CHUYÊN ĐỀ HỆ ĐIỀU HÀNH LINUX

Tuần 3: HỆ THỐNG QUẢN LÝ FILE – QUYỀN TRÊN FILE VÀ THƯ MỤC

GVLT: NGUYỄN Thị Minh Tuyền

Nội dung

1. Hệ thống quản lý file trong Linux
2. Phát triển tên file
3. Phát triển các dấu ngoặc
4. Cú pháp cơ bản trong ngôn ngữ bash

Nội dung

1. Hệ thống quản lý file trong Linux
2. Phát triển tên file
3. Phát triển các dấu ngoặc
4. Cú pháp cơ bản trong ngôn ngữ bash

Hệ thống quản lý file

- Cấu trúc cây phân cấp (cây) :
  - Thư mục gốc /
  - Mỗi thư mục có thể chứa:
    - Các file
    - Các thư mục con
- Đường dẫn :
  - Tuyệt đối : từ thư mục gốc, bắt đầu bởi "/ ": `/home/cours/Linux/test.txt`
  - Tương đối: so với thư mục hiện hành : `/Linux/test.txt`
- Một số ký hiệu
  - . Thư mục hiện hành
  - .. Thư mục cha
  - // /
- Vai trò của "/":
  - Đặt trước tên đường dẫn tuyệt đối để chỉ gốc của cây
  - Là dấu ngăn cách giữa các thư mục

Cây phân cấp chuẩn

/etc	Các file và script cấu hình hệ thống

Các quyền trên file và thư mục [1]

- Hiển thị các quyền :

```
$ ls -l hello.sh
-rwxr-xr-x  1 tuyennguyen1  everyone  54 May  9 22:02 hello.sh
```

(Bản chất, quyền, số liên kết vật lý, sở hữu, nhóm, kích thước, ngày giờ cập nhật, tên)

- Trong `-rwxr-xr-x`: 10 ký tự:
  - Ký tự 1 – file thường, d thư mục, 1 liên kết, ...
  - Ký tự 2 – 10 r, w, x = quyền được thiết lập, - = không có quyền.
- Ba mức về quyền :
  - 2 – 4 `user` chủ sở hữu
  - 5 – 7 `group` nhóm sở hữu
  - 8 – 10 `others` người khác

Các quyền trên file và thư mục [2]

- Các quyền trên file :
  - r Đọc file
  - w Cập nhật file
  - x Thực thi file
- Các quyền trên thư mục :
  - r đọc nội dung thư mục → `ls`
  - w thay đổi nội dung của thư mục
  - x duyệt qua thư mục → `cd`

Thay đổi quyền với lệnh chmod

- Cú pháp :

  chmod mode files

  mode = chuỗi các kỹ tự để gán quyền hay huỷ quyền

- Ví dụ:

  `$ chmod u=rwx,g=rx,o= f1.txt` quyền rwxr-x---

  `$ chmod ugo=rw f1.txt` quyền rw cho tất cả

  `$ chmod ugo+x f1.txt` gán quyền thực thi cho tất cả

- Sử dụng mã octal : r= 4, w= 2, x= 1

`rwxr-x--- = (4+2+1)(4+1)(0) = 750 → chmod 750 f1.txt`

Lệnh umask

- Quyền mặc định cho tất cả các file mới tạo : 666
  - Quyền đọc và ghi cho tất cả mọi người
- Quyền mặc định cho tất cả các đường dẫn mới tạo : 777
- Linux hỗ trợ cơ chế quyền truy cập mặc định: umask
- Cú pháp:

  umask [-S|⟨*umask*⟩]

- Ví dụ:

  $ umask 026 → Tất cả các file được tạo với quyền 640

Lệnh chown, chgrp

- Cú pháp:

```
chown ⟨user name⟩[:][⟨group name⟩] ⟨name⟩ ...
chown :⟨group name⟩ ⟨name⟩ ...
chgrp ⟨group name⟩ ⟨name⟩ ...
```

- Thay đổi quyền sở hữu và nhóm sở hữu không thay đổi quyền truy cập
- Hỗ trợ tuỳ chọn –R để áp dụng thay đổi cho cây phân cấp thư mục

Nội dung

1. Hệ thống quản lý file trong Linux
2. Phát triển tên file
3. Phát triển các dấu ngoặc
4. Cú pháp cơ bản trong ngôn ngữ bash

Trích xuất chuỗi con và tìm theo pattern

- pattern = tham số sử dụng các ký hiệu : `*` `?` `[...]`
- Bash thay thế mỗi pattern bởi danh sách các file tương ứng với pattern và cách nhau bởi các khoảng trắng, theo thứ tự từ điển.
- Một số pattern:
  - `*` bất cứ chuỗi nào, kể cả chuỗi rỗng
  - `?` Bất cứ ký tự nào
  - $[c_1c_2 \cdots c_n]$ bất cứ ký tự nào nằm trong danh sách
  - $[c_1\text{-}c_2]$ bất cứ ký tự nào nằm trong khoảng

Ví dụ

```
$ ls
bu2.c bu2.h ga.c ga.h ga.o meu1.c zo5.h
$ echo *.c
bu2.c ga.c meu1.c
$ echo *.[ch]
bu2.c bu2.h ga.c ga.h meu1.c zo5.h
$ echo ??[0-9].[ch]
bu2.c bu2.h zo5.h
```

- Dùng được với tên thư mục:

```
$ mv ../t[dp][1-5]/*.[ch] tmp
```

Các ký tự loại trừ

[^...] hoặc [!...] tương đương với một ký tự ∉[...]

```
$ ls
essai.txt prog1.c resume.txt
$ ls *[^0-9].*
essai.txt resume.txt
```

- Cú pháp :
  - Ta có thể lộn lẫn các phần tử và các khoảng :

    `[A-Za-z0-9_. ] [^A-Za-z0-9_. ]`
  - Thêm hoặc loại trừ ']' : đặt ký tự đó ở đầu

    `[]abc] [^]abc]`
  - Thêm hoặc loại trừ '-' : đặt ký tự đó ở đầu hoặc cuối

    `[-abc] [abc-] [^-abc] [^abc-]`
  - Thêm hoặc loại trừ '^' hoặc '!' : KHÔNg đặt nó ở đầu : [abc^!] [^abc^!]

Không có chuỗi tương ứng

- Nếu không có tên file hay thư mục nào tương ứng với pattern, pattern sẽ không được phát triển.

```
$ ls
essai.txt prog1.c resume.txt
$ echo *.txt
essai.txt resume.txt
$ echo *.h
*.h
$ ls *.h
ls: *.h: No such file or directory
```

Các file ẩn

- Các file hoặc thư mục bắt đầu bằng dấu '.' được ẩn đi, bao gồm cả các thư mục '.' và '..'

```
$ ls
essai.txt prog1.c resume.txt
$ ls –a
. .. .ancien.txt essai.txt prog1.c resume.txt
```

- Các pattern bắt đầu bang * hoặc ? không tương ứng với các file ẩn.

```
$ ls *.txt
essai.txt resume.txt
$ ls .*.txt
.ancien.txt
```

Các file ẩn thường gặp

- Đa phần các file ẩn được đặt trong $HOME :

```
$ ls -a $HOME
./
../
.bash_history  Lịch sử các lệnh
.bashrc        được chạy khi khởi động bash
.profile       được chạy khi đăng nhập
.config/       XDG, cấu hình của các chương trình
.cache/        XDG, dữ liệu không cần thiết
.local/share/  XDG, dữ lieu người dùng
.gconf/        các thiết lập thông qua gconf-editor
.mozilla/      dữ liệu của firefox
.thunderbird/  dữ liệu của thunderbird
.ssh/          cấu hình của ssh
```

Nội dung

1. Hệ thống quản lý file trong Linux
2. Phát triển tên file
3. Phát triển các dấu ngoặc
4. Cú pháp cơ bản trong ngôn ngữ bash

Phát triển các dấu ngoặc

- Cho phép tạo ra các chuỗi độc lập với sự tồn tại của file.
- Cú pháp :

  `{mot1,...,motn}` không có khoảng trắng

→ bash thay thế chuỗi ban đầu bằng cách thực hiện tích Descartes các danh sách

```
$ echo {ga,bu,meu}
ga bu meu
$ echo a{ga,bu,meu}z
agaz abuz ameuz
$ echo {ga,bu,meu}-{eggs,ham}
ga-eggs ga-ham bu-eggs bu-ham meu-eggs meu-ham
```

Sự xếp lớp

- Ta có thể xếp lớp sử dụng {}

```
$ echo ga{zo,pti{foo,bar}lo}bu
        ↓
$ echo ga{zo,ptifoolo,ptibarlo}bu
        ↓
gazobu gaptifoolobu gaptibarlobu
```

Bảo vệ các ký tự

- Ta có thể bảo vệ các ký tự {},␣ với \ hoặc '' hoặc ""
- Ví dụ:

```
$ echo {\{hop\},ploum,a\,b\␣c}
{hop} ploum a,b␣c
$ echo {"{hop}",ploum,'a,b␣c'}
{hop} ploum a,b␣c
```

- Phải có ít nhất một dấu phẩy

```
$ echo {ga}
{ga}
```

→ Không phát triển

```
$ echo {ga,}
ga
```

Trộn lẫn pattern và dấu ngoặc

- Trong khi một pattern chứa các dấu ngoặc :
  - Việc phát triển các {} được thực hiện đầu tiên, các pattern sẽ được nhân lên ;
  - Sau đó việc phát triển các pattern được thực hiện trên các file đã tồn tại.
- Ví dụ:

```
$ ls /home/{nguyen,guest}/tp[1-7]/*.c
        ↓
$ ls /home/nguyen/tp[1-7]/*.c /home/guest/tp[1-7]/*.c
        ↓
$ ls /home/nguyen/tp2/ga.c /home/nguyen/tp3/bu.c /home/guest/tp1/zo.c
```

Nội dung

1. Hệ thống quản lý file trong Linux
2. Phát triển tên file
3. Phát triển các dấu ngoặc
4. Cú pháp cơ bản trong ngôn ngữ bash

Lệnh rẽ nhánh if

```
if các tham số lệnh
then
    echo "thành công"
else
    echo "thất bại"
fi
```

- Phải cần một dấu ; hoặc một dấu RC trước then, else và fi
  → Lệnh được thực thi. Ở điểm kết thúc, rẽ nhánh theo mã kết thúc $?.
- Biến thể :

```
if .. ; then .. ; fi
if .. ; then .. ; else .. ; fi
if .. ; then .. ; elif .. ; then .. ; else .. ; fi
```

Vòng lặp while

```
while tham số lệnh
do
    echo "một vòng lặp"
done
```

- Phải cần ; hoặc một RC trước **do** và **done**
  → Lệnh được thực thi. Tại điểm kết thúc,, nếu **$?** là 0 (thành công), thì khối lệnh **do .. done** được thực hiện, và tiếp tục vòng lặp.

Lệnh `test` [1]

- Nhiều tuỳ chọn để kiểm tra: `file`, chuỗi, số nguyên
- Ước lượng một biểu thức bởi tham số, sau đó trả về thành công hay thất bại → được gọi bởi **`if`** hoặc **`while`**

```
if test -f "hello.sh" ; then
    echo "File đã tồn tại"
fi
```

- Biến thể :

```
if␣[␣-f␣"hello.sh"␣];␣then
```

→ chú ý các khoảng trắng !

- Các tuỳ chọn: xem `man test`

Lệnh `test` [2]:tuỳ chọn kiểm tra file

- `-d` True nếu file tồn tại và là một thư mục
- `-e` True nếu file tồn tại
- `-f` True nếu là file
- `-s` True nếu file không rỗng
- `-r` True nếu ta có thể đọc file
- `-w` True nếu ta có thể ghi file
- `-x` True nếu ta có thể thực thi file
- `file1 -nt file2` True nếu file1 tồn tại và mới hơn file2
- `file1 -ot file2` True nếu file1 tồn tại và cũ hơn file2

Lệnh `test` [3]: tuỳ chọn cho chuỗi

- `string` True nếu string là chuỗi không rỗng.
- `s1 = s2` True nếu chuỗi s1 và s2 giống nhau.
- `s1 != s2` True nếu chuỗi s1 và s2 không giống nhau.
- `s1 < s2` True nếu chuỗi s1 đứng trước chuỗi s2 dựa vào giá trị nhị phân của các ký tự của chuỗi.
- `s1 > s2` True nếu chuỗi s1 đứng sau chuỗi s2 dựa vào giá trị nhị phân của các ký tự của chuỗi

Lệnh `test` [3]: tuỳ chọn cho số nguyên

- `n1 -ne n2` True nếu n1 và n2 không bằng nhau.
- `n1 -gt n2` True nếu n1 lớn hơn n2.
- `n1 -ge n2` True nếu n1 lớn hơn hoặc bằng n2.
- `n1 -lt n2` True nếu n1 nhỏ hơn n2.
- `n1 -le n2` True nếu n1 nhỏ hơn hoặc bằng n2.

Lệnh `test` [4]: ước lượng biểu thức

- `! expression` True nếu biểu thức là false.
- `expression1 -a expression2` True nếu cả hai biểu thức expression1 và expression2 đều là true.
- `expression1 -o expression2` True nếu hoặc expression1 hoặc expression2 là true.
- (expression) True nếu expression là true.
- Toán tử `-a` có độ ưu tiên cao hơn toán tử `-o`.

Đảo ngược kết quả

- Ta có thể đảo ngược kết quả của một lệnh bằng ! :

```
! Các tham số lệnh
```

- Ví dụ :

```
$ false ; echo $? 1
$ ! false ; echo $? 0
```

- Sử dụng ! trong các cú pháp :

```
if ! Các tham số lệnh ; then ... ; fi
if ! test ... ; then ... ; fi
if ! [ ... ]; then ... ; fi
while ! Các tham số lệnh ; do ... ; done
```