuyển tới tập tin kế tiếp trong trường hợp mở nhiều tập tin đồng thời.
- **:prev** Chuyển tới tập tin kế trước trong trường hợp mở nhiều tập tin đồng thời.
- **:e file** Đóng tập tin hiện hành và mở tập tin file.
- **:sh** Chuyển tạm sang shell để thi hành các lệnh của shell. Từ shell để trở lại vi thì đánh exit.

17. Trình tiện ích mc (midnight commander)