LINUX
CÁC LỆNH CƠ BẢN

1.Trình thông dịch lệnh....3
2.Kiểm tra có phải sử dụng bash shell không?....3
3.Phím tắt trong bash....3
4.Lệnh dưới Linux....4
5.Nhiều lệnh trên một dòng....4
6.Đường dẫn (path)....6
7.Đường dẫn hiện hành (current path)....6
8.Đường dẫn tuyệt đối (Absolute paths)....6
9.Đường dẫn tương đối (Relative paths)....6
10.Lệnh chuyển thư mục hiện hành "cd"....6
11.Sử dụng thư mục cha ".."....7
12.Một số ví dụ về đường dẫn tương đối....7
13.Thư mục hiện hành "."....7
14.Thư mục cá nhân (home directory)....7
15.Đường dẫn đến thư mục cá nhân của người dùng bất kỳ....7
16.Xem nội dung của một thư mục với lệnh ls....8
17.Xem thông tin về thư mục....8
18.Xem nội dung của tất cả các thư mục con một cách đệ qui....9
19.Inodes....9
20.Lệnh tạo thư mục mkdir....9
21.Lệnh tạo tập tin rỗng (touch)....10
22.Lệnh xem nội dung văn bản cat....11
23.Lệnh sao chép tập tin....11
24.Lệnh đổi tên hoặc di chuyển tập tin/thư mục mv....11
25.Liên kết cứng (Hard links)....12
26.Liên kết mềm (Symbolic links)....12
27.Lệnh xóa tập tin rm....13
28.Lệnh xóa thư mục rmdir....13
29.Sử dụng ký tự *....15
30.Sử dụng ký tự ?....15
31.Ký tự []....15
32.Ký tự [!]....16
33.Wildcard caveats....16

Chủ đề - 1 Truy cập vào hệ thống

Có hai chế độ đăng nhập: Đồ họa và văn bản.
Khi đăng nhập phải cung cấp username và password.
Vào chế độ đồ họa (graphic mode):

- Chế độ mặc định của workstation
- Ctrl-Fn-Alt-F7 (Laptop)
- Alt-F7 (Dessktop)

Vào chế độ văn bản (text mode)/dòng lệnh (command line mode):

- Chế độ mặc định của Server
- Ctrl-Fn-Alt-F1 -> Ctrl-Fn-Alt-F6 (Laptop)
- Alt-F1 -> Alt-F6 (Desktop)
- Mở một terminal trong chế độ đồ họa của máy trạm

Chủ đề - 2 Làm việc dưới chế độ dòng lệnh

1. Trình thông dịch lệnh

Dưới Linux, sau khi đăng nhập (login) vào hệ thống dưới chế độ văn bản (text) hoặc mở một cửa sổ lệnh (terminal) trong chế độ đồ họa (graphics) , bạn được chào đón bằng một **Dấu nhắc** (prompt) mời đánh lệnh như sau:

```
$
```

Dạng thức của **Dấu nhắc** có thể thay đổi tùy từng hệ thống. Nó có thể bao gồm cả tên của máy tính (hostname), thư mục hiện hành đang làm việc của bạn (current working directory). Bất kể dạng thức của dấu nhắc như thế nào, một điều chắc chắn là bạn đang giao tiếp với một chương trình được gọi là **Trình thông dịch lệnh** (shell).
Trình thông dịch lệnh được dùng phổ biến trên hệ thống Linux là chương trình **bash** (Viết tắt của cụm từ Bourne-Again Shell).

2. Kiểm tra có phải sử dụng bash shell không?

Bash là shell mặc định trên hầu hết các hệ điều hành Linux. Nhiệm vụ của một chương trình shell là thực hiện các lệnh do bạn nhập vào, giúp bạn tương tác được với hệ điều hành Linux.
Bạn có thể kiểm tra xem hệ thống bạn đang sử dụng có dùng bash shell không bằng lệnh sau:

```
$ echo $SHELL
/bin/bash
```

Nếu kết quả trả về không đúng như trên mà xuất hiện một thông báo lỗi, điều đó cho thấy hệ thống của bạn đang chạy một shell khác.
Bạn có thể thay đổi shell mặc định của hệ thống là bash shell bằng lệnh sau:

```
$chsh -s /bin/bash
```

Xem danh sách shell đang được cài đặt trong hệ điều hành Linux hiện tại bằng lệnh sau:

```
$cat /etc/shells
/bin/csh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/sh
/usr/bin/screen
/bin/dash
/bin/bash
/bin/rbash
/bin/false
```

Bạn có thể kết thúc chương trình shell khi đã hoàn thành phiên làm việc bằng lệnh **exit** hoặc **logout**, hay nhấn tổ hợp phím Control-D tại dấu nhắc của bash.

3. Phím tắt trong bash

- Ctrl+A: Di chuyển con trỏ về đầu dòng lệnh
- Ctrl+C: Kết thúc lệnh đang thực hiện
- Ctrl+D: Logout ra khỏi shell, tương đương với lệnh logout hoặc exit
- Ctrl+E: Di chuyển con trỏ về cuối dòng lệnh

- Ctrl+H: Xóa về phía trước 1 ký tự
- Ctrl+L: Xóa màn hình, tương đương lệnh clear
- Ctrl+R: Tìm lại các lệnh đã đánh
- Ctrl+Z: Tạm dừng chương trình
- Mũi tên trái, phải: Di chuyển con chuột về phía trái hoặc phải của dòng lệnh
- Mũi tên xuống hoặc lên: Di chuyển tới lui giữa các lệnh trong quá khứ
- Shift+PageUp và Shift+PageUp: Di chuyển lên xuống trong dữ liệu của màn hình
- Tab: Bổ sung hoàn chỉnh tên lệnh hay tập tin, thư mục

4. Lệnh dưới Linux

Cú pháp tổng quát:

```
$tên_lệnh tùy_chọn tham_số # chú thích được bỏ qua
```

Ví dụ:

```
$ls -l /etc
```

5. Nhiều lệnh trên một dòng

Dấu chấm phẩy ; để ngăn cách giữa các lệnh

Ví dụ:

```
$ echo line 1;echo line 2; echo line 3
line 1
line 2
line 3
```

Dấu && để yêu cầu thực hiện lệnh phía sau dấu && nếu lệnh phía trước thực thi thành công và trả về giá trị 0

Ví dụ:

```
$ echo line 1&&echo line 2&&echo line 3
line 1
line 2
line 3
```

Dấu || để yêu cầu thực hiện lệnh phía sau dấu && nếu lệnh phía trước thực thi không thahf công và trả về giá trị khác 0.

Chủ đề - 3 Hệ thống tập tin dưới Linux

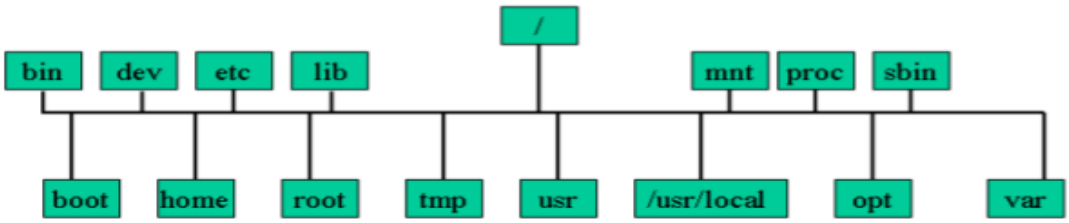
Dưới Linux/Unix, tất cả đề là tập tin (file). Có nhiều kiểu tập tin được ký hiệu như sau:

Ký hiệu	
-	Tập tin bình thường (văn bản, dữ liệu, chương trình, tập tin cấu hình)
d	Tập tin là một thư mục
l	Một liên kết
c	Tập tin là thiết bị vào ra kiểu ký tự, ví dụ bàn phím
s	Tập tin là một socket (nối kết mạng)
p	Tập tin là một ống dẫn pipe
b	Tập tin là một thiết bị ngoại vi dạng khối

Dùng lệnh ls -l để liệt kê các tập tin của một thư mục với ký tự đầu thể hiện loại tập tin

```
drwxr-xr-x  4 nbhung nbhung  4096 Dec 27 13:06 Softs
-rw-r--r--  1 nbhung nbhung  9091 Dec 27 14:22 svn.html
```

Tập tin dưới Linux được tổ chức dưới dạng một cây có một góc (/). Mỗi nút trên cây là một thư mục. Thư mục phía trên là thư mục cha, thư mục phía dưới là thư mục con.



/	Thư mục gốc
/boot	Các tập tin tĩnh cần thiết cho tiến trình khởi động
/dev	Các tập tin thiết bị
/etc	Các tập tinh cấu hình hệ thống và các ứng dụng
/lib	Các thư viện chia sẻ và các môdule của hạt nhân
/mnt	Điểm gắn nối các hệ thống tập tin một cách tạm thời
/opt	Nơi tích hợp các gói chương trình ứng dụng
/sbin	Các tập tin thực thi cần thiết cho hệ thống
/tmp	Nơi chứa các tập tin tạm
/usr	Hệ phân cấp thứ cấp
/var	Dữ liệu biến đổi
/home	Chứa thư mục cá nhân (home directory) của các người dùng

Chủ đề - 4 Các lệnh cơ bản của Linux

6. Đường dẫn (path)

Là một chuỗi các tên thư mục ngăn cách nhau bởi ký tự '**/**', kết thúc đường dẫn có thể là tên một tập tin. Đường dẫn dùng để xác định một tập tin hay một thư mục mà một lệnh cần thao tác.

7. Đường dẫn hiện hành (current path)

Tại một thời điểm, người dùng xem như đang ở tại một thư mục nào đó trên cây thư mục. Thực mục này được gọi là **thư mục hiện hành**. Để xem thư mục làm việc hiện hành của bạn, hãy nhập vào lệnh sau:

```
$ pwd
/
```

Lệnh trên cho thấy bạn đang ở thư mục gốc (ký hiệu bằng ký tự '/') của cây thư mục.
Có hai loại đường dẫn: Đường dẫn tuyệt đối và đường dẫn tương đối.

8. Đường dẫn tuyệt đối (Absolute paths)

Đường dẫn tuyệt đối được bắt đầu bằng ký tự /. Ví dụ:

```
/dev
/usr
/usr/bin
/usr/local/bin
```

Bạn có thể chuyển vào thư mục /user/loal/bin bằng lệnh sau, từ bất kỳ thư mục hiện hành nào:

```
cd /usr/local/bin
```

9. Đường dẫn tương đối (Relative paths)

Đường dẫn tương đối được tính toán với điểm bắt đầu của đường dẫn được ngầm định là thư mục hiện hành của bạn. Một đường dẫn tương đối không bao giờ bắt đầu với ký tự '/'.
Ví dụ:
Để chuyển vào thư mục /usr bạn sử dụng lệnh sau:

```
$ cd /usr
```

Lúc này thư mục hiện hành của bạn là /usr. Ở đó bạn có thể chuyển vào thư mục /usr/local/bin bằng cách sử dụng đường dẫn tương đối như sau:

```
$ cd local/bin
$ pwd
/usr/local/bin
```

10. Lệnh chuyển thư mục hiện hành "cd"

Mục đích: Lệnh **cd** cho phép bạn di chuyển thư mục làm việc hiện hành đến những điểm khác nhau trên cây thư mục.
Cú pháp tổng quát:

```
$cd  path
```

Trong đó `path` là đường dẫn (tương đối hoặc tuyệt đối) chỉ đến thư mục mà ta muốn chuyển đến.
Ví dụ: Chuyển về thư mục gốc của cây thư mục ta đánh lệnh:

```
$ cd /
```

11. Sử dụng thư mục cha ".."

Thư mục cha của thư mục hiện hành được ký hiệu bằng chuỗi hai dấu chấm "..". Hãy thực hiện các lệnh sau:

```
$ pwd
/usr/local/bin
$ cd ..
$ pwd
/usr/local
```

Như vậy, thư mục hiện hành hiện nay là /usr/local.
Bạn có thể sử dụng thư mục cha trong đường dẫn tương đối, ví dụ:

```
$ pwd
/usr/local
$ cd ../share
$ pwd
/usr/share
```

12. Một số ví dụ về đường dẫn tương đối

```
$ cd /bin
$ cd ../usr/share/zoneinfo
$ cd /usr/lib/jvm
$ cd ../lib/X11
$ cd /usr/bin
$ cd ../bin/../bin
```

13. Thư mục hiện hành "."

Thư mục hiện hành được ký hiệu bởi dấu chấm ".". Thư mục hiện hành thường được dùng để yêu cầu shell thực hiện một chương trình nào đó trong thư mục hiện hành.
Ví dụ:

```
$ echo "echo Hello world" > myprog.sh
$ chmod 755 myprog.sh
$ ./myprog.sh
```

Ví dụ cuối cùng yêu cầu shell thực thi chương trình myprog.sh đang nằm trong thư mục hiện hành của bạn.

14. Thư mục cá nhân (home directory)

Mỗi người dùng có một thư mục cá nhân để lưu trữ thông tin riêng của họ (gọi là home directory) nơi mà họ có toàn quyền trên đó trong khi những người khác thì không có quyền truy xuất đến. Bạn có thể chuyển về home directory của bạn từ bất kỳ vị trí nào trong cây thư mục bằng cách đánh lệnh cd không tham số:

```
$ cd
```

Khi đó bạn sẽ được chuyển về thư mục /home/username với username là tên đăng nhập của bạn.
Thư mục cá nhân được ký hiệu bằng ký tự '~'. Ví dụ ta thực hiện chương trình myprog trong thư mục hiện hành, với tham số là myfile.txt ở thư mục cá nhân bằng lệnh sau:

```
$ ./myprog ~/myfile.txt
```

15. Đường dẫn đến thư mục cá nhân của người dùng bất kỳ

Bash dùng ký hiệu '**~**' để chỉ đến thư mục các nhân của bạn. Tuy nhiên bạn cũng có thể dùng nó để chỉ đến thư mục cá nhân của những người dùng khác. Ví dụ, bạn muốn tham khảo tập tin fredsfile.txt trong thư mục cá nhân của người dùng fred, bạn nhập vào dòng lệnh:

```
$ ./myprog ~fred/fredsfile.txt
```

16. Xem nội dung của một thư mục với lệnh ls

Cú pháp của lệnh như sau:

```
$ls [options] [path-to-dir]
```

Trong đó:

- options là các tùy chọn, path là đường dẫn (tuyệt đối/tương đối) đến một tập tin hay thư mục muốn xem thông tin/nội dung
- Nếu không có path sẽ xem thư mục hiện hành, ví dụ

```
$ cd /usr
$ ls
X11R6 doc i686-pc-linux-gnu lib man sbin ssl
bin gentoo-x86 include libexec portage share tmp
distfiles i686-linux info local portage.old src
```

- Có mô tả đường dẫn

```
$cd ~
$ls /usr
X11R6 doc i686-pc-linux-gnu lib man sbin ssl
bin gentoo-x86 include libexec portage share tmp
distfiles i686-linux info local portage.old src
```

- Nếu có thêm tham số -a, lệnh ls sẽ liệt kê luôn cả các tập tin ở dạng ẩn (tên tập tin bắt đầu bằng dấu chấm **.**):

```
$ ls -a ~
.gnashpluginrc          .purple                 .xxe4
.gnome                  .pykaraoke              .yahoorc
.gnome2                 .qt                     .yEd
```

- Nếu tham số **-l** được đưa vào, lệnh ls sẽ liệt kê nhiều thông tin hơn về các tập tin và thư mục nằm trong thư mục được mô tả, gồm các thông tin như: quyền truy cập tập tin, chủ sở hữu , ngày giờ cập nhật sau cùng, kích thước tập tin. Ví dụ

```
$ ls -l /usr
drwxr-xr-x 7 root root 168 Nov 24 14:02 X11R6
drwxr-xr-x 2 root root 14576 Dec 27 08:56 bin
drwxr-xr-x 2 root root 8856 Dec 26 12:47 distfiles
lrwxrwxrwx 1 root root 9 Dec 22 20:57 doc -> share/doc
drwxr-xr-x 62 root root 1856 Dec 27 15:54 gentoo-x86
lrwxrwxrwx 1 root root 10 Dec 22 20:57 tmp -> ../var/tmp
```

  Cột thứ nhất mô tả quyền truy cập đối tượng (tập tin hay thư mục)
  Cột thứ hai mô tả số lượng các nối kết đến tập đối tượng
  Cột thứ ba và tư mô tả chủ sở hữu và nhóm chủ sở hữu đối tượng.
  Cột thứ năm mô tả kích thước đối tượng.
  Cột thứ sáu mô tả ngày, giờ cập nhật đối tượng sau cùng.
  Cuối cùng là tên của đối tượng.
  Nếu tên tập tin là một liên kết thì nó sẽ có dấu mũi tên "->" để chỉ đến đối tượng mà nó liên kết tới.

17. Xem thông tin về thư mục

Để chỉ xem thông tin về thư mục mà không cần thiết phải liệt kê nội dung của thư mục ta dùng lệnh ls với tham số là **–dl**. Ví dụ:

```
$ ls -dl /usr  /usr/bin   /usr/X11R6/bin ../share
drwxr-xr-x 4 root root 96 Dec 18 18:17 ../share
drwxr-xr-x 17 root root 576 Dec 24 09:03 /usr
```

```
drwxr-xr-x 2 root root 3192 Dec 26 12:52 /usr/X11R6/bin
drwxr-xr-x 2 root root 14576 Dec 27 08:56 /usr/bin
```

18. Xem nội dung của tất cả các thư mục con một cách đệ qui

Sử dụng lệnh ls với tham số -R để xem nội dung của tất cả các thư mục con, của một thư mục.

```
$ls -R ~
```

19. Inodes

Mỗi đối tượng trong hệ thống tập tin của Linux được gán một số chỉ mục (index) duy nhất, được gọi là số inode. Bạn có thể xem thông tin về số inode của một tập tin hay thư mục bằng lệnh ls với tham số là –i. Ví dụ:

```
$ ls -id /usr/local
5120 /usr/local
```

Kết quả lệnh trên cho thấy thư mục /usr/local có số inode là 5120.
Tiếp tục xem inode của thư mục /usr/local/bin/.. bằng lệnh sau:

```
$ ls -id /usr/local/bin/..
5120 /usr/local/bin/..
```

Ta thấy, /usr/local/bin/.. có cùng số inode với thư mục /usr/local. Điều đố có vẻ nghịch lý vì theo nguyên tắc mỗi thư mục hay tập tin được gán một inode riêng. Đây là cơ chế liên kết (link) dưới Linux. Thực tế chỉ tồn tại một thư mục có số inode là 5120. Cả hai thư mục /usr/local và /usr/local/bin/.. đều liên kết đến cùng một thư mục có số inode là 5120. Như vậy cả hai thư mục trên có cùng một nội dung.
Dùng lệnh ls với tham số -dl để hiển thị thông tin chi tiết về một thư mục:

```
$ ls -dl /usr/local
drwxr-xr-x 8 root root 240 Dec 22 20:57 /usr/local
```

Tại cột thứ hai từ bên trái tính sang ta thấy có số 8 biểu thị rằng có 8 thư mục cùng liên kết đến inode số 5120. Trong hệ thống hiện tại là các thư mục sau (Có thể khác với hệ thống bạn đang sử dụng):
/usr/local
/usr/local/.
/usr/local/bin/..
/usr/local/games/..
/usr/local/lib/..
/usr/local/sbin/..
/usr/local/share/..
/usr/local/src/..

20. Lệnh tạo thư mục mkdir

Lệnh **mkdir** cho phép tạo một thư mục mới trong hệ thống tập tin. Cú pháp tổng quát như sau:

```
$mkdir [options]  path-new-dir1 path-new-dir2 path-new-dir3
```

Ví dụ sau tạo ra 3 thư mục mới có tên là tic, tac và toe, trong thư mục /tmp:

```
$ cd /tmp
$ mkdir tic tac toe
```

Đánh lệnh `$ls /tmp` để kiểm tra xem hệ thống có tạo ra các thư mục tic, tac và toe không.
Mặc định, lệnh **mkdir** không tạo thư mục cha của thư mục mới khi chưa có. Vì thế nếu bạn muốn tạo thư mục `won/der/ful` bằng lệnh sau:

```
$ mkdir won/der/ful
```

Thì sẽ bị báo lỗi nếu một trong các thư mục won hay der chưa tồn tại:

```
mkdir: cannot create directory `won/der/ful': No such file or directory
```

Bạn phải thực hiện lần lượt các lệnh sau:

```
$ mkdir won
$ mkdir won/der
$ mkdir won/der/ful
```

Tuy nhiên bạn có thể yêu cầu lệnh mkdir tạo tất cả các thư mục cha chưa tồn tại bằng cách đưa thêm tùy chọn **-p**. Ví dụ dưới đây sẽ tạo ra 3 thư mục mới `easy`, `as` và `pie` theo thứ tự `easy/as/pie`:

```
$ mkdir -p easy/as/pie
```

21. Lệnh tạo tập tin rỗng (touch)

Cú pháp: `touch fileName`

Nếu tập tin đã tồn tại, touch sẽ cập nhật lại thời gian cập nhật sau cùng của tập tin là thời điểm thực hiện lệnh touch. Ngược lại, nếu tập tin chưa tồn tại, một tập tin rỗng được tạo ra.
Ví dụ tạo file copyme trong thư mục /tmp:

```
$ cd /tmp
$ touch copyme
```

Dùng lệnh `ls –l` để xem lại thông tin về tập tin vừa được tạo ra. Chú ý đến ngày cập nhật sau cùng và kích thước của tập tin (Kích thước là 0 bytes).
Sau đó đánh lại lệnh touch:

```
$ touch copyme
```

Dùng lệnh ls –l để kiểm tra ngày cập nhật sau cùng của tập tin copyme đã được thay đổi so với lần trước chưa.

Chủ đề - 5 Hiển thị thông tin và định hướng lại

Lệnh `echo` dùng để xuất dữ liệu ra thiết bị xuất chuẩn (mặc định là màn hình).
Cú pháp: `echo "Data to export to standard ouput".`
Ví dụ:

```
$ echo "firstfile"
firstfile
```

Tuy nhiên ta có thể chuyển hướng dữ liệu xuất ra các thiết bị xuất khác thiết bị xuất chuẩn bằng cách sử dụng cơ chế định hướng lại.
Cú pháp: `command > newOutput`
Ví du sau cho lệnh echo xuất dữ liệu ra tập tin `copyme`:

```
$ echo "firstfile" > copyme
```

Dùng lệnh ls để kiểm tra lại kích thước của tập tin copyme

```
$ ls -l copyme
-rw-r--r-- 1 root root 10 Dec 28 14:13 copyme
```

22. Lệnh xem nội dung văn bản cat

Để hiển thị nội dung của một tập tin trên cửa sổ lệnh ta dùng lệnh **cat**.
Cú pháp: cat fileName

```
$ cat copyme
firstfile
```

23. Lệnh sao chép tập tin

Để sao chép một tập tin mới từ một tập tin đã có ta dùng lệnh **cp.**
Cú pháp: cp oldFile newFile
Ví dụ, chép tập tin mới copiedme từ tập tin copyme:

```
$ cp copyme copiedme
```

Đây thật sự là 2 tập tin khác nhau với có số inode riêng biệt. Ta có thể kiểm tra bằng lệnh ls:

```
$ ls -i copyme copiedme
648284 copiedme 650704 copyme
```

24. Lệnh đổi tên hoặc di chuyển tập tin/thư mục mv

Lệnh mv có cú pháp tổng quát như sau: `$mv path-to-source path-to-destination`

- path-to-source là đường dẫn đến tập tin nguồn cần đổi tên hay di chuyển
- path-to-destination là tên mới hoặc là thư mục sẽ di chuyển tập tin nguồn đến
  - Nếu path-to-destination là một thư mục đã tồn tại thì lệnh mv sẽ di chuyển tập tin nguồn path-to-source vào thư mục path-to-destination
  - Nếu path-to-destination là tên mới thì lệnh mv sẽ đổi tên tập tin nguồn path-to-source thành tên mới path-to-destination. Số hiệu inode của tập tin nguồn vẫn được giữ lại.

Ví dụ:

```
$ls -i copiedme
648284 movedme
$mv copiedme movedme
$ls -i movedme
648284 movedme
$mv movedme /tmp
$ls -i movedme
ls: cannot access movedme: No such file or directory
$ls -i /tmp/movedme
648284 movedme
```

Chủ đề - 6 Tạo các liên kết và xóa các tập tin

25. Liên kết cứng (Hard links)

Có hai loại liên kết trên Linux: Liên kết cứng và liên kết mềm. Một inode có thể có nhiều liên kết cứng nối với nó và inode sẽ được tồn tại đến khi nào tất cả các liên kết cứng nối với nó không còn nữa. Một liên kết mới có thể được tạo ra bằng cách sử dụng lệnh **ln** .
Cú pháp : ln fileName newLink
Ví dụ:

```
$ cd /tmp
$ touch firstlink
$ ln firstlink secondlink
$ ls -i firstlink secondlink
15782 firstlink 15782 secondlink
```

Hạn chế của Liên kết cứng là nó chỉ cho phép tạo các liên kết cứng đến các inode của tập tin. Liên kết cứng không cho phép chúng ta tạo liên kết cho thư mục và không cho phép mở rộng hệ thống tập tin trên nhiều ổ đĩa. Có nghĩa là bạn không thể tạo một liên kết từ /usr/bin/bash đến /bin/bash nếu thư mục gốc của bạn / và thư mục /usr tồn tại trên 2 hệ thống tập tin riêng biệt.

```
$cd /tmp
$ln /usr/local/bin bin1
$ls -l bin1
ln: `/usr/local/bin': hard link not allowed for directory
```

26. Liên kết mềm (Symbolic links)

Trong thực tế liên kết mềm thường được dùng nhiều hơn liên kết cứng. Liên kết mềm là một kiểu tập tin đặc biệt mà ở đó liên kết tham khảo đến tập tin khác bằng tên thay vì tham khảo trực tiếp đến inode. Liên kết mềm không ngăn ngừa được trường hợp tập tin mà nó tham khảo đến đã bị xóa. Nếu tập tin đích không còn tồn tại, liên kết mềm xem như không còn sử dụng được hay hoàn toàn bị đỗ vỡ.
Một liên kết mềm có thể được tạo ra bằng lệnh **ln** với tùy chịn là **–s**.
Cú pháp: ln –s fileName newLink
Ví dụ:

```
$ ln -s secondlink thirdlink
$ ls -l firstlink secondlink thirdlink
-rw-rw-r-- 2 agriffis agriffis 0 Dec 31 19:08 firstlink
-rw-rw-r-- 2 agriffis agriffis 0 Dec 31 19:08 secondlink
lrwxrwxrwx 1 agriffis agriffis 10 Dec 31 19:39 thirdlink -> secondlink
```

Liên kết mềm có thể phân biệt với các tập tin thường trong lệnh **ls -l** ở 3 đặc điểm trong cửa sổ kết quả:
Cột thứ nhất có chứa ký tự **l** để báo hiệu đó là một liên kết mềm.
Kích thước của liên kết mềm thì bằng với kích thước của tên tập tin đích (trong trường hợp này là tập tin **secondlink**).
Cột cuối cùng có hiển thị tên tập tin mà liên kết mềm trỏ đến.
Liên kết mềm thì mềm dẽo hơn liên kết cứng rất nhiều. Bạn có thể tạo một liên kết mềm đến bất kỳ kiểu đối tượng nào của hệ thống tập tin, bao gồm cả thư mục. Nó có thể trỏ đến một đối tượng trên một hệ thống tập tin khác.
Xét trường hợp mà ở đó ta muốn tạo một liên kết trong thư mục /tmp trỏ đến thư mục /usr/local/bin.
Hãy nhập vào các lệnh sau:

```
$cd /tmp
$ ln -s /usr/local/bin bin1
```

```
$ ls -l bin1
lrwxrwxrwx 1 root root 14 Jan 1 15:42 bin1 -> /usr/local/bin
```

Tương tự

```
$ ln -s ../usr/local/bin bin2
$ ls -l bin2
lrwxrwxrwx 1 root root 16 Jan 1 15:43 bin2 -> ../usr/local/bin
```

Ta thấy cả 2 liên kết mềm đều trỏ đến cùng một thư mục đích. Tuy nhiên nếu liên kết thứ hai được chuyển sang một thư mục khác thì nó sẽ vị gẫy ngay vì địa chỉ của tập tin đích được mô tả theo kiểu đường dẫn tương đối.

Hãy kiểm tra bằng các kệnh sau:

```
$ ls -l bin2
lrwxrwxrwx 1 root root 16 Jan 1 15:43 bin2 -> ../usr/local/bin
$ mkdir mynewdir
$ mv bin2 mynewdir
$ cd mynewdir
$ cd bin2
bash: cd: bin2: No such file or directory
```

Bởi vì thư mục `/tmp/usr/local/bin` không tồn tại, nên chúng ta không thể chuyển vào thư mục bin2, có nghĩa là liên kết bin2 đã bị "gẫy"

Vì lý do này thông thường người ta không dùng đường dẫn tương đối khi tạo liên kết mềm. Tuy nhiên cũng có trường hợp sử dụng đường dẫn tương đối trong liên kết mềm thì hợp lý hơn.

Xem ví dụ dưới đây, ở đó ta muốn tạo một tên mới cho một chương trình đang nằm trong thư mục /usr/bin:

```
$ ls -l /usr/bin/keychain
-rwxr-xr-x 1 root root 10150 Dec 12 20:09 /usr/bin/keychain
$ cd /usr/bin
$ ln -s /usr/bin/keychain kc
```

Giải pháp trên là đúng đắng, tuy nhiên sẽ có vấn đề ngay khi bạn quyết định chuyển cả hai tập tin vào thư mục /usr/local/bin:

```
$ mv /usr/bin/keychain /usr/bin/kc /usr/local/bin
```

Bởi vì sử dụng đường dẫn tuyệt đối trong liên kết mềm, **kc** vẫn tiếp tục trỏ đến /usr/bin/keychain, mà hiện tại nó không còn nữa. Như thế liên kết đã bị gẫy.

Cả đường dẫn tuyệt đối và tương đối sử dụng trong liên kết mềm đều có những điểm bất lợi riêng. Tùy tường trường hợp mà bạn chọn lựa loại đường dẫn cho phù hợp.

27. Lệnh xóa tập tin rm

Lệnh **rm** cho phép bạn xóa một tập tin ra khỏi hệ thống tập tin.

Cú pháp: rm fileName

Ví dụ:

```
$ cd /tmp
$ touch file1 file2
$ ls -l file1 file2
-rw-r--r-- 1 root root 0 Jan 1 16:41 file1
-rw-r--r-- 1 root root 0 Jan 1 16:41 file2
$ rm file1 file2
$ ls -l file1 file2
ls: file1: No such file or directory
ls: file2: No such file or directory
```

28. Lệnh xóa thư mục rmdir

Để xóa thư mục bạn có hai cách. Cách thứ nhất là xóa tất cả các đối tượng bên trong thư mục muốn xóa (làm cho thư mục trở nên rỗng). Sau đó dùng lệnh rmdir để xóa nó.

Ví dụ:

```
$ mkdir mydir
$ touch mydir/file1
$ rm mydir/file1
$ rmdir mydir
```

Cách thứ hai là sử dụng lệnh **rm** với tùy chọn **–rf**.

Ví dụ

```
$ rm -rf mydir
```

Chủ đề - 7 Sử dụng ký tự đại diện (wildcards)

Trong khi sử dụng Linux, đôi khi bạn muốn thực hiện một tác vụ (ví dụ **rm**) trên nhiều đối tượng của hệ thống tập tin trên cùng một lượt. Trong trường hợp đó ta có thể dùng lệnh với tham số gồm nhiều tên tập tin.
Ví dụ:

```
$ rm file1 file2 file3 file4 file5 file6 file7 file8
```

Để giải quyết vấn đề này, bạn có thể sử dụng tiện ích hỗ trợ sẵn của Linux là các ký tự đại điện. Nó cho phép bạn chỉ ra nhiều tập tin một lượt bằng cách sử dụng một mẫu đại diện (*wildcard pattern)*. Trình thông dịch lệnh Bash và các trình thông dịch lệnh khác sẽ thông dịch mẫu bằng cách dò tìm trên đĩa những tập tin mà nó trùng khớp với mẫu được mô tả
Giả sử bạn có các tập tin file1 đến file8 trong thư mục hiện hành, bạn có thể xóa cả 8 file này bằng lệnh sau:

```
$ rm file[1-8]
```

Hay đơn giản hơn có thể mô tả rằng bạn muốn xóa tất cả các tập tin có tên mở đầu bằng file, hãy đánh lệnh sau:

```
$ rm file*
```

Hoặc bạn muốn liệt kê tất cả các đối tượng trong hệ thống tập tin trong thư mục /etc có tên bắt đầu với ký tự **g**, bạn có thể gỏ lệnh sau:

```
$ ls -d /etc/g*
/etc/gconf /etc/ggi /etc/gimp /etc/gnome /etc/gnome-vfs-mime-magic /etc/gpm
```

Điều gì sẽ xảy ra nếu bạn mô tả một mẫu mà nó không trùng với bất kỳ đối tượng nào nằm trong hệ thống tập tin? Trong ví dụ sau, chúng muốn liệt kê tất cả các thư mục có trong thư mục /usr/bin mà chúng có tên bắt đầu với **asdf** và kết thúc với **jkl**:

```
$ ls -d /usr/bin/asdf*jkl
ls: /usr/bin/asdf*jkl: No such file or directory
```

Trong trường hợp này kết quả là một thông báo lỗi. Như vậy, với bash thì kết quả của các mẫu phải có ít nhất một trường hợp trùng nếu không lệnh thực hiện xem như bị lỗi.

29. Sử dụng ký tự *

Ký tự * đại diện cho 0 hoặc nhiều ký tự bất kỳ nào đó. Có nghĩa là “bất cứ cái gì ở đây”. Ví dụ:
/etc/g* sẽ trùng với tất cả các tập tin trong thư mục /etc mà chúng có tên bắt đầu bằng ký tự **g**.
/tmp/my*1 trùng với tất cả các tập tin trong thư mục /tmp mà chúng có tên bắt đầu với **my** và kết thúc với **1**.

30. Sử dụng ký tự ?

Ký tự **?** đại diện cho bất kỳ **một** ký tự nào đó.
Ví dụ:
myfile? sẽ trùng với tất cả các tập tin mà tên bao gồm **myfile** và theo sau bằng một ký tự bất kỳ
/tmp/notes?txt sẽ trùng với cả hai tập tin **/tmp/notes.txt** và **/tmp/notes_txt**, nếu chúng tồn tại.

31. Ký tự []

Ký tự này cũng giống như ký tự **?**, nhưng có một số điểm đặc biệt hơn. Để sử dụng ký tự này, hãy để bất kỳ ký tự nào mà bạn muốn trùng khớp vào bên trong **[]**. Kết quả của biểu thức sẽ là một ký tự mà nó trùng khớp với bất kỳ một ký tự nào trong cặp **[]** Bạn cũng có thể dùng dấu **–** để mô tả khoảng ký tự sẽ trùng khớp.

Ví dụ:
myfile[12] sẽ trùng khớp với myfile1 và myfile2.
[Cc]hange[Ll]og sẽ trùng khớp với Changelog, ChangeLog, changeLog, và changelog.
ls /etc/[0-9]* sẽ liệt kê tất cả các tập tin trong thư mục /etc mà chúng bắt đầu với một số.
ls /tmp/[A-Za-z]* sẽ liệt kê tất cả các tập tin trong thư mục /etc mà chúng bắt đầu với một ký tự hoa hay thường.

32. Ký tự [!]

Ký tự **[!]** tương tự như cấu trúc **[]** , ngoại trừ thay vì trùng khớp bất kỳ ký tự nào nằm trong cặp dâu [], nó sẽ trùng khớp với bất kỳ ký tự nào mà **không** được liệt kê giữa **[!** và **]**.
Ví dụ:
rm myfile[!9] sẽ xóa tất cả các tập tin có tên bắt đầu là **myfile** theo sau là một ký tự không phải là ký tự 9 (tức ngoại trừ tập tin **myfile9**).

33. Wildcard caveats

Bởi vì **bash** xử lý các ký tự đại diện (**?**, **[**, **]**, *****) một cách đặc biệt, bạn cần cẩn thận khi nhập đối số của các lệnh có chứa các ký tự đặc biệt này. Ví dụ nếu bạn muốn tạo một tập tin có chứa chuỗi **[fo]***, lệnh sau đây có thể thực hiện công việc mà bạn không trong đợi:

```
$ echo [fo]* > /tmp/mynewfile.txt
```

Nếu mẫu **[fo]*** trùng với bất kỳ tập tin nào trong thư mục hiện hành, khi đó bạn sẽ thấy tên của các tập tin này trong nội dung của tập tin /tmp/mynewfile.txt hơn là chuỗi **[fo]*** như mong đợi. Có thể khắc phục tình trạng trên bằng cách bao xung quanh các ký tự đặc biệt dấu nhấy đơn ’ để bảo với **bash** xử lý các ký tự đặc biệt như những ký tự thông thường:

```
$ echo '[fo]*' > /tmp/mynewfile.txt
```

Trong ví dụ trên, tập tin mới tạo ra sẽ chứa chuỗi **[fo]*** như ta mong đợi.
Hoặc có thể dùng ký tự ‘\’ đi liền phía trước ký tự đặc biệt để yêu cầu **bash** xử lý chúng như các ký tự bình thường:

```
$ echo \[fo\]\* > /tmp/mynewfile.txt
```

Cả hai giải pháp trên đều hoạt động rất tốt.