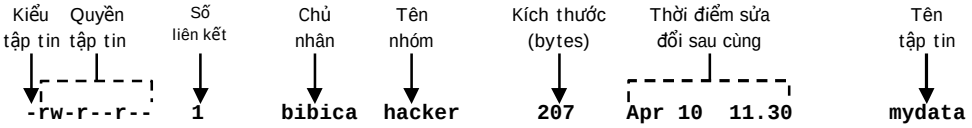
BÀI 5
CÁC TẬP LỆNH LINUX

I. Quyền sử dụng tập tin và thư mục

- Tất cả các tập tin và thư mục của Linux đều có người sở hữu và quyền truy nhập. Có thể đổi các tính chất này cho phép nhiều hay ít quyền truy nhập hơn đối với một tập tin hay thư mục. Quyền của tập tin còn cho phép xác định tập tin có là một chương trình (application) hay không (khác với Windows xác định tính chất này qua phần mở rộng của tên tập tin).

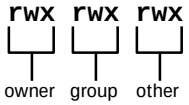


1. Kiểu tập tin:

Thể hiện bằng ký tự đầu tiên trong chuỗi thuộc tính của tập tin.

Ký tự đầu	Giải thích
-	File thông thường
-	Hard link
d	Thư mục
l	Link mềm (symbolic link)
c/b	Character (c) / Block device (b)
s	Domain socket
p	Name pipe

2. Quyền tập tin



Có 3 đối tượng chính là {**owner**, **group**, **other**} và mỗi đối tượng ứng với 3 quyền cụ thể {**read**, **write**, **execute**}.

r: Read ⇨ Thuộc tính đọc (không có quyền ghi/xóa)
w: Write ⇨ Thuộc tính ghi (hiệu chỉnh nội dung)
x: Execute ⇨ Thuộc tính thực thi (chạy chương trình)
-: None ⇨ Không có quyền trên đối tượng

- Quyền tập tin được thay đổi bằng lệnh **chmod**: chú ý không thay đổi được quyền của symbolic link.

```
chmod [-R] mode file
```

Có hai phương pháp dùng **chmod**: phương pháp tượng trưng và phương pháp tuyệt đối.

- Phương pháp tượng trưng:

Đối tượng truy nhập (Who)	Gán/thu hồi quyền (Operation)	Quyền (Permission)
Một bộ: **u** (user, tức owner) **g** (group – nhóm) **o** (other – người dùng còn lại) **a** (all –tương đương ugo)	Có quyền: **=** (gán quyền) **+** (thêm quyền) **-** (thu hồi quyền)	Một bộ: **r** (read – đọc) **w** (write - ghi) **x** (execute – thực thi) hoặc một trong các đối tượng sau để sao chép quyền: **u** (user, tức owner) **g** (group – nhóm) **o** (other – người dùng còn lại) để thiết lập user id, group id: **s**

Ví dụ:

```
$ chmod u+rx-w /tmp
$ chmod –R u+rwx,g+r-w,o-rwx /tmp
$ chmod a=rx abc.txt
```

- Phương pháp tuyệt đối:

Song song với cách ký hiệu miêu tả bằng ký tự như ở trên, quyền thao tác tập tin còn có thể cho dưới dạng 3 số, ví dụ quyền 644. Các số có thể nhận tất cả các giá trị từ 0 đến 7. Số đầu tiên miêu tả quyền của **owner** (sở hữu), số thứ hai cho **group** (nhóm) và số thứ ba cho **other** (những người còn lại).

Mỗi số là tổng của các quyền theo quy tắc sau :

Quyền đọc 4
Quyền ghi 2
Quyền thực thi 1

Kết hợp như sau (cho một đối tượng **u**, **o** hoặc **g**)

Số bát phân	Số nhị phân	Quyền	Giải thích
0	**000**	**---**	không có mọi quyền
1	**001**	**--x**	quyền thực thi
2	**010**	**-w-**	chỉ ghi (hiếm gặp)
3	**011**	**-wx**	ghi và thực thi (hiếm gặp)
4	**100**	**r--**	chỉ đọc
5	**101**	**r-x**	đọc và thực thi
6	**110**	**rw-**	đọc và ghi
7	**111**	**rwx**	đọc, ghi và thực thi

Ví dụ:

```
$ chmod 751 abc.txt
```

có nghĩa là **owner** có quyền **read**, **write**, và **execute** (4+2+1=7), **group** có quyền **read** và **execute** (4+1=5), và **other** chỉ có quyền **execute** (1). Xem cách tính toán trong bảng sau:

owner			group			other		
r	w	x	r	w	x	r	w	x
4			4			0		
	2			0			0	
		1			1			1
7			5			1		

Thư mục được gán sticky bit thì chỉ owner của tập tin và root mới được phép sửa, xóa file. Nhận diện thư mục có sticky bit **drwxr-xr-t**: quyền thực thi của người dùng khác là **t**.

Gán bit sticky cho thư mục theo mẫu **0xxx** như sau:

```
$ mkdir test
$ chmod -R 1777 test
$ ls –l
drwxwrxrwt   2   s01   student   40   Apr 12   16:00   test
```

- Các quyền là mặc định khi tạo tập tin. Khi một tập tin hay thư mục được tạo ra, quyền mặc định sẽ được xác định bởi các quyền 666 trừ bớt các quyền hiển thị bằng lệnh **umask**. Quyền cao nhất ở đây là 666 do quyền thực thi chương trình cần được gán cố ý bởi người sử dụng hay các chương trình biên dịch.

```
$ umask 002
$ echo tao mot file > tmp
$ ls -l
total 5472
-rw-rw-r--   1  s01   student   12 Apr  3 21:55 tmp          //Quyền 664
$ umask 022
$ echo tao mot file khac > tmp1
$ ls -l
```

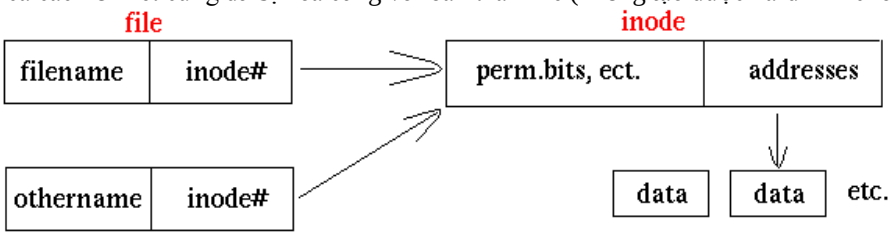
```
-rw-rw-r--   1  s01   student   12 Apr  3 21:55 tmp
-rw-r--r--   1  s01   student   12 Apr  3 21:59 tmp1        //Quyền 644
```

Giá trị mặc định của các quyền thường được gán mỗi khi người sử dụng login vào hệ thống thông qua các tập tin ẩn khởi tạo biến môi trường như **.profile**, **.bashrc**. Đứng trên quan điểm bảo mật hệ thống, giá trị 027 là tốt nhất, nó cho người cùng nhóm có quyền đọc và không cho quyền nào với những người khác.

3. Số liên kết

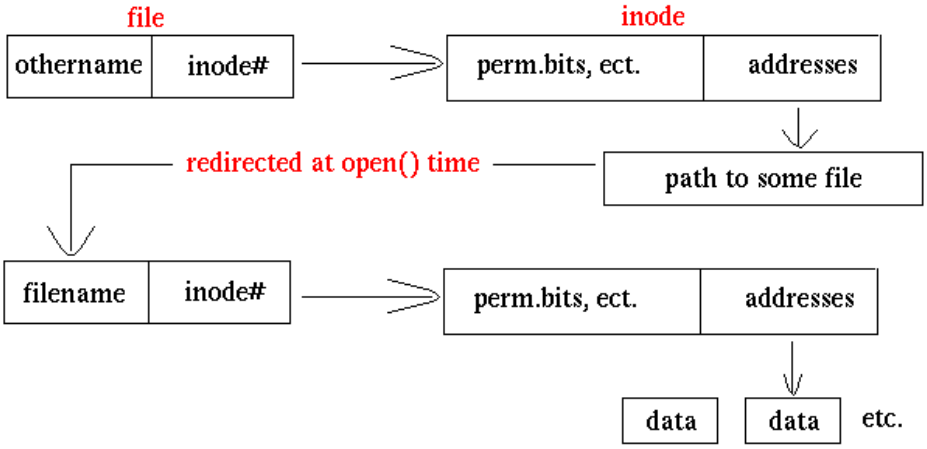
Linux và UNIX cho phép bạn tạo ra một file liên kết tắt (symbol link) đến một file vật lý khác. File liên kết tắt có thể trỏ đến một file hoặc thư mục. Có 2 loại liên kết:

- Liên kết tắt cứng (hard link): tạo ra một file trỏ đến cùng mục nhập i-node của file vật lý trên đĩa. Và do đó file vật lý trên đĩa chỉ thật sự bị xóa khi tất cả các liên kết cứng đã bị xóa cùng với bản thân file (không tạo được hard link cho thư mục).




```
$ln filename othername
```

- Liên kết tắt mềm (soft link): chỉ chứa các thông tin trỏ đến file vật lý. File liên kết mềm không tham chiếu trực tiếp đến điểm nhập i-node của file vật lý mà nó trỏ đến. Nếu bạn xóa file vật lý ban đầu đi thì file liên kết mềm sẽ không biết đường tham chiếu đến file gốc ban đầu nữa. Tuy nhiên một khi bạn tạo lại file gốc vật lý thì file liên kết mềm vẫn tiếp tục có hiệu lực.




```
$ln -s filename othername
```

4. Thay đổi chủ sở hữu

- Tạo người dùng mới tên user1: **useradd user1**
- Tạo một tập tin test1.txt ở thư mục gốc "/": **touch /test1.txt**
- Thay đổi quyền ownership của tập tin text1.txt là user1: **chown user1 /test1.txt**
- Kiểm tra lại: **ls -l | grep test1.txt**

Lưu ý: Nếu muốn thay đổi ownership cho một thư mục và các thư mục con bên trong thì ta dùng tùy chọn **(-R)** cho lệnh chown.VD: **chown -R user1 /test**

5. Thay đổi nhóm sở hữu

- Tạo nhóm mới tên group1: **groupadd group1**
- Thay đổi group sở hữu của tập tin test1.txt là group1: **chgrp group1 /test1.txt**
- Kiểm tra lại**: ls –l /**

Lưu ý: Nếu muốn thay đổi group sở hữu cho một thư mục và các thư mục con bên trong ta dùng tùy chọn (-R) cho lệnh chgrp.
VD: **chgrp –R group1 /test**

II. Điều khiển tiến trình

1. Định hướng nhập xuất

Các tiến trình thường nhận dữ liệu đầu vào xử lý và ghi kết xuất ra một nơi nào đó. Linux quy định cơ bản đầu vào là bàn phím stdin (thiết bị nhập chuẩn) và đầu ra là màn hình stdout (thiết bị xuất chuẩn).
Ví dụ: lệnh **ls –l** sẽ lấy thông số dòng lệnh gõ vào từ bàn phím đọc duyệt thư mục và in kết quả ra màn hình.

```
$ls –l
-rw-rw-r--   1  s01   student   12 Apr  3 21:55 tmp
…
```

Linux cung cấp cơ chế chuyển hướng xuất nhập. Ký hiệu lệnh **>** dùng để chuyển hướng kết xuất đầu ra trong khi **<** dùng để chuyển hướng kết xuất đầu vào.
Ví dụ: sử dụng lệnh **ls** sau đó ghi kết xuất ra file **data.txt**

```
$ls –l >data.txt
```

Nếu muốn kết xuất của lệnh ghi nối đuôi vào file hiện có, bạn dùng chuyển hướng **>>**. Ví dụ:

```
$ls –l >>data.txt
```

Lệnh **more** của Linux cho phép hiển thị dữ liệu của đầu vào theo từng trang. Bạn có thể chuyển cho lệnh **more** nội dung file bằng định hướng đầu vào như sau:

```
$more <bigfile.txt
```

2. Kiểm soát tiến trình

a. Xem thông tin về tiến trình

Muốn xem các tiến trình đang chạy trong hệ thống Linux hiện hành bạn gọi lệnh **ps**

```
$ps
PID     TTY        TIME       CMD
128     tty1     00:00:00     bash
137     pts/9    00:00:00      mc
235     pts/0    00:00:00     bash
…
```

Lệnh **ps** tương tự chức năng Task list trên Window khi bạn nhấn **Ctrl – Alt – Del**. Lệnh **ps** có rất nhiều tùy chọn. Ví dụ ta sử dụng tùy chọn **–a** (all) yêu cầu liệt kê tất cả các tiến trình trong Linux.

b. Tiến trình tiền cảnh

Khi đang trên dấu nhắc hệ thống (**#** hoặc **$**) và gọi một chương trình, chương trình trở thành tiến trình và đi vào hoạt động dưới sự kiểm soát của hệ thống. Dấu nhắc hệ thống sẽ không hiển thị khi tiến trình đang chạy. Khi tiến trình hoàn thành tác vụ và chấm dứt, hệ điều hành (chính xác hơn là hệ vỏ Shell) sẽ trả lại dấu nhắc để bạn gõ lệnh thực hiện chương trình khác. Chương trình hoạt động theo cách này gọi là chương trình tiền cảnh (foreground). Ví dụ: lện **ls –R** liệt là lệnh liệt kê đệ quy tất cả các thư mục con. Bạn hãy thực hiện lệnh này từ dấu nhắc hệ thống như sau:

```
$ls –R /
```

Dấu nhắc hệ thống trở lại khi chương trình đã thực hiện xong.

c. Tiến trình hậu cảnh

Nếu có cách nào đó yêu cầu Linux đưa các tiến trình chiếm nhiều thời gian xử lý hoặc ít tương tác với người dùng ra hoạt động phía hậu cảnh (background) trả lại ngay dấu nhắc để các tiến trình ở tiền cảnh có thể thực thi thì tốt hơn. Linux cung cấp khả năng này bằng lệnh **&** kết hợp với lệnh của chương trình mà bạn gõ từ dấu nhắc hệ thống. Tất cả những lệnh gõ kèm theo chỉ thị **&** đều được hệ điều hành đưa vào hoạt động ngầm bên trong. Ví dụ:

```
$ls –R / &
[1] 23978
$
```

Khi tiến trình hậu cảnh chấm dứt, hệ thống sẽ tự động đưa ra thông báo như sau:

```
$
[1] Done                    ls –R
```

d. Tạm dừng tiến trình

Sử dụng phím **Ctrl – Z** để đưa một tiến trình đang chạy ở tiền cảnh vào chạy ở hậu cảnh. Khi một tiến trình nhận được tín hiệu **Ctrl – Z** nó sẽ bị hệ thống cho tạm dừng và đưa vào hậu cảnh. Dấu nhắc hệ thống được trả lại cho người dùng. Tuy đưa vào hậu cảnh nhưng tiến trình đang bị tạm dừng, nó chỉ thực sự chạy lại ở hậu cảnh khi bạn cho phép . Ví dụ:

```
$ls –R / >allfiles.txt
^Z
[1]+ Stopped        ls –R / >allfiles.txt
$
```

Muốn xem PID của tiến trình bạn gọi lệnh **ps –af**

e. Đánh thức tiến trình

Sử dụng lệnh **jobs** để kiểm tra chương trình của ta đang dừng hay đang chạy.

```
$jobs
[1]+ Stopped        ls –R / >allfiles.txt
```

Lệnh **jobs** hiển thị trạng thái của tất cả các tiến trình đang chạy ở hậu cảnh. Như kết quả trên: tác vụ **[1]** đang ở trạng thái dừng. Để yêu cầu tiến trình của ta tiếp tục hoạt động ở hậu cảnh: sử dụng lệnh **bg**.

```
$bg 1
Ls –R / >allfiles.txt
$jobs
[1]+ Running        ls –R />allfiles.txt &
```

Dùng lệnh **fg** để mang tiến trình trở lại hoạt động ở phía tiền cảnh.

```
$fg 1
Ls –R / >allfiles.txt
```

f. Hủy tiến trình

Không phải lúc nào tiến trình cũng hoạt động tốt đẹp. Có thể chúng sẽ bị treo hoặc bước vào vòng lặp vô tận và không bao giờ chấm dứt. Trong trường hợp này, ta cần phải loại bỏ chương trình ra khỏi hệ thống. Lệnh **kill** của Linux thường được dùng cho mục đích này, **kill** yêu cầu cung cấp mã số định danh PID của tiến trình. Lệnh **kill** thường dùng chung với lệnh **ps –af**. Bạn dùng lệnh **ps –af** để xem thông tin về tiến trình đang chạy, sau đó lấy PID của tiến trình cần hủy và gọi lệnh **kill**.

```
$ls –R / >data.txt
^Z
$ps -af
PID     TTY         TIME        CMD
128     tty1      00:00:00     bash
137     pts/9     00:00:00     ls -R
235     pts/0     00:00:00     bash
…
$kill 137

$ps -af
PID     TTY         TIME        CMD
128     tty1      00:00:00     bash
235     pts/0     00:00:00     bash
…
```

Có một số tiến trình có độ ưu tiên cao và không thể loại bỏ theo cách thông thường. Lúc này ta sử dụng kill ở cấp độ -9. Ví dụ:

```
$kill -9 137
```

III. Luyện tập

1. Thay đổi quyền truy xuất

- Truy cập bằng quyền root
- Tạo 2 người dùng: **user1** và **user2**
- Đặt password đăng nhập cho **user1** và **user2** và **roo**t
- Tạo nhóm người dùng: **group1**
- Chuyển **user1** và **user2** vào nhóm **group1**
- Tạo thư mục **/baitap1** với quyền **770**
- Đăng nhập vào tài khoản **user1**
- Viết chương trình c: **program1.c** đặt trong thư mục **/baitap1** in ra câu thông báo: "Hello world"

2. Liên kết tắt

- Viết chương trình **program2.c** in ra màn hình các số nguyên từ 0 đến 9 đặt trong thư mục **/baitap2/**
- Chuyển ra thự mục gốc "**/**"
- Tạo liên kết cứng với tên mới là **hardlink** đến tập tin thực thi
- Tạo liên kết mềm với tên mới là **softlink** đến tập tin thực thi
- Chạy chương trình với 2 liên kết vừa tạo ở trên
- Xóa tập tin thực thi trong thư mục **/baitap2/**
- Chạy lại chương trình với 2 liên kết ở trên

3. Điều khiển tiến trình

- Viết chương trình **program3.c** in ra các số từ nguyên 0 đến 20, mỗi giây in ra một số, ghi kết xuất ra tập tin **ketqua.txt**
- Cho chương trình chạy ở hậu cảnh.
- Khi chương trình hoàn tất, xem nội dung tập tin **ketqua.txt**