LINUX
HỆ THỐNG TẬP TIN

1 Chuẩn phân cấp hệ thống tập tin (FHS-Filesystem Hierarchy Standard)

Chuẩn phân cấp hệ thống tập tin là một tài liệu mô tả cách sắp xếp các thư mục trên hệ thống Linux. FHS được phát triển để cấp một khuôn mẫu chung nhằm giúp cho việc phát triển các ứng dụng mà không phụ thuộc vào bản phân phối Linux. FHS mô tả các thư mục sau:
* `/` : 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

1.1.1 Hai kiểu FHS độc lập

FHS mô tả về khuôn mẫu các thư mục của nó dựa trên ý tưởng về các tập tin : có thể chia sẻ (shareable) hay không thể chia sẻ (unshareable), và biến đổi (variable) hay tĩnh (static). Các dữ liệu có thể chia sẻ có thể được chia sẻ giữa các máy tính với nhau; dữ liệu không thể chia sẻ chỉ sử dụng cho từng máy riêng biệt (ví dụ như các tập tin cấu hình). Dữ liệu biến đổi có thể được thay đổi, điều chỉnh; dữ liệu tĩnh thì không cho phép thay đổi. Bảng sau mô tả việc phân loại các thư mục trong FHS

	shareable	**unshareable**
static	`/usr` `/opt`	`/etc` `/boot`
variable	`/var/mail` `/var/spool/news`	`/var/run` `/var/lock`

1.1.2 Cấu trúc phân cấp thứ cấp trên thư mục /usr

Dưới thư mục `/usr` bạn sẽ tìm thấy một cấu trúc phân cấp thứ hai giống như hệ thống tập tin gốc. Không bắt buộc thư mục `/usr` tồn tại khi máy tính được khởi động mà nó có thể được chia sẻ từ mạng ("shareable") hay nối kết vào từ CD-ROM ("static") . Hầu hết các chương trình cài đặt Linux không ứng dụng hình thức chia sẻ thư mục `/usr`, nhưng nó cũng rất đáng để hiểu sự hữu ích của việc phân biệt giữa cấu trúc phân cấp chính tại thư mục gốc và cấu trúc phân cấp phụ tại thư mục `/usr`.

2 Tìm kiếm tập tin

Hệ thống Linux thường chứa hàng trăm ngàn tập tin. Linux hỗ trợ nhiều công cụ khác nhau để giúp bạn tìm ra một tập tin nào đó.

1.1.3 Biến môi trường PATH

Khi bạn thực thi một chương trình tại dòng lệnh, chương trình thông dịch lệnh bash sẽ tìm kiếm chương trình trong danh sách các thư mục đã được mô tả trong biến môi trường PATH. Ví dụ, khi bạn đánh lệnh `ls`, bash không hiểu ngay là chương trình `ls` nằm trong thư mục `/usr/bin`. Thay vào đó, bash tham khảo đến biến môi trường có tên là `PATH`, nó là một danh sách các thư mục được phân cách bởi dấu hai chấm '**:**'. Chúng ta có thể khảo sát giá trị của PATH:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11R6/bin
```

Với giá trị của `PATH` như trên, để tìm chương trình ls, bash trước tiên sẽ kiểm tra thư mục `/usr/local/bin`, kế đến là thư mục `/usr/bin`. Thông thường , `ls` được đặt trong thư mục `/usr/bin`, vì thế bash sẽ dừng lại ở tại điểm này.

1.1.4 Thay đổi PATH

Bạn có thể mở rộng thêm biến `PATH` bằng cách gán thêm các phần tử vào nó với lệnh sau:

```
$ PATH=$PATH:~/bin
$ echo $PATH
/
usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11R6/bin:/home/agriffis/bi
n
```

Bạn cũng có thể xóa bỏ một phần tử ra khỏi `PATH`, mặc dù đó không phải là một công việc đơn giản bởi vì bạn không thể tham khảo đến giá trị đang tồn tại của `$PATH`. Cách tốt nhất là đánh lại nội dung mới hoàn toàn cho biến `PATH`:

```
$ PATH=/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:~/bin
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/agriffis/bin
```

1.1.5 Lệnh "which"

Bạn có thể kiểm tra một chương trình nào đó có nằm trong các thư mục được chỉ ra bởi `PATH` hay không bằng cách dùng lệnh `which`. Ví dụ, ở đây ta sẽ thấy rằng trong hệ thống Linux hiện tai không có chương trình tên sense:

```
$ which sense
which: no sense in
(/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11R6/bin)
```

Trường hợp khác ta lại thành công khi tìm lệnh `ls`:

```
$ which ls
/usr/bin/ls
```

1.1.6 Sử dụng "which -a"

Dùng cờ `-a` để yêu cầu `which` hiển thị tất cả các chương trình được yêu cầu có trong các thư mục được mô tả bởi PATH:

```
$ which -a ls
/usr/bin/ls
/bin/ls
```

1.1.7 Lệnh whereis

Nếu bạn muốn tìm nhiều thông tin hơn ngòai vị trí của một chương trình bạn có thể dùng chương trình `whereis` :

```
$ whereis ls
ls: /bin/ls /usr/bin/ls /usr/share/man/man1/ls.1.gz
```

Ta thấy rằng that `ls` có mặt trong hai nơi là `/bin` và `/usr/bin`. Hơn thế chúng ta còn được thông báo rằng có tài liệu hướng dẫn sử dụng nằm trong thư mục `/usr/share/man`.

1.1.8 Lệnh find

Lệnh `find` là một tiện ích khác cho phép bạn tìm kiếm các tập tin. Với `find` bạn không bị giới hạn trong phạm vi tìm kiếm chương trình, bạn có thể tìm kiếm một tập tin bất kỳ mà bạn muốn bằng cách sử dụng một tiêu chuẩn tìm kiếm nào đó. Ví dụ, để tìm một tập tin có tên README, bắt đầu từ thư mục `/usr/share/doc` ta thực hiện lệnh sau:

```
$ find /usr/share/doc -name README
/usr/share/doc/ion-20010523/README
/usr/share/doc/bind-9.1.3-r6/dhcp-dynamic-dns-examples/README
/usr/share/doc/sane-1.0.5/README
```

1.1.9 Sử dụng các ký tự đại diện với lệnh find

Bạn có thể sử dụng các ký tự đại diện trong tham số `-name`, mà bạn bao bọc nó bằng cặp nháy đơn hay đặt ký tự \ phía trước ký tự đại diện đó. Ví dụ , chúng ta muốn tìm tập tin README với các phần mở rộng khác nhau như sau:

```
$ find /usr/share/doc -name README\*
/usr/share/doc/iproute2-2.4.7/README.gz
/usr/share/doc/iproute2-2.4.7/README.iproute2+tc.gz
/usr/share/doc/iproute2-2.4.7/README.decnet.gz
/usr/share/doc/iproute2-2.4.7/examples/diffserv/README.gz
/usr/share/doc/pilot-link-0.9.6-r2/README.gz
/usr/share/doc/gnome-pilot-conduits-0.8/README.gz
/usr/share/doc/gimp-1.2.2/README.i18n.gz
/usr/share/doc/gimp-1.2.2/README.win32.gz
/usr/share/doc/gimp-1.2.2/README.gz
/usr/share/doc/gimp-1.2.2/README.perl.gz
[578 additional lines snipped]
```

1.1.10 Tìm không phân biệt chữ hoa và chữ thường với find

Ta có thể dùng lệnh sau:

```
$ find /usr/share/doc -name '[Rr][Ee][Aa][Dd][Mm][Ee]*'
```

Hay dùng tham số -iname:

```
$ find /usr/share/doc -iname readme\*
```

1.1.11 Lệnh find và các biểu thức quan hệ

Nếu bạn đã quen với biểu thức quan hệ, bạn có thể dùng tùy chọn `–regex` để chỉ in ra tên các tập tin mà nó trùng khớp với mẫu. Nếu không phân biệt chữ hoa chữ thường trong mẫu thì dùng tùy chọn `-iregex`.

1.1.12 Tùy chọn type của find

Tùy chọn `-type` cho phép bạn tìm kiếm các đối tượng trong hệ thống tập tin theo những kiểu khác nhau. Các tham số có thể của tùy chọn `-type` là `b` (cho thiết bị dạng khối), `c` (thiết bị dạng ký tự), `d` (thư mục), `p` (ống dẫn có tên), `f` (tập tin thường), `l` (liên kết mềm), và `s` (socket). Ví dụ, để tìm kiếm các liên kết mềm trong thư mục `/usr/bin` mà nó có chứa chuỗi `vim` ta thực hiện lệnh sau:

```
$ find /usr/bin -name '*vim*' -type l
/usr/bin/rvim
/usr/bin/vimdiff
/usr/bin/gvimdiff
```

1.1.13 find và tùy chọn mtimes

Tùy chọn `-mtime` cho phép bạn chọn các tập tin dựa trên thời gian cập nhật sau cùng của nó. Tham số của mtime là những khoảng 24 giờ, và có thể thêm dấu cộng (có nghĩa là sau) hoặc dấu trừ (có nghĩa là trước). Ví dụ , Xem xét trường hợp sau:

```
$ ls -l ?
-rw------- 1 root root 0 Jan 7 18:00 a
-rw------- 1 root root 0 Jan 6 18:00 b
-rw------- 1 root root 0 Jan 5 18:00 c
-rw------- 1 root root 0 Jan 4 18:00 d
$ date
Mon Jan 7 18:14:52 EST 2002
```

Bạn có thể tìm các tập tin mà nó được tạo ra trong vòng 24 giờ vừa qua bằng lệnh sau:

```
$ find . -name \? -mtime -1
./a
```

Hoặc bạn có thể tìm các tập tin mà chúng được tạo ra trước cách đây đã hơn 24 giờ:

```
$ find . -name \? -mtime +0
./b
./c
./d
```

1.1.14 The -daystart option

Nếu bạn mô tả tùy chọn `-daystart`, khi đó khoảng thời gian sẽ được tính từ giờ bắt đầu của ngày hiện tại chứ không là 24 giờ trước đây. Ví dụ, tìm các tập tin được tạo ngày hôm qua và ngày hôm kia:

```
$ find . -name \? -daystart -mtime +0 -mtime -3
./b
./c
$ ls -l b c
-rw------- 1 root root 0 Jan 6 18:00 b
-rw------- 1 root root 0 Jan 5 18:00 c
```

1.1.15 Tùy chọn -size

Tùy chọn `-size` cho phép bạn tìm các tập tin dựa trên kích thước của chúng. Mặc định, đối số của `-size` là các khối 512-byte, tuy nhiên việc thêm vào các hậu tố làm mọi việc dễ dàng hơn. Các hậu tố được chấp nhận là `b` (khối 512-byte), `c` (bytes), `k` (kilobytes), and `w` (2-byte). Bên cạnh đó, bạn có thể thêm vào dấu cộng để nói rằng lớn hơn hoặc dấu trừ đề nói là nhỏ hơn.

Ví dụ, để tìm các tập tin bình thường mà nó nhỏ hơn 50 bytes ta dùng lệnh sau:

```
$ find /usr/bin -type f -size -50c
/usr/bin/krdb
/usr/bin/run-nautilus
/usr/bin/sgmlwhich
/usr/bin/muttbug
```

1.1.16 Xử lý các tập tin được tìm thấy

Bạn có thể tự hỏi rằng bạn có thể làm gì với các tập tin tìm ra được. Lệnh `find` có khả năng tác động trên các tập tin tmf được với tùy chọn `-exec`. Tùy chọn này nhận một dòng lệnh để thực thi như là tham số của nó và kết thức bằng ký tự ; và sẽ thay thế bất kỳ một thể hiện nào của cặp `{}` với tên tập tin tìm được. Xem ví dụ sau:

```
$ find /usr/bin -type f -size -50c -exec ls -l '{}' ';'
-rwxr-xr-x 1 root root 27 Oct 28 07:13 /usr/bin/krdb
-rwxr-xr-x 1 root root 35 Nov 28 18:26 /usr/bin/run-nautilus
-rwxr-xr-x 1 root root 25 Oct 21 17:51 /usr/bin/sgmlwhich
-rwxr-xr-x 1 root root 26 Sep 26 08:00 /usr/bin/muttbug
```

1.1.17 Lệnh locate

Tìm kiếm tập tin bằng lệnh find mất nhiều thời gian, vì nó phải thực hiện việc tìm trên từng thư mục một. Lệnh `locate` có thể cải tiến tốc độ tìm kiếm bằng việc sử dụng một cơ sở dữ liệu phục vụ cho việc tìm kiếm thông tin trên hệ thống. Lệnh `locate` sẽ dò tìm tất cả các phần trên đường dẫn chứ không dừng lại tên tập tin

Ví dụ:

```
$ locate bin/ls
/var/ftp/bin/ls
/bin/ls
/sbin/lsmod
/sbin/lspci
/usr/bin/lsattr
/usr/bin/lspgpot
/usr/sbin/lsof
```

1.1.18 Dừng lệnh updatedb

Hầu hết các hệ điều hành Linux đều có một tiến trình thực hiện việc cập nhật cơ sở dữ liệu. Nếu lệnh locate bị lỗi, hãy dùng lệnh `updatedb` để cập nhật lại cơ sở dữ liệu tìm kiếm.

Ví dụ:

```
$ locate bin/ls
locate: /var/spool/locate/locatedb: No such file or directory
$ su
Password:
# updatedb
```

Lệnh `updatedb` mất thời gian để thực hiện.

Trong nhiều phiên bản Linux, lệnh `locate` được thay thế bằng lệnh `slocate`.

3 Biểu thức quan hệ

1.1.19 Biểu thức quan hệ là gì ?

Một biểu thức quan hệ là một cấu trúc đặc biệt được dùng để biểu diễn **các mẫu văn bản** (text patterns). Trên hệ thống Linux, các biểu thức quan hệ thường dùng để tìm kiếm các mẫu văn bản cũng như được sử dụng để thực hiện các tác vụ tìm và thay thế trên các dòng văn bản.

1.1.20 So sánh với các ký tự đại diện

Khi chúng ta nhìn vào các biểu thức quan hệ, bạn có thể thấy rằng cú pháp của chúng rất giống như cú pháp dùng các ký tự đại diện trong tên tập tin. Tuy nhiên khi nghiên cứu kỹ chúng ta sẽ thấy chúng hoàn toàn khác biệt nhau.

1.1.21 Tìm chuỗi con đơn giản

Lệnh `grep` dò nội dung của một tập tin theo một biểu thức quan hệ để in ra các dòng mà có chứa đoạn văn bản trùng khớp với mô tả trong biểu thức quan hệ.
Cú pháp: grep regex filename:
Trong đó:

regex: là biểu thức quan hệ.
filename: là tập tin để dò tìm

Ví dụ:

```
$ grep bash /etc/passwd
operator:x:11:0:operator:/root:/bin/bash
root:x:0:0::/root:/bin/bash
ftp:x:40:1::/home/ftp:/bin/bash
```

Trong ví dụ trên `Grep` đọc từng dòng của tập tin `/etc/passwd` và áp dùng biểu thức quan hệ `bash` vào nội dung tập tin để tìm và in ra các dòng có chứa chuỗi `bash`.

Thông thường, để tìm một chuỗi con bạn chỉ việc mô tả chuỗi muốn tìm mà không cần mô tả thêm một ký tự đặc biệt nào cả. Tuy nhiên, nếu chuỗi con cần tìm có chứa cá chuỗi a +, `.`, *, `[`, `]`, hoặc `\`, thì cần phải đưa chúng vào cặp dấu nháy đôi và đặt phía trước chúng ký tự '\' (backslash). Dưới đây là một số chuỗi con:

- `/tmp` (dò tìm chuỗi `/tmp`)
- `*"\[box\]"` (dò tìm chuỗi `[box]`)
- `*"\*funny\*"` (dò tìm chuỗi `*funny*`)
- `*"ld\.so"` (dò tìm chuỗi `ld.so`)

1.1.22 Metacharacters

Với biểu thức quan hệ, bạn có thể thực hiện những tác tìm kiếm phức tạp bằng cách ứng dụng các ký tự đa năng (*metacharacters)*. Một trong số các ký tự đa năng là ký tự dấu chấm `.`, nó sẽ trùng khớp với bất kỳ một ký tự đơn nào.
Ví dụ:

```
$ grep dev.hda /etc/fstab
/dev/hda3 / reiserfs noatime,ro 1 1
/dev/hda1 /boot reiserfs noauto,noatime,notail 1 2
/dev/hda2 swap swap sw 0 0
#/dev/hda4 /mnt/extra reiserfs noatime,rw 1 1
```

Trong ví dụ trên, chuỗi `dev.hda` không xuất hiện trong bất kỳ dòng nào của tập tin `/etc/fstab`. Tuy nhiên, grep đã không tìm chuỗi `dev.hda` mà tìm mẫu (pattern) `dev.hda`.

Hãy nhớ rằng ký tự . sẽ trùng khớp với bất kỳ ký tự đơn nào. Như vậy ký tự .có cùng chức năng với ký tự đại diện ? trong tên tập tin.

1.1.23 Sử dụng []

Nếu bạn muốn sự trùng khớp trong các mẫu nhiều hơn một ký tự như trường hợp dùng ký tự . , bạn có thể sử dụng cặp ký tự [và](square brackets) để mô tả một tập con các ký tự cần trùng khớp.

Ví dụ:

```
$ grep dev.hda[12] /etc/fstab
/dev/hda1 /boot reiserfs noauto,noatime,notail 1 2
/dev/hda2 swap swap sw 0 0
```

Ví dụ trên sử dụng mẫu [1,2] để đại diện cho ký tự '1' hoặc ký tự '2'

1.1.24 Sử dụng [^]

Bạn có thể đảo ngược ý nghĩa của cặp [] bằng cách đặt ký tự ^ ngay sau dấu [, để mô tả một ký tự khác với các ký tự được liệt kê trong cặp dấu ngoặc vuông [ ]

Ví dụ:

```
$ grep dev.hda[^12] /etc/fstab
/dev/hda3 / reiserfs noatime,ro 1 1
#/dev/hda4 /mnt/extra reiserfs noatime,rw 1 1
```

1.1.25 Điểm lưu ý

Khi đặt một ký tự đa năng vào bên trong cặp dấu [] sẽ làm mất ý nghĩa của ký tự đó. Ví dụ, nếu bạn đặ ký tự . vào trong cặp dấu [] thì nó được xem như là ký tự dấu chấm bình thường như các ký tự khác 1,2,3 ..

Ví dụ để in ra các dòng trong tập tin /etc/fstab có chứa chuỗi con dev.hda ta đánh dòng lệnh:

```
$ grep dev[.]hda /etc/fstab
```

Hay

```
$ grep "dev\.hda" /etc/fstab
```

1.1.26 Ký tự đa năng "*"

Một vài ký tự đa năng không trùng khớp với bất kỳ ký tự nào khác nhưng dùng để bổ sung ý nghĩa cho các ký tự đứng phía trước nó. Một trong số chúng là ký tự * (asterisk), được sử dụng để mô tả sự trùng khớp của 0 hoặc nhiều lần lặp lại của ký tự đúng trước . Ví dụ:

- ab*c (sẽ trùng với abbbbc nhưng không trùng với abqc)
- ab*c (sẽ trùng với abc nhưng không trùng với abbqbbc)
- ab*c (sẽ trùng với ac nhưng không trùng với cba)
- b[cq]*e (sẽ trùng với bqe nhưng không trùng với eb)
- b[cq]*e (sẽ trùng với bccqqe nhưng không trùng với bccc)
- b[cq]*e (sẽ trùng với bqqcce nhưng không trùng với cqe)
- b[cq]*e (sẽ trùng với bbbeee)
- .* (sẽ trùng với bất kỳ chuỗi nào)
- foo.* (sẽ trùng với bất kỳ chuỗi nào bắt đầu bằng chuỗi foo)

1.1.27 Mô tả bắt đầu và kết thức chuỗi

Hai ký tự ^ và $ dùng để mô tả sự trùng khớp ở đầu vài cuối chuỗi. Bằng cách sử dụng một dấu ^ tại đầu của biểu thức quan hệ, bạn đã yêu cầu mẫu dò tìm diễn ra tại đầu tập tin.

Ví dụ sau sử dụng biểu thức ^# để tìm tất cả các dòng bắt đầu bằng ký tự # trong tập tin /etc/fstab:

```
$ grep ^# /etc/fstab
# /etc/fstab: static file system information.
#
```

Ký tự ^ và $ có thể được kết hợp để tìm kiếm trên cả dòng.

Ví dụ, biểu thức sau sẽ trùng khớp với tất cả các dòng bắt đầu bằng ký tự # và kết thức bằng ký tự . character, ở giữa dòng là bất ký chuỗi ký tự nào:

```
$ grep '^#.*\.$' /etc/fstab
# /etc/fstab: static file system information.
```

Trong ví vụ trên, chúng ta bao biểu thức quan hệ lại bằng cặp ký tứ nháy đơn ‘ ‘ để ngăn ngừa trường hợp ký tự $ được thông dịch bởi shell