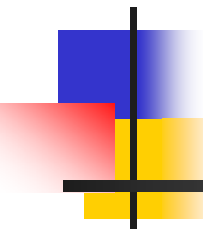


Hệ điều hành Linux

Phần 1: Sử dụng Linux

Đăng nhập vào Linux

- Ví dụ màn hình đăng nhập:

```
pc29 login: bctruong
password:   bctruong123
[bctruong@pc29 bctruong]$
```

  - `bctruong` và `bctruong123` là thông tin *username* và *password* đã cấp trong hệ thống (lưu ý Linux có phân biệt chữ hoa và chữ thường).
  - `pc29` là tên máy tính (hostname)
  - Ký hiệu *$* là dấu nhắc lệnh (bash shell)
  - Ký hiệu *#* là dấu nhắc lệnh của user *root*

Đăng nhập vào Linux (tt)

- Các loại user account trên Linux:
  - `root`: Admin account, tương tự account Administrator trên WinNT/200x
  - `daemon, shutdown, ftp, apache, …`: System account – dùng khi thực thi chương trình. Không thể đăng nhập bằng các account này.
  - `lan, viet, tuan, …`: User account, mỗi account thường được cấp một thư mục làm việc (home directory), ví dụ `/home/tuan`

Đăng nhập vào Linux (tt)

- Thông tin về user account:
  - `uid`: Mã người dùng (User ID)
  - `gid`: Mã nhóm người dùng (Group ID)
  - username, password
  - Home directory: Có chứa một số file chuẩn:
    - `.bash_profile`: Thực thi mỗi khi user login, thường dùng để đặt biến môi trường PATH
    - `.bash_logout`: Thực thi mỗi khi user logout.
    - `.bash_history`: Chứa các dòng lệnh user đã gõ.

Đăng nhập vào Linux (tt)

- Đề nghị về password an toàn:
  - Từ 8 kí tự trở lên
  - Không dùng các từ trong từ điển
  - Không đặt trùng với username
  - Chứa cả chữ, số và ký tự đặc biệt
  - Không ghi chép ra giấy
  - Nên thay đổi theo định kỳ
- Tránh đăng nhập bằng tài khoản `root`:
  - => Ngừa việc vô tình làm hỏng hệ thống

Đăng nhập vào Linux (tt)

- Tạo một user account mới:

```
pc29 login: root
password:    password
# useradd tuan          -> tạo user tuan
# passwd tuan           -> đặt mật khẩu
Changing password for user tuan
New password: password
Retype new password: password
# exit          -> thoát khỏi root
```

Lưu ý: password không hiển thị khi nhập

Đăng nhập vào Linux (tt)

- Đăng nhập bằng user account vừa tạo:

```
pc29 login: tuan
Password:    nhtuan123
$ passwd                -> đổi mật khẩu (nếu cần)
```

- Xem username của mình:

```
$ whoami
tuan
$ who am i
tuan   pts/0    May 18 18:29
```

Đăng nhập vào Linux (tt)

- Xem các user đang login:

```
$ who
tuan pts/0 May 18 18:29
root tty01 May 15 15:18
root tty03 May 15 15:17
```

  - Kí hiệu terminal nối trực tiếp vào máy:
    - tty01, tty02, tty03, …
  - Kí hiệu terminal giả lập (pseudo terminal):
    - pts/0, pts/1, pts/2, …

Đăng nhập vào Linux (tt)

- Xem chi tiết các user đang login:

```
$ w
15:10:41 up 4 days, 5:29, 3 users, load
  average: 0.91, 0.73, 0.35
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
tuan pts/0 10.1.1.3 18:29 .0s .01s .01s w
root tty01 -        15:18 2:0 .03s .00s w
root tty03 -        15:17 5:23 0s   0s  w
```

  - PCPU: Thời gian chương trình "WHAT" chạy
  - JCPU: Tổng thời gian các chương trình của user đang chạy

Thoát khỏi Linux

- Logout:
  - `$ logout` hoặc
  - `$ exit`
- Khởi động lại máy:
  - `$ reboot` hoặc
  - Nhấn tổ hợp phím `Ctrl - Alt - Del`
- Tắt máy:
  - `$ poweroff` hoặc
  - `$ shutdown –h now`

Các lệnh Linux cơ bản

- Sử dụng tùy chọn trên dòng lệnh:
  - Lệnh liệt kê nội dung thư mục:

```
$ ls
$
```

  - Liệt kê tất cả các file, kể cả file ẩn:

```
$ ls -a
./    .bash_history  .kermrc    .lessrc
```

Các lệnh Linux cơ bản (tt)

- Sử dụng tùy chọn trên dòng lệnh: (tt)
  - Liệt kê chi tiết tất cả file:

```
$ ls -al
total 10
drwxr-xr-x 3 tuan users 1024 Dec 21 ./
drwxr-xr-x 6 root root  1024 Dec 14 ../
-rw-r--r-- 1 tuan users  163 Dec  7 .kermrc
-rw-r--r-- 1 tuan users   34 Jun  6 .less
-rw-r--r-- 1 tuan users  114 Nov 23 .lessrc
drwxr-xr-x 2 tuan users 1024 Dec  7 .term/
```

  - Có thể thay đổi trật tự các tùy chọn:

```
$ ls -la          -> cho kết quả tương đương lệnh trên
$ ls -a -l        -> cho kết quả tương đương lệnh trên
```

Các lệnh Linux cơ bản (tt)

- Sử dụng tùy chọn trên dòng lệnh: (tt)
  - Liệt kê theo thứ tự thời gian file (giảm dần):

```
$ ls -alt
total 10
drwxr-xr-x 3 tuan users 1024 Dec 21 ./
drwxr-xr-x 6 root root  1024 Dec 14 ../
drwxr-xr-x 2 tuan users 1024 Dec  7 .term/
-rw-r--r-- 1 tuan users  163 Dec  7 .kermrc
-rw-r--r-- 1 tuan users  114 Nov 23 .lessrc
-rw-r--r-- 1 tuan users   34 Jun  6 .less
```

    - **t**: Viết tắt từ "sort by time"

Các lệnh Linux cơ bản (tt)

- Sử dụng tùy chọn trên dòng lệnh: (tt)
  - Liệt kê theo thứ tự thời gian file (tăng dần):

```
$ ls -altr
total 6
-rw-r--r-- 1 tuan users   34 Jun  6 .less
-rw-r--r-- 1 tuan users  114 Nov 23 .lessrc
drwxr-xr-x 2 tuan users 1024 Dec  7 .term/
-rw-r--r-- 1 tuan users  163 Dec  7 .kermrc
drwxr-xr-x 6 root root  1024 Dec 14 ../
drwxr-xr-x 3 tuan users 1024 Dec 21 ./
```

    - Tùy chọn **r** dùng để xuất kết quả theo thứ tự ngược lại
    - Lưu ý tùy chọn **r** khác với **R** – dùng khi liệt kê cả những thư mục con bên trong.

Các lệnh Linux cơ bản (tt)

- Sử dụng tham số dòng lệnh:
  - Liệt kê chi tiết thư mục /usr/local/src:

    ```
    $ ls –l /usr/local/src
    ```

  - Liệt kê thư mục cha:

    ```
    $ ls ..
    ```

  - Liệt kê thư mục làm việc của user:

    ```
    $ ls ~
    ```

Các lệnh Linux cơ bản (tt)

- Chuyển hướng nhập/xuất (redirection):
  - Dữ liệu đầu ra từ command1 được chuyển cho đầu vào của command2 thông qua "pipe":

    ```
    command1 | command2
    ```

  - Kí hiệu "|" có thể tìm thấy ở phím "\"
  - Liệt kê thư mục theo từng trang bằng cách sử dụng pipe:

    ```
    $ ls /usr/local/src | more
    ```

Các lệnh Linux cơ bản (tt)

- Chuyển hướng nhập/xuất: (tt)
  - Để gửi kết quả thực thi lệnh ra file, sử dụng kí hiệu “>”
  - Liệt kê một thư mục ra file output.txt trên đĩa:

```
$ ls /usr/local/src > output.txt
```

  - Sau đó, có thể xem nội dung output.txt:

```
$ more output.txt          hoặc
$ less output.txt          hoặc
$ tail output.txt    -> Chỉ xem 10 dòng cuối
```

Các lệnh Linux cơ bản (tt)

- Chuyển hướng nhập/xuất: (tt)
  - Kí hiệu "**>**" luôn tạo một file mới hoặc thay thế nội dung file đã có
  - Để bổ sung thêm nội dung mà không ghi đè lên file đã có, sử dụng kí hiệu "**>>**"
  - Bổ sung danh sách user đang làm việc vào file output.txt:

```
$ who >> output.txt
```

Các lệnh Linux cơ bản (tt)

- Quy ước mô tả lệnh Linux:
  - Chữ không nằm trong các dấu [], <>, {} được giữ nguyên:

    ```
    $ ls [-l]
    ```

  - Chữ trong [] là tùy chọn (không bắt buộc):

    ```
    $ ls [-l]
    ```

  - Dấu <> và chữ bên trong được thay bằng đoạn chữ tương ứng:

    ```
    $ more <filename>
    ```

    nếu <filename> là output.txt -> more output.txt

Các lệnh Linux cơ bản (tt)

- Quy ước mô tả lệnh Linux: (tt)
  - Dấu {} ý nói chỉ được chọn một trong nhiều giá trị liệt kê bên trong:

    `$ hostname` **`{<hostname> | -F <file>}`**

    -> `hostname <hostname>` hoặc

    -> `hostname -F <file>`

    Dấu | dùng để phân cách các lựa chọn với nhau
  - Dấu ... mô tả nhiều tham số tương tự:

    `$ more [`**`filenames`** `…]`

    -> `more file1 file2 file3 file4 file5`

Các lệnh Linux cơ bản (tt)

- Trợ giúp về lệnh Linux:
  - Xem tài liệu mô tả lệnh (manual):

    `$ man <command>` -> xem help của command

    `$ man -k <keyword>` -> tìm dòng chứa keyword
  - Xem hướng dẫn các lệnh shell (lệnh nội trú):

    `$ help`
  - Tài liệu LDP (Linux Documentation Project)

    http://www.tldp.org

Các lệnh Linux cơ bản (tt)

- Sử dụng ký tự thay thế (wildcard):
  - Ký tự *****:

```
$ ls /home/tuan/*.txt
input.txt  output.txt vanban.txt
```

  - Ký tự **?**:

```
$ ls /home/tuan/??put.txt
input.txt
```

- Sử dụng tính năng autocomplete:
  - `$ pass*` **<->** `$ passwd`
  - `$ pass<nhấn TAB>` **<->** `$ passwd`

Các lệnh Linux cơ bản (tt)

- Biến môi trường (environment variables):
  - Biến môi trường được định nghĩa riêng cho từng user, từng phiên làm việc.
  - Xem các biến môi trường:

    `$ env` hoặc `$ set`

    ```
    HOME=/home/tuan
    LOGNAME=tuan
    PATH=/usr/local/bin:/usr/bin
    SHELL=/bin/bash
    ```

  - PATH là nơi shell sẽ tìm lệnh để thực thi

Các lệnh Linux cơ bản (tt)

- Biến môi trường: (tt)
  - Đặt biến môi trường:

```
$ PATH = .:/usr/bin:/home/tuan
$ export PATH
```

Tương đương với:

```
$ export PATH = .:/usr/bin:/home/tuan
```

  - Thiết lập tự động biến môi trường khi login:
    - Gọi lệnh trên trong file **~/.bash_profile**

(**~** là ký hiệu thư mục làm việc của user)

  - Thực thi lại file .bash_profile:

```
$ source ~/.bash_profile
```

Các lệnh Linux cơ bản (tt)

- Tiến trình (process):
  - Là các chương trình đang thực thi bởi hệ thống hoặc người sử dụng.
  - Xem các process ta đang chạy:

```
$ ps          -> Viết tắt từ "process status"
PID  TTY      STAT TIME COMMAND
 41  pts/0    S    0:00 -bash
134  pts/0    R    0:00 ps
```

    - PID: Process ID (từ 0 -> 65565)
    - TTY: Terminal chạy process
    - Dấu -: Login shell (–bash)

Các lệnh Linux cơ bản (tt)

- Tiến trình (process): (tt)
  - Xem chi tiết các process ta đang chạy:

```
$ ps -u
USER PID %CPU %MEM VSZ  RSS TTY STAT START TIME COMMAND
tuan 26509 0.0 0.1 4328 1416 pts/0 S 09:18 0:00 -bash
tuan 26575 0.0 0.0 2592  640 pts/0 R 09:21 0:00 ps -u
```

    - STAT: Process Status (**S**=Sleeping, **R**=Running)
  - Xem tất cả các process đang chạy:

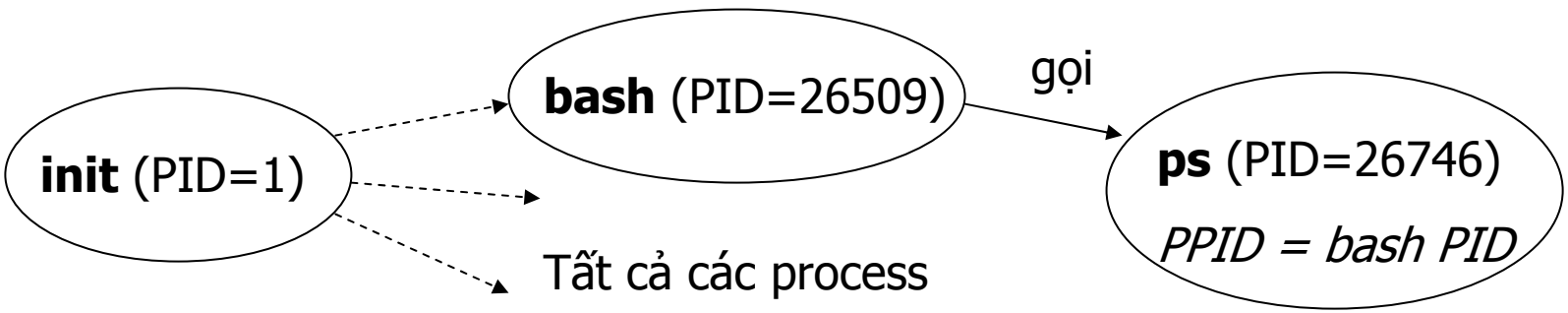
```
$ ps -a    -> Xem các process trên terminal này
$ ps -ax   -> Xem tất cả process
$ ps -aux  -> Xem chi tiết tất cả process
$ ps -axf  -> Xem cả quan hệ giữa các process
```

Các lệnh Linux cơ bản (tt)

- Tiến trình (process): (tt)
  - ps với tùy chọn –l:

```
$ ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
0 S 508 26509 26508 0 75 0 - 1083 wait4 pts/0 0 bash
0 R 508 26746 26509 0 80 0 -  779 -     pts/0 0 ps
```

    - PPID: Parent Process ID



Các lệnh Linux cơ bản (tt)

- Tiến trình (process): (tt)
  - Hủy tiến trình: Lệnh kill
    - Dùng khi không thể kết thúc chương trình bằng cách thông thường hay tổ hợp phím Ctrl-C, q
    - Bước 1: Login vào **root**
    - Bước 2: Dùng **`ps -u`** tìm PID của process cần kill
    - Bước 3: `$` **`kill`** `<PID>`
    - Bước 4: Dùng `ps -u <PID>` kiểm tra lại; nếu không còn -> bước 8; nếu vẫn còn nhưng với PID mới -> bước 6.

Các lệnh Linux cơ bản (tt)

- Tiến trình (process): (tt)
  - Hủy tiến trình: (tt)
    - Bước 5: Nếu process vẫn còn, kill với tùy chọn **-9**:

      `$ kill -9 <PID>`

      Dùng `ps -u` kiểm tra lại; nếu process vẫn còn -> bước 7; nếu process đã mất -> bước 8
    - Bước 6: Process xuất hiện lại với **PID mới** -> do parent process sinh ra -> cần kill parent process.
    - Bước 7: Dùng `ps -l` để **tìm PPID** và kill parent process theo các bước đã nêu.
    - Bước 8: Process đã bị kill. Cần logout khỏi root:

      `$ exit`

Các lệnh Linux cơ bản (tt)

- Chuyển thành user khác:
  - Thay vì phải logout khỏi user hiện tại và login lại từ đầu với user mới, ta dùng lệnh **`su`**:
  - Chuyển thành user root:

    `$ su` -> Viết tắt từ "super user"

    `Password:` -> Nhập password của root
  - Chuyển thành user root (với profile của root):

    `$ su -`

    `Password:` -> Nhập password của root
  - Chuyển thành user (khác root):

    `$ su [-] <username>`

Các lệnh Linux cơ bản (tt)

- Tìm kiếm chuỗi với lệnh grep:
  - Lệnh grep (Global Regular Expression Parser) tìm và hiển thị các dòng chứa chuỗi cần tìm.
  - Công dụng 1: Lọc kết quả lệnh:
    - **<command> | grep <pattern>**

```
$ ls | grep put          -> Liệt kê file có chữ put
input.txt        output.txt
$ ps –ax | grep “bash”   -> “” Tìm chính xác
 6368 tty2   S 0:00 -bash
27514 pts/0  S 0:00 -bash
27959 pts/0  S 0:00 grep bash
```

Các lệnh Linux cơ bản (tt)

- Tìm kiếm chuỗi với lệnh grep: (tt)
  - Công dụng 2: Tìm dòng có chứa chuỗi con trong file:
    - **grep <pattern> <filename>**

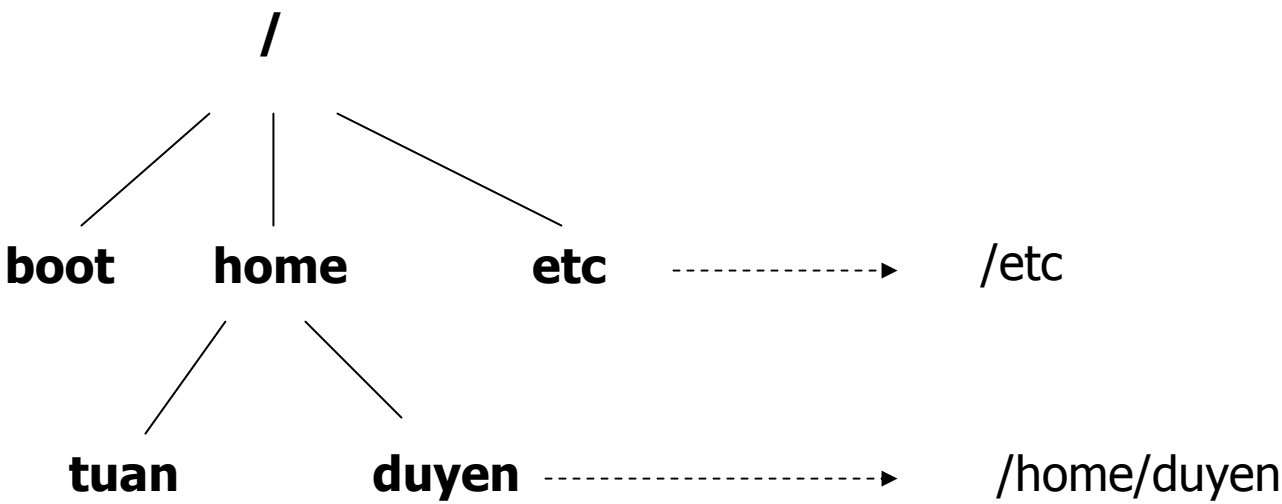
```
$ grep Window input.txt
X Window is a GUI of Linux
But Windows is a Microsoft's OS
$ grep "Window" input.txt
X Window is a GUI of Linux
```

Sử dụng hệ thống file

- Phân loại file theo nội dung:
  - Dữ liệu người dùng
  - Dữ liệu hệ thống: Ví dụ file /etc/passwd
  - File thực thi (chương trình)
- Tên file:
  - Tối đa 256 ký tự
  - Chứa các ký tự hoa, thường, số, dấu ., dấu -, dấu _, …
  - Một số hệ thống chỉ phân biệt 32 hay 64 ký tự đầu tiên trong tên file

Sử dụng hệ thống file (tt)

- Thư mục:
  - Là một loại file đặc biệt
  - Quy tắc đặt tên tương tự như file



Sử dụng hệ thống file (tt)

- Thư mục làm việc (home directory):
  - Mỗi user được cấp một thư mục để lưu trữ dữ liệu riêng của mình.
  - Tên thư mục thường trùng tên username và đặt trong thư mục `/home`.
  - Home directory còn được ký hiệu là dấu ~:

    `[tuan@pc29 /]$ cd ~` <-> `cd /home/tuan`
  - User có toàn quyền đọc ghi trong thư mục này
  - Mặc định các user không thể đọc ghi được trong home directory của nhau.

Sử dụng hệ thống file (tt)

- Lệnh in thư mục hiện hành (pwd):

```
cd /var/log
$ pwd          -> viết tắt từ Print working dir
/var/log
```

- Đường dẫn tương đối và tuyệt đối:
  - Đường dẫn tương đối: Căn cứ vào thư mục hiện hành
  - Đường dẫn tuyệt đối: Đường dẫn đầy đủ từ thư mục gốc đến file

Sử dụng hệ thống file (tt)

- Ví dụ đường dẫn tương đối:

```
$ cd /var/log          -> /var/log hiện hành
$ more messages        -> /var/log/messages
$ more ./samba/smb.log
                       -> /var/log/samba/smb.log
$more ../lib/mylib.txt
                       -> /var/lib/mylib.txt
```

  - Dấu "." kí hiệu thư mục user đang đứng
  - Dấu ".." kí hiệu thư mục cha

Sử dụng hệ thống file (tt)

- Lệnh chuyển thư mục (cd):

```
$ pwd
/home/tuan
$ cd ..        -> Chuyển ra thư mục cha /home
$ pwd
/home
$ cd ..        -> Chuyển ra thư mục cha /
$ pwd
/
```

Sử dụng hệ thống file (tt)

- Tạo file văn bản với lệnh cat:
  - Tạo file mới:

```
$ cat > newfile
Hello World
Here's some text
^D
```

  - Xem lại nội dung file:

```
$ cat newfile
Hello World
Here's some text
```

Sử dụng hệ thống file (tt)

- Tạo file văn bản với lệnh cat: (tt)
  - Bổ sung thêm vào file:

```
$ cat >> newfile
Some more lines
```

  - Nối nhiều file thành một file:

```
$ cat > anotherfile
Diffrerent text
$ cat newfile.txt anotherfile > bigfile
```

  - Lệnh cat, viết tắt của *concatenate* nối các input để xuất ra một output.

Sử dụng hệ thống file (tt)

- Tạo thư mục (mkdir):

```
$ mkdir <tên/đường dẫn thư mục>
```

- Di chuyển (mv) và sao chép file (cp):

```
$ mv <source> <destination>
$ cp  <source> <destination>
```

- Di chuyển và sao chép với wildcard:
- Di chuyển thư mục:

```
$ mvdir <directory> <destination>
```

Sử dụng hệ thống file (tt)

- Đổi tên file / thư mục:

```
$ mv <tên cũ> <tên mới>
```

- Xóa file:

```
$ rm <file>
```

  - Nhắc trước khi xóa: tùy chọn **–i** (interactive)

```
$ rm –i *duck
rm: remove ‘dead_duck’? Y
rm: remove ‘guiduck’? N
rm: remove ‘lame-duck’? y
```

Sử dụng hệ thống file (tt)

- Xóa thư mục:
  - Dùng lệnh **rmdir** để xóa thư mục:

    ```
    $ rmdir <thư mục>
    ```

  - Chỉ xóa được thư mục đã trống (không còn nội dung)
  - Để xóa toàn bộ nội dung và các thư mục con bên trong (thư mục chưa trống):

    ```
    $ rm -r <thư mục>
    ```

    - Tùy chọn **–r** (recursive) duyệt đệ quy toàn bộ thư mục con bên trong

Sử dụng hệ thống file (tt)

- Giải nén tập tin:

```
$ gunzip onefile.gz      ->   onefile
$ unzip onefile.zip      ->   onefile
```

- Nén tập tin:

```
$ gzip onefile       ->   onefile.gz
$ zip onefile        ->   onefile.zip
```

- Giải nén tập tin tar.gz:

```
$ tar -xvfz onefile.tar.gz
```

Sử dụng hệ thống file (tt)

- Các thư mục quan trọng:
  - Thư mục gốc /
  - /home: Chứa các home directory của các user
  - /bin (binaries): Chứa các chương trình Linux cơ bản
  - /usr: Chứa các phần mềm, tài liệu, …
  - /usr/bin: Chứa các chương trình đã cài đặt
  - /dev: Linux xem mọi thứ là file, đây là nơi chứa các file thiết bị
    - $ cat **/dev/cdrom** > MyCDImage.iso
    - Thiết bị **/dev/null**: Dùng nhận dữ liệu rác

Hệ điều hành Linux

Phần 2: Quản trị Linux I

Quyền truy cập file và thư mục

- Thuộc tính file:

```
$ ls -l /usr/games
-rwxr-xr-x  3 root root  4096 Feb 21 11:45 banner
-r-xrwxr-x  3 root games 4096 Feb 21 11:44 fortune
```

  - 7 trường thuộc tính cho mỗi file:
    - Tập quyền truy xuất (permission)
    - Số liên kết đến file (link count)
    - User ID sở hữu file – owner (ví dụ root)
    - Nhóm user của file (ví dụ games)
    - Kích thước file (tính theo byte)
    - Ngày giờ tạo file
    - Tên file

Quyền truy cập file và thư mục (tt)

- Chủ sở hữu của file (owner):
  - Là user tạo ra file đó:

```
[tuan@pc29 tuan]$ mkdir thumuc1
[tuan@pc29 tuan]$ ls -l
drwxr-xr-x    2 tuan users 4096 11:44 thumuc1
```

    - Chủ sở hữu của **thumuc1** là **tuan**
  - Owner có thể gán quyền truy cập file đó.
  - root có thể đổi owner của file:

```
# chown <chủ sở hữu mới> <file>
[root@pc29 tuan]# chown root thumuc1
[root@pc29 tuan]# ls -l
drwxr-xr-x    2 root users 4096 11:44 thumuc1
```

Quyền truy cập file và thư mục (tt)

- Nhóm người dùng (group):
  - Mỗi group có thể chứa nhiều user.
  - Một user có thể thuộc nhiều group nhưng chỉ có một nhóm chính (*primary group*).
  - Thông thường nhóm chính của mỗi user là một group có trùng tên với User ID, ví dụ: `UID=tuan, Primary GID=tuan`
- Ý nghĩa của group:
  - Dùng khi cần cấp quyền truy cập file cho một nhóm người.

Quyền truy cập file và thư mục (tt)

- Nhóm user của file (group):
  - Là nhóm chính của owner khi tạo file:

```
[tuan@pc29 tuan]$ mkdir thumuc1
[tuan@pc29 tuan]$ ls -l
drwxr-xr-x    2 tuan users 4096 11:44 thumuc1
```

    - Nhóm chính (*primary group*) của **tuan** là **users**
    - Nhóm của **thumuc1** là **users**
  - root có thể đổi nhóm của file:

```
# chgrp <nhóm mới> <file>
[root@pc29 tuan]# chgrp admins thumuc1
[root@pc29 tuan]# ls -l
drwxr-xr-x    2 tuan admins 4096 11:44 thumuc1
```

Quyền truy cập file và thư mục (tt)

- Quyền truy cập file:
  - 3 quyền: **đọc**(**r**), **ghi**(**w**), **thực thi**(**x**)
  - áp dụng cho
  - 3 đối tượng: **owner**, **group**, **other** (các user khác)
- Quyền truy cập thư mục:
  - 3 quyền: **liệt kê**(**r**), **ghi**(**w**), **chuyển vào**(**x**)
  - áp dụng cho
  - 3 đối tượng: **owner**, **group**, **other** (các user khác

Quyền truy cập file và thư mục (tt)

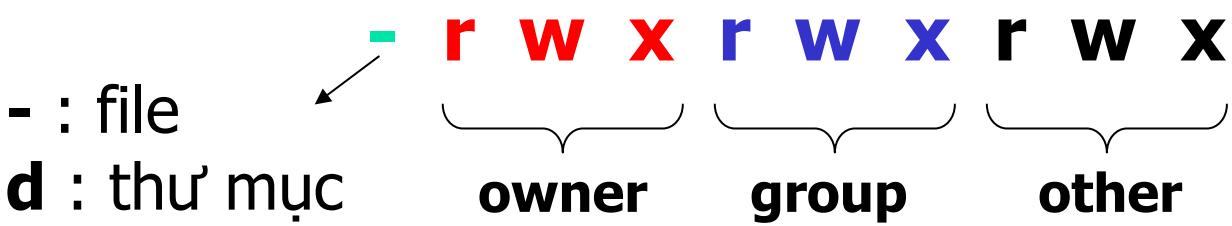
	r	w	x
owner	x	x	x
group	x	x	x
other	x	x	x

Ký hiệu "x":

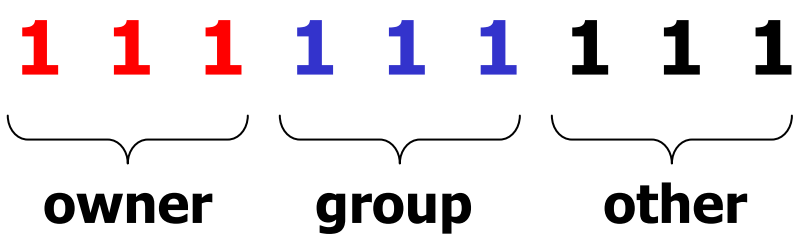
- **Dạng chuỗi quyền:**
  - **r**,**w**,**x**: cho phép
  - **-** : cấm
- **Dạng bit quyền:**
  - **1**: cho phép
  - **0** : cấm

Quyền truy cập file và thư mục (tt)

- => Cần 9 bit để biểu diễn quyền cho một file
- Biểu diễn dạng chuỗi quyền:



- Biểu diễn dạng bit quyền:



Quyền truy cập file và thư mục (tt)

- Biểu diễn quyền ở dạng số:
  - Quy đổi dạng bit quyền sang hệ bát phân.
  - Ví dụ:
    - rwxr-xr-x = 111101101 = 755
      - owner có toàn quyền, các user còn lại chỉ được đọc và thực thi
    - rwx------ = 111000000 = 700
      - owner có toàn quyền, còn lại không có quyền
    - r--rw---- = 100110000 = 460
      - owner chỉ đọc, user trong nhóm được quyền đọc ghi, còn lại không có quyền

Quyền truy cập file và thư mục (tt)

- Đặt quyền truy cập file:

chmod <specification> <file>

  - Dạng specification thứ nhất:

chmod **[u|g|o|a][+|-][r|w|x]** <file>

    - **u**: owner; **g**: group; **o**: other; **a**: cả 3 đối tượng
    - **+**: bật quyền; **-**: tắt quyền
    - **r**: quyền r; **w**: quyền w; **x**: quyền x
  - Dạng specification thứ hai:

chmod **<số bát phân>** <file>

    - Số bát phân biểu diễn chuỗi bit quyền ở trên

Quyền truy cập file và thư mục (tt)

- Một số ví dụ đặt quyền file:

```
$ chmod ugo-rw myfile          hoặc
$ chmod a-rw myfile
```

  - => Tắt quyền đọc/ghi của tất cả user

```
$ ls -l
-rwxrwxrwx 1 root admins 640 05:40 myfile
dr-xr-x-wx 2 root admins 4096 05:40 mydir
$ chmod go-rw my*
$ ls -l
-rwx--x--x 1 root admins 640 05:40 myfile
dr-x--x--x 2 root admins 4096 05:40 mydir
```

  - => Tắt quyền đọc/ghi của các user trong group và user khác

Quyền truy cập file và thư mục (tt)

- Một số ví dụ đặt quyền file: (tt)

```
$ ls -l
-rw-r--r-- 1 tuan users 1024 myfile
$ chmod 755 myfile
$ ls -l
-rwxr-xr-x 1 tuan users 1024 myfile
```

  - => group và other không có quyền ghi

```
$ chmod 644 myfile
$ ls -l
-rw-r--r-- 1 tuan users 1024 myfile
```

  - => tắt quyền ghi của group và other, tắt tất cả quyền thực thi

Quyền truy cập file và thư mục (tt)

- User có quyền đọc file thì cũng có quyền copy file. User đó sẽ là owner của file copy mới:

```
$ whoami
tuan
$ ls -l
-rwxr-xr-x 1 root admins 100 myfile
$ cp myfile ~
$ ls -l ~
-rwxr-xr-x 1 tuan users 100 myfile
```

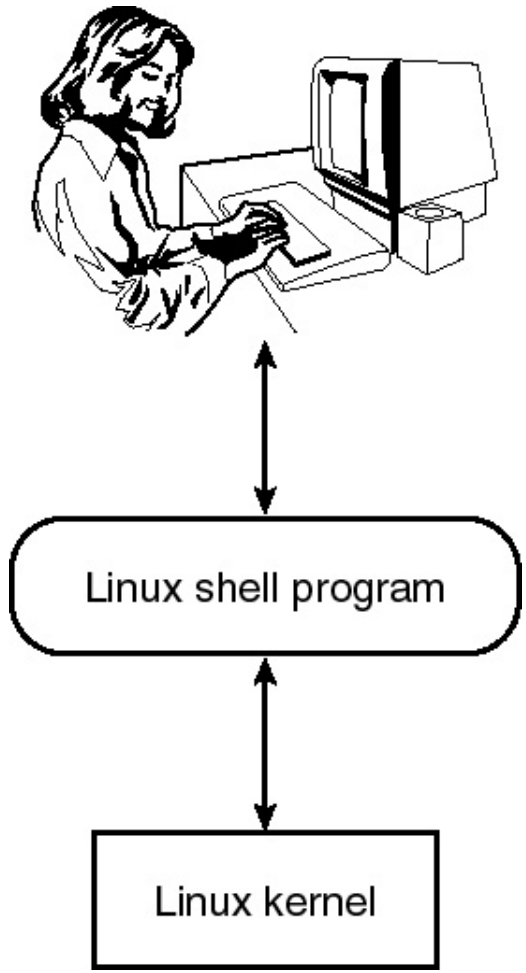
Quyền truy cập file và thư mục (tt)

- User có quyền ghi trong thư mục thì cũng có quyền xóa các file trong thư mục đó, cho dù các file đó có đặt quyền ghi cho user hay không:

```
$ whoami
tuan
$ groups users      => xem danh sách group users
users: duyen nam tuan viet
$ ls -la
drwxrwx--- 2 root users 4096 .
-rw------- 2 root root  916  file01
$ rm file01
```

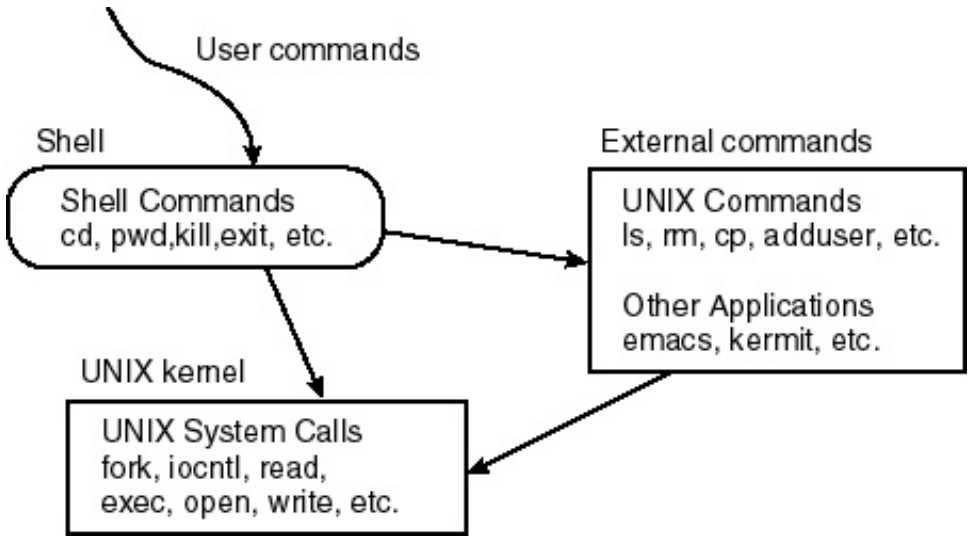
Trình thông dịch lệnh Shell

- Shell là gì?
  - Chương trình **diễn dịch các lệnh** do user nhập từ bàn phím và chuyển cho Linux thực thi.
  - Tựa như trình **command.com** trong MS-DOS.
  - Shell thông dụng trên Linux hiện nay là **Bash** (Bourne Again Shell).



Trình thông dịch lệnh Shell (tt)

- Shell chứa một số lệnh buit-in (tựa như lệnh nội trú trong MS-DOS), ví dụ `pwd`.
- Các lệnh khác đều là những chương trình thực thi nằm trong các thư mục (tựa như lệnh ngoại trú trong MS-DOS), ví dụ `cp`,`mv`.



Trình thông dịch lệnh Shell (tt)

- Sau khi user đăng nhập (login), Linux sẽ chạy một shell mới cho user làm việc.
- Khi user logout, shell của user sẽ kết thúc.
- Shell thường là chương trình đầu tiên được chạy sau khi user login.
- Loại shell làm việc của mỗi user được định nghĩa trong thuộc tính của user account.

Trình thông dịch lệnh Shell (tt)

- Các loại shell truyền thống:
  - **Bourne shell (sh)**: Shell đầu tiên và có mặt trên mọi hệ UNIX, hỗ trợ lập trình shell rất tốt.
  - **C shell (csh)**: Cú pháp lập trình tựa như ngôn ngữ C, hỗ trợ tương tác với user tốt.
  - **Korn shell (ksh)**: Kết hợp các ưu điểm của Bourne shell và C shell, cung cấp môi trường Bourne shell với khả năng hỗ trợ tương tác với user.

Trình thông dịch lệnh Shell (tt)

- Shell trên Linux:
  - **bash**: Phiên bản mở rộng của **sh**
  - **tcsh**: Phiên bản mở rộng của **csh**
  - **pdksh**: Phiên bản mở rộng của **ksh**
- Bourne Again Shell (bash):
  - Là shell mặc định trên Linux
  - Tương thích Bourne shell
  - Bổ sung khả năng xử lý user input:
    - Auto-completion, wildcards, history, alias

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Command-line Auto-completion:
    - Phím TAB, …
  - Sử dụng wildcard:

```
$ ls
ch1.doc ch2.doc ch3.doc ch4.doc ch5.doc
$ lpr ch*                    hay
$ lpr ch?.doc                hay
$ lpr ch[12345].doc          hay
$ lpr ch[1-5].doc
```

    - => In từ file ch1.doc cho đến ch5.doc

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Command history:
    - Lưu lại các lệnh user đã gõ trong file ***HISTFILE*** (mặc định là **.bash_history**) ở home directory của user đó.
    - Kích thước history quy định bởi biến ***HISTSIZE***.
    - User có thể lấy lại các lệnh đã gõ bằng **phím mũi tên lên và xuống**, tương tự như **doskey** trong MS-DOS.
    - Xem history:

      `$ history [số dòng sau cùng]`

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Alias: Thay các lệnh dài bằng tên (alias) ngắn hơn để giảm thao tác gõ phím.

```
$ cd /usr/local/src/gbench1          (1)
```

    - Đặt tên cho lệnh (1) là **gb**:

```
$ alias gb='cd /usr/local/src/gbench-1'
```

    - Sử dụng: `$ gb`
    - Một số alias mặc định trong RedHat bash:
      - alias **ls**='ls **--color**=tty'
      - alias **ll**='**ls -l** --color=tty'
      - alias **mc**='. /usr/share/mc/bin/mc-wrapper.sh'

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Input redirection: Ví dụ:
    - Đếm số dòng, số từ, số ký tự trong một đoạn văn bản (được nhập từ bàn phím):

```
$ wc
Hello World!
Ctrl-D
1     2     13
```

    - Dùng input redirection để đọc đoạn văn bản từ file:

```
$ wc < hello.txt
1     2     13
```

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Output redirection: Chuyển kết quả xuất của lệnh ra file thay vì in ra màn hình. Ví dụ:

```
$ ls > dirlist.txt
```

    - => Liệt kê nội dung thư mục vào file dirlist.txt

```
$ find / -name "*.txt" > result.txt
```

    - => Tìm các file *.txt kể từ thư mục / và in kết quả + thông báo ra file result.txt

```
$ find / -name "*.txt" 2> /dev/null
```

    - => Tìm các file *.txt kể từ thư mục /, chỉ in kết quả ra màn hình, không in thông báo lỗi (kết quả từ thiết bị xuất lỗi chuẩn stderr (2) sẽ bị cắt).

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Pipe (|): Truyền dữ liệu giữa các lệnh – output của lệnh này sẽ là input của lệnh kế. Ví dụ:

```
$ cat vidu.txt
X Window is a GUI of Linux
But Windows is a Microsoft's OS
Openwin is from Sun Solaris
$ cat vidu.txt | grep Window | wc -l
2
```

    - => In số dòng chứa chữ Window trong file vidu.txt

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Prompt: bash dùng 2 mức dấu nhắc:
    - **PS1**: Dấu nhắc lệnh, mặc định là:
      `[\u@\h \W]\$` chẳng hạn: `[tuan@pc29 usr]$`
    - **PS2**: Dấu nhắc chờ nhập dữ liệu, mặc định là **>**
    - Một số ký tự đặc biệt dùng cho prompt:
      - **\$**: Ký tự $ hay # (đối với root)
      - ****: Ký tự \
      - **\d**: Ngày tháng
      - **\h**: Hostname
      - **\n**: Xuống hàng
      - **\s**: Tên shell
      - **\w**: Thư mục hiện hành

Trình thông dịch lệnh Shell (tt)

- Bourne Again Shell (bash): (tt)
  - Thiết lập Prompt cho shell:

    $ PS1="\t\\ "
    - => Dấu nhắc lệnh sẽ là:

    16:43:24\

Quản trị hệ thống Linux

- Gán kết (mount) hệ thống file:
  - Mọi hệ thống file trên Unix/Linux đều phải mount mới có thể sử dụng được.
  - Mount là thao tác gán kết một hệ thống file vào một thư mục nào đó (mount point) trong cây thư mục gốc /.
  - Mọi truy cập đến thư mục mount point được hiểu là truy cập vào hệ thống file đã mount.

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Trong quá trình boot, lệnh mount sẽ được gọi để gán kết tất cả các hệ thống file chỉ định trong file **/etc/fstab**.
  - Ví dụ file /etc/fstab:

```
/dev/hda1       /              ext3
/dev/hda2       swap           swap
/dev/cdrom      /mnt/cdrom     udf,iso9660
/dev/fd0        /mnt/floppy    auto
```

    - => mount hda1 vào thư mục gốc, cdrom vào thư mục /mnt/cdrom, đĩa mềm vào /mnt/floppy

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Định dạng lệnh mount như sau:

    mount <filesystem> <mountpoint>
    - filesystem là tên thiết bị
    - mountpoint là tên thư mục sẽ gán kết vào thiết bị
  - Trường hợp thiết bị chưa được mô tả trong /etc/fstab hoặc Linux không tự động biết:

    mount **–t <fstype>** <filesystem> <mountpoint>
    - **fstype** là kiểu hệ thống file của thiết bị
  - Mount ổ đĩa CDROM vào /mnt/cdrom:

    mount **/dev/cdrom** /mnt/cdrom
    - /dev/**cdrom** là tên thiết bị ổ đĩa CDROM 1

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Mount ổ đĩa mềm vào thư mục /mnt/floppy:

    `$ mount` **`/dev/fd0`** `/mnt/floppy`

    - /dev/**fd0** là tên thiết bị ổ đĩa mềm 1
    - Trường hợp đĩa mềm format FAT12:

    $ mount **–t msdos** /dev/fd0 /mnt/floppy
  - Mount partition Win9x:

    $ mount **–t vfat /dev/hda1** /mnt/win
    - Giả sử /dev/**hda1** là partition cài Windows 9x (ổ C)
    - => Đọc ghi trong thư mục /mnt/win như trên ổ đĩa C: của Windows 9x

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Mount ổ đĩa USB:

    `$ mount` **`/dev/sda1`** `/mnt/usb`
    - /dev/**sda1** là tên thiết bị USB 1 (khe cắm đầu tiên)
    - Trường hợp máy đã có /dev/sda là thiết bị SCSI HDD, tên thiết bị USB sẽ là /dev/sdb, …
  - Mount ổ đĩa SCSI vào thư mục /mnt/temp

    $ mount **/dev/sda1** /mnt/temp
    - /dev/**sda1** là partition1 của ổ đĩa SCSI HDD 1
  - Với lệnh mount, mount point có thể là bất cứ thư mục nào, tuy nhiên thông thường nên đặt trong **/mnt**

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Một số tên thiết bị:
    - /dev/**fd0** là tên thiết bị ổ đĩa mềm 1
    - /dev/**fd1** là tên thiết bị ổ đĩa mềm 2
    - /dev/**cdrom** là tên thiết bị ổ CDROM 1
    - /dev/**cdrom1** là tên thiết bị ổ CDROM 2
    - /dev/**hda1** là partition 1 của HDD 1
    - /dev/**hda2** là partition 2 của HDD 1
    - /dev/**hdb1** là partition 1 của HDD 2
    - /dev/**hdb3** là partition 3 của HDD 2
    - /dev/**sda1** là partition 1 đĩa SCSI 1 hoặc đĩa USB 1
    - /dev/**sdb3** là partition 3 đĩa SCSI 2 hoặc đĩa USB 3

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Các kiểu hệ thống file (**fstype**):
    - **ext2**: hệ thống file trên Linux
    - **ext3**: hệ thống file trên Linux
    - **iso9660**: hệ thống file CDROM
    - **msdos**: hệ thống file FAT
    - **vfat**: hệ thống file FAT
    - **ntfs**: hệ thống file NTFS
    - **nfs**: Network File System
    - **smbfs**: hệ thống file trên các SMB share
    - **tmpfs**: hệ thống file tạm

Quản trị hệ thống Linux (tt)

- Gán kết (mount) hệ thống file: (tt)
  - Gỡ gán kết ổ CDROM:

    $ **umount** /mnt/cdrom hoặc

    $ **eject** umount và đẩy ổ CDROM ra
- Kiểm tra hệ thống file:
  - Nên unmount hệ thống file trước khi kiểm
  - Gọi lệnh fsck:

    $ **fsck** <tên thiết bị>

    $ **fsck -A**

Hệ điều hành Linux

Phần 3: Quản trị Linux II

Thiết bị

- Device (thiết bị):
  - Là những gì gắn vào máy tính chạy Linux và gửi nhận dữ liệu với hệ thống.
  - Ví dụ: Terminal, hard disk, printer, CDROM, modem, …
  - UNIX/Linux đối xử mọi thứ như device.
  - Trình điều khiển thiết bị (device driver) là một phần trong kernel, chứa tập lệnh giao tiếp với device.
  - Device driver trong Linux cho phép tháo lắp dễ dàng, độc lập với phần mềm hệ điều hành.

Thiết bị (tt)

- Có hai loại device trên Linux:
  - **Character device** (thiết bị kí tự): Thiết bị nhập xuất dữ liệu ở dạng "một lần một kí tự". Ví dụ: terminal, printer, modem, …
  - **Block device** (thiết bị khối): Thiết bị nhập xuất dữ liệu theo từng khối lớn dữ liệu, có tốc độ nhanh truyền hơn thiết bị kí tự. Ví dụ hard disk, tape, …
  - Phân biệt character và block device:

```
$ ls -l /dev
crw------- root root beep => c=character
brw-rw---- root floppy fd0 => b=block
```

Thiết bị (tt)

- Số major và minor của thiết bị:
  - Mỗi device được đặc trưng bởi 2 con số này.
  - **Số major** phân biệt các kiểu device (mỗi device driver khác nhau là một kiểu).
  - **Số minor** phân biệt các device cùng kiểu (khi hệ thống gắn nhiều device giống nhau).

Thiết bị (tt)

- Tạo device:

  mknod [option] ***device*** b|c|p|u ***major minor***

  - option:
    - -m [**mode**] đặt mode của file thành **mode** thay vì giá trị mặc định là **0666**.
  - **b**: block mode device, c: character mode device
  - **p**: FIFO device, u: unbuffered character mode device
- Xóa device:
  - Dùng lệnh rm xóa file thông thường

Thiết bị (tt)

- Printer device:
  - Linux hỗ trợ parallel printer và serial printer. Cả hai loại đều là character mode device.
  - Tên cổng máy in (printer device), địa chỉ cổng và tên tương ứng trên MS-DOS:

/dev/**lp0**	0x03BC	**LPT1**
/dev/**lp1**	0x0378	**LPT2**
/dev/**lp2**	0x0278	**LPT3**

    - lp0, lp1, lp2 là tên các cổng song song trên Linux.
    - Địa chỉ cổng có thể thay đổi tùy máy

Thiết bị (tt)

- Tạo printer device:
  - Printer device có thể tạo ra như sau (nếu hệ thống chưa tạo sẵn):

```
$ mknod -m 620 /dev/lp0 c 6 0
```

    - => Tạo **character** device tên /dev/lp0 với số **major=6, minor=0**, quyền truy cập file là **620**
    - major=6 là kiểu thiết bị cổng parallel
    - minor=0 vì đây là device đầu tiên, các device kế tiếp sẽ đánh số tăng dần theo thứ tự.

```
$ chown root.daemon /dev/lp0
```

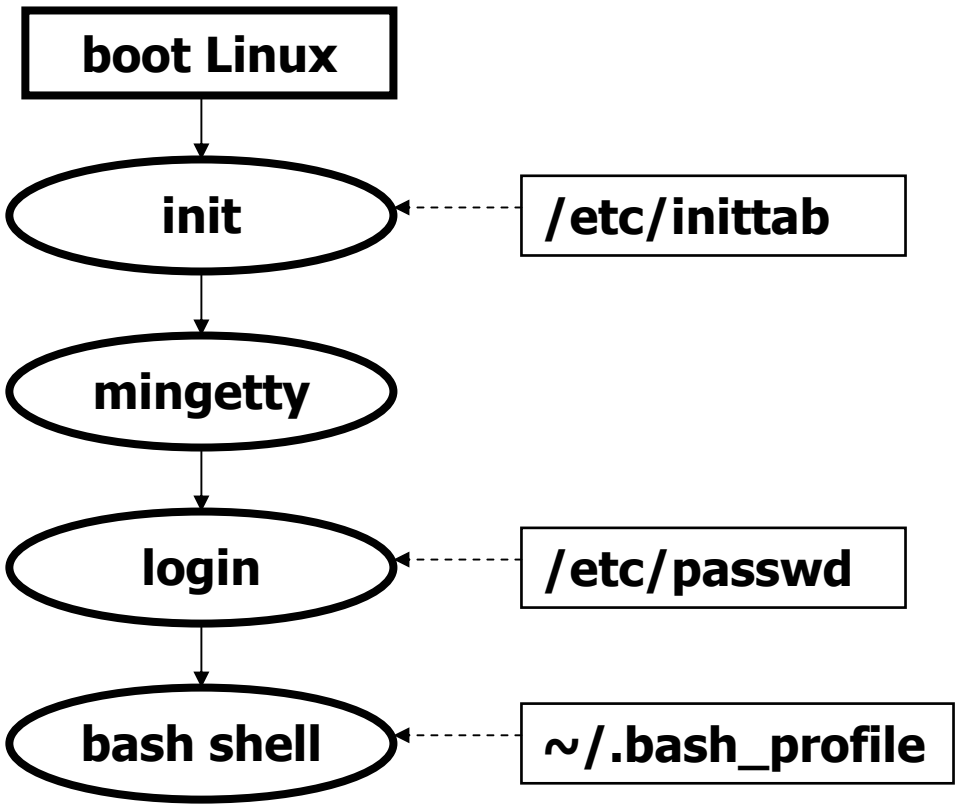
    - => Đặt **owner**=root, **group**=daemon cho file lp0
    - *Lưu ý cách dùng **chown để** gán owner và group.*

Terminal

- Terminal:
  - Là bộ thiết bị giao tiếp giữa người sử dụng với các máy tính UNIX, dữ liệu nhập xuất ở dạng kí tự.
  - Terminal truyền thống:
    - Là thiết bị phần cứng hoặc máy PC chạy phần mềm giả lập (ví dụ *HyperTerminal* trên Windows)
    - Kết nối vào máy tính UNIX/Linux qua cổng serial.
  - Terminal ảo: Chương trình giả lập terminal
    - Kích hoạt qua các tổ hợp phím **Alt-F1** -> **Alt-F6**
    - Thâm nhập qua mạng bằng trình **telnet**, **ssh**

Terminal (tt)

- Login process:



Chọn mức khởi động (**run level**)
Gọi chương trình **getty** để khởi tạo các **terminal**

Thiết lập thông số truyền thông với terminal
Gọi trình **login**

Xác thực username và password
Gán biến môi trường ***TERM***
Sau đó kết thúc và mở **shell**

Terminal (tt)

- Tên cổng serial, địa chỉ cổng và tên tương ứng trên MS-DOS:

/dev/**ttyS0**	0x03F8	**COM1**
/dev/**ttyS1**	0x02F8	**COM2**
/dev/**ttyS2**	0x03E8	**COM3**
/dev/**ttyS3**	0x02E8	**COM4**

  - ttyS0, ttyS1, ttyS2, ttyS3 là tên các cổng serial trên Linux.
  - Địa chỉ cổng có thể thay đổi tùy máy

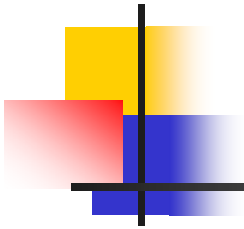
Terminal (tt)

- Tạo terminal device:
  - Terminal có thể tạo ra như sau (nếu hệ thống chưa tạo sẵn):

```
$ mknod -m 660 /dev/ttyS0 c 4 64
```

    - => Tạo **character** device tên /dev/ttyS0 với số **major=4, minor=64**, quyền truy cập file là **660**
    - major=4 là kiểu thiết bị cổng serial
    - minor=64: đây là device đầu tiên, các device kế tiếp sẽ đánh số tăng dần theo thứ tự.

```
$ chown root.tty /dev/ttyS0
```



Process

- Process:
  - Là chương trình đơn, chạy trong một virtual address space riêng của hệ thống.
  - Chạy nhiều process trên dòng lệnh shell:

```
$ cat vidu.txt | grep Window | wc -l
$ cd /usr/games && banner
```

    - => Dấu **&&** dùng để phân cách các lệnh cần chạy tuần tự.

Process (tt)

- Các loại process:
  - Interactive process: Yêu cầu user nhập dữ kiện trong quá trình thực thi, ví dụ: *text editor*, *shell*, *game*, …
  - Batch process: Không tương tác với user khi thực thi, dữ kiện đầu vào được cho trước.
  - Daemon process: thường được kích hoạt vào lúc boot máy, chạy ở chế độ background.

Process (tt)

- Lệnh ps (process status):
  - Xem chi tiết các process của mình:

```
$ ps -u
USER PID  %CPU %MEM VSZ  RSS  TTY  STAT START TIME COMMAND
tuan 13814 1.5  0.1 4344 1432 pts/0  S  18:00 0:00 -bash
tuan 13851 0.0  0.0 2600 648  pts/0  R  18:00 0:00 ps -u
```

  - Xem các process của một user nào đó:

```
# ps -u tuan
```

  - Xem chi tiết tất cả process trong hệ thống:

```
# ps -aux
```

Quản trị người dùng

- Tài khoản superuser:
  - Là user account có **UID = 0**, thường là **root**.
  - Nên tạo từng account cho từng công việc quản trị để tránh đăng nhập vào root.
- File **/etc/passwd**:
  - File text lưu cơ sở dữ liệu user account trên hệ thống.
  - Chỉ có root mới được quyền ghi lên file này.

Quản trị người dùng (tt)

- File **/etc/passwd**: (tt)
  - Định dạng mỗi dòng trong file:

    `username:password:UID:GID:`
    `comment:home_directory:login_command`
    - password: Là mã hóa của user password
    - home_directory: Thường là /home/<username>
    - login_command: Chương trình cần gọi ngay sau khi login thành công
  - Lưu ý:
    - Hệ thống không lưu trữ trực tiếp user password.
    - Hệ thống chỉ có thể kiểm chứng user password chứ không thể giải mã được user password.

Quản trị người dùng (tt)

- Ví dụ file **/etc/passwd**:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
sync:x:5:0:/sbin:/sbin/nologin
halt:x:7:0:halt:/sbin:/sbin/halt
rpm:x:37:37::/var/lib/rpm:/bin/bash
uucp:x:10:14:uucp:/var/spool/uucp:
nobody:x:99:99:Nobody:/:/sbin/nologin
ftp:x:14:50:FTP:/var/ftp:/sbin/nologin
tuan:x:500:500::/home/tuan:/bin/bash
```

Quản trị người dùng (tt)

- Trường password trong /etc/passwd:
  - Để trống: Account có password rỗng.
  - **x**: Password (mã hóa) lưu trong /etc/shadow.
- File **/etc/shadow**:
  - Lưu các password đã mã hóa của mỗi account
  - Chỉ có root mới được quyền đọc ghi file này
  - Ví dụ dòng chứa password của account tuan trong /etc/shadow:

```
tuan:$1$/347I5oy$cfG13OXnR3tNKmRvraEyG.
  :12560:0:99999:7:::
```

Quản trị người dùng (tt)

- Các account hệ thống:
  - root: superuser account (UID = 0)
  - daemon: Dùng để chạy các system process
  - bin: Owner cho các file thực thi
  - ftp: Dùng trong kết nối FTP
- Trường password của các account như daemon, bin, ftp là một dấu *, thể hiện user không được login bằng account này.

Quản trị người dùng (tt)

- Bổ sung user mới vào hệ thống: 2 cách
  - Sửa trực tiếp file **/etc/passwd**:
    - Soạn thảo file này và bổ sung 1 dòng cho user mới
    - Tạo home directory và đặt quyền sở hữu cho user
    - Copy các file khởi động (.bash_profile, …) vào home directory và đặt quyền sở hữu cho user.
    - Lưu ý: Sao lưu lại và cẩn thận khi dùng cách này
  - Sử dụng lệnh **useradd**:

```
# useradd tuan
```

    - => Tạo user account **tuan** với các giá trị mặc định:
      - home directory là **/home/tuan**
      - login command (shell) là **/bin/bash**
      - Nhóm chính (primary group) là **tuan**, không có nhóm phụ.

Quản trị người dùng (tt)

- Thêm thông số cho useradd:

```
# useradd -d /tuan -s /bin/sh tuan
```

  - => Tạo user tuan có home directory là **/tuan** và shell làm việc là **Bourne shell** (/bin/sh)

```
# useradd -g user tuan
```

  - => Tạo user tuan và đặt nhóm chính (primary group) của tuan là nhóm **user**

```
# useradd -g user -G teacher,admin tuan
```

  - => Tạo user tuan, đặt nhóm chính của tuan là **user**, nhóm phụ là **teacher** và **admin**

- File **/etc/group**: Cơ sở dữ liệu user group

Quản trị người dùng (tt)

- Sửa đổi thông tin user: 2 cách
  - Sửa trực tiếp file **/etc/passwd, /etc/group**
  - Sử dụng lệnh **usermod**:

```
# usermod -c "Nguyen Van Tuan" tuan
```

    - => Đặt thông tin comment cho account tuan

```
# usermod -G students,operators tuan
```

    - => Đặt nhóm phụ của tuan là **students và operator**

```
# usermod -g admins tuan
```

    - => Đổi nhóm chính của tuan thành **admins**

Quản trị người dùng (tt)

- Xóa user khỏi hệ thống: 2 cách
  - Sửa trực tiếp file **/etc/passwd**:
    - Xóa dòng thông tin user trong file /etc/passwd
    - Xóa dòng thông tin group trong file /etc/group
    - Xóa thư mục làm việc của user
  - Sử dụng lệnh **userdel**:

```
# userdel tuan
```

    - => Xóa tuan khỏi hệ thống (xóa cả group tuan)

```
# userdel -r tuan
```

    - => Xóa user tuan và xóa cả home dir của tuan

Quản trị người dùng (tt)

- File **/etc/group**: (tt)
  - Định dạng mỗi dòng trong file:

    group_name:password:GID:users_list

    - password: Thường là dấu *, x hoặc để trống, chỉ sử dụng ở một số hệ thống yêu cầu user nhập password khi tham gia nhóm.
    - users_list: Danh sách các user thành viên của nhóm, cách nhau bởi dấu phẩy (,).
  - Ví dụ file /etc/group:

    root:x:0:root

    bin:x:1:root,bin,daemon

    tuan:x:500:

Quản trị người dùng (tt)

- Thêm, sửa, xóa nhóm người dùng: 2 cách
  - Sửa trực tiếp file **/etc/group**
  - Dùng lệnh **groupadd**, **groupmod**, **groupdel**:

    ```
    # groupadd students
    ```

    - => Thêm nhóm students vào hệ thống

    ```
    # groupmod -n sinhvien students
    ```

    - => Đổi tên nhóm students thành sinhvien

    ```
    # groupdel sinhvien
    ```

    - => Xóa nhóm sinhvien
- Đưa thành viên vào group:
  - Dùng lệnh **usermod** đã trình bày ở trên

Cấu hình RedHat với setup

- File cấu hình trong Linux:
  - Hầu hết dữ liệu cấu hình trong Linux và các ứng dụng đều lưu trong các file cấu hình.
  - File cấu hình thường là file văn bản, có phần mở rộng **.conf** và lưu trong thư mục **/etc**.
  - Cấu hình của hệ thống được thiết lập bằng cách hiệu chỉnh các file cấu hình.
- Tiện tích **setup** trong RedHat:
  - Dùng để thiết lập các cấu hình cơ bản cho hệ thống thông qua giao diện menu gần gũi.
  - Trình setup sẽ hiệu chỉnh các thông số trong các file cấu hình tương ứng của hệ thống.

Cấu hình RedHat với setup (tt)

Cấu hình RedHat với setup (tt)

- Trình **setup** có 7 mục cấu hình sau:
  - Xác thực (Authentification): **authconfig**
  - Tường lửa (Firewall): **lokkit**
  - Thiết bị chuột (Mouse): **mouseconfig**
  - Mạng (Network): **netconfig**
  - Máy in (Printer): printconf
  - Dịch vụ hệ thống (System Services): **ntsysv**
  - Ngày giờ hệ thống (Timezone): **timeconfig**
- Mỗi mục tương ứng với một chương trình cấu hình có tên nêu trên.

Cấu hình RedHat với setup (tt)

- Cấu hình xác thực người dùng:
  - Xác thực qua **NIS**: Chọn mục này nếu cơ sở dữ liệu user được lưu trữ tập trung trong một NIS server trên mạng.
  - Xác thực qua **LDAP**: Chọn mục này nếu cơ sở dữ liệu user được lưu trữ tập trung trong một LDAP server trên mạng.
  - Xác thực qua **Samba**: Chọn mục này nếu muốn máy Linux logon vào NT Domain như một máy trạm Windows NT.

Cấu hình RedHat với setup (tt)

- Cấu hình Firewall:
  - Chọn **High**, **Medium** hoặc **No firewall** tùy theo nhu cầu về an ninh của hệ thống.
    - Ví dụ các máy nối trực tiếp với Internet cần dùng High firewall, các máy trạm trong mạng LAN được bảo vệ có thể không cần firewall.
  - Chọn **Customize** trong trường hợp cần thiết lập firewall nhưng vẫn cho phép truy cập vào một số cổng TCP/IP nào đó như HTTP, FTP, SSH.

Cấu hình RedHat với setup (tt)

- Cấu hình Mouse:
  - Chọn đúng loại mouse đang dùng
- Cấu hình Printer:
  - Chọn **printer queue** và cấu hình máy in
- Cấu hình Timezone:
  - Chọn time zone địa phương
- Cấu hình System Sevices:
  - Chọn các service (thường là **daemon**) sẽ chạy tự động khi boot máy

Cấu hình RedHat với setup (tt)

- Một số service quan trọng:
  - **autofs**: Dịch vụ mount tự động khi boot
  - **crond**: Dịch vụ định thời thực thi
  - **cups**: Dịch vụ in ấn
  - **dhcpd**: DHCP Server - cấp địa chỉ IP động
  - **httpd**: Apache Web Server
  - **iptables**: Firewall
  - **kudzu**: Dịch vụ nhận diện phần cứng
  - **ldap**: LDAP Server – dịch vụ thư mục

Cấu hình RedHat với setup (tt)

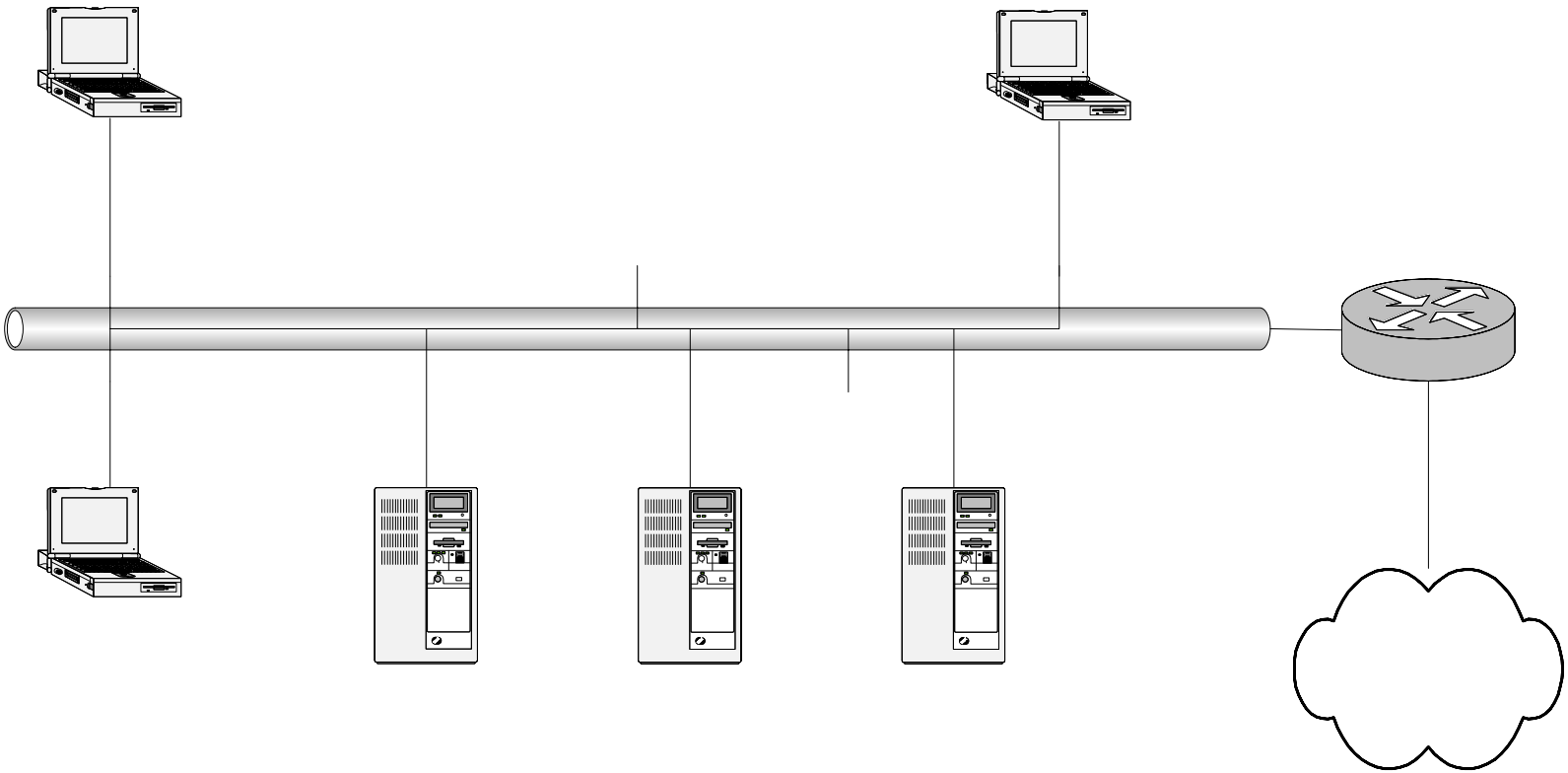
- Một số service quan trọng: (tt)
  - **named**: DNS Server – phân giải tên miền
  - **network**: Kích hoạt các thiết bị mạng
  - **nfs**: NFS Server – Network File System
  - **smb**: Samba – kết nối Microsoft Network
  - **squid**: Proxy Server
  - **sshd**: SSH Server – cho phép truy cập từ xa
  - **telnet**: Telnet Server – cho truy cập từ xa
  - **vsftpd**: FTP Server
  - **xinetd**: Quản lý các dịch vụ Internet

Cấu hình RedHat với setup (tt)

- Cấu hình Network:
  - Mỗi máy phải thiết lập các thông số:
    - **Địa chỉ IP** theo quy định của người quản trị mạng LAN (địa chỉ thuộc lớp A, B hay C; netmask là bao nhiêu).
    - **Gateway**: Thiết bị/server đóng vai trò cửa ngõ kết nối ra các mạng LAN/WAN/Internet bên ngoài.
    - Primary **name server**: Để phân giải tên miền ra IP
      - Ví dụ phân giải www.yahoo.com -> 66.94.230.37
    - **Secondary** name server: Server dự phòng cho Primary name server
    - **Domain name** (tên miền): Tên máy và tên miền

Cấu hình RedHat với setup (tt)

- Cấu hình Network (tt): Ví dụ



Cấu hình RedHat với setup (tt)

- Cấu hình Network: Các thông số mạng (IP, netmask, gateway, DNS, domain) được thiết lập bằng 2 phương pháp
  - Sử dụng địa chỉ IP tĩnh:
    - Thiết lập bằng tay các thông số trên.
    - Phải liên hệ với quản trị mạng LAN mỗi khi thiết lập, thay đổi cấu hình mạng cho máy.
    - Các máy trạm phải thiết lập lại cấu hình khi có sự thay đổi về cấu hình mạng LAN.

Cấu hình RedHat với setup (tt)

- Cấu hình Network: (tt)
  - Sử dụng địa chỉ IP động:
    - Các thông số cấu hình mạng được tự động thiết lập thông qua một DHCP Server.
    - Người dùng không cần quan tâm đến việc thiết lập các thông số trên.
    - Khi mạng LAN thay đổi cấu hình, các máy trạm sẽ tự động được DHCP Server cập nhật cấu hình.
    - => Phải duy trì một **DHCP Server** (và có thể cả một DHCP Server dự phòng) trên mạng LAN.
    - => Chọn option "Use dynamic IP configuration (DHCP)" trong phần cấu hình network của **setup**.

Cấu hình RedHat với setup (tt)

- Cấu hình Network:
  - Xem thông tin cấu hình mạng: Lệnh **ifconfig**

```
# ifconfig
eth0    Link encap:Ethernet  HWaddr 00:06:B5:8C:76:0C
        inet addr:10.1.1.12  Bcast:10.1.255.255  Mask:255.255.0.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
eth1    Link encap:Ethernet  HWaddr 00:03:74:E0:C4:17
        inet addr:10.2.1.12  Bcast:10.2.255.255  Mask:255.255.0.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

    - => Xem thông tin tất cả thiết bị mạng

```
# ifconfig eth0
```

    - => Xem thông tin card mạng 1

```
# ifconfig eth1
```

    - => Xem thông tin card mạng 2

Cấu hình RedHat với setup (tt)

- Cấu hình Network: (tt)
  - Ngoài phương pháp dùng setup để cấu hình mạng, có thể hiệu chỉnh trực tiếp các file cấu hình ở **/etc/sysconfig/network-scripts**:
    - File **ifcfg-eth0**: Cấu hình cho thiết bị **eth0**

```
DEVICE=eth0
IPADDR=10.1.1.12
NETMASK=255.255.0.0
NETWORK=10.1.0.0
ONBOOT=yes
HWADDR=00:06:5b:8c:c4:17
```

Cấu hình RedHat với setup (tt)

- Cấu hình Network: (tt)
  - Kích hoạt thiết bị mạng eth0:

    `# ifconfig eth0` **`up`** hoặc

    `# ifup eth0`
  - Tạm ngừng thiết bị mạng eth0:

    `# ifconfig eth0` **`down`** hoặc

    `# ifdown eth0`
  - Ngoài ra, lệnh **ifconfig** còn cho phép thiết lập các thông số cấu hình mạng và nhiều lựa chọn khác.

Thiết lập các service

- Dịch vụ hệ thống (service):
  - Các **service** được thực thi thông qua các script đặt ở thư mục **/etc/rc.d/init.d/**
  - Ngoài ra còn có 7 thư mục **rc0.d ... rc6.d** trong thư mục **/etc/rc.d/** chứa các liên kết đến những script này.
- Các service cho từng runlevel:
  - Linux có **7 mức runlevel**, khi boot máy vào runlevel nào, Linux sẽ thực thi các service trong thư mục **rc0.d .. rc6.d** tương ứng với runlevel đó.

Thiết lập các service (tt)

- Ví dụ: Dịch vụ web server **httpd**
  - File script: /etc/rc.d/init.d/**httpd**
  - Các liên kết ở thư mục /etc/rc.d/rc0.d .. rc6.d:
    - `/etc/rc.d/rc3.d/`**`K15httpd`**
      `->` **`/etc/rc.d/init.d/httpd`**
    - `/etc/rc.d/rc5.d/`**`S85httpd`**
      `->` **`/etc/rc.d/init.d/httpd`**
    - `/etc/rc.d/rc1.d/`**`K15httpd`**
      `->` **`/etc/rc.d/init.d/httpd`**
  - Kí hiệu K - độ ưu tiên kết thúc; S - bắt đầu

Thiết lập các service (tt)

- Bổ sung service mới:
  - Tạo hoặc copy file script của service vào /etc/rc.d/init.d: Ví dụ file sshd

```
#!/bin/bash
# Init file for OpenSSH server daemon
# chkconfig: 2345 55 25
```

..... Nội dung file script .....

  - Dòng chkconfig bắt buộc phải có, ý nghĩa như sau:
  - **2345**: sshd chạy ở các runlevel 2, 3, 4, 5
  - **55**: sshd có độ ưu tiên bắt đầu 55 (trong 100 service thì sshd là service thứ 55 được start)
  - **25**: sshd có độ ưu tiên kết thúc là 25

Thiết lập các service (tt)

- Bổ sung service mới: (tt)
  - Dùng lệnh **chkconfig** để tạo các liên kết tương ứng trong /etc/rc.d/rc0.d … rc6.d:

    ```
    # chkconfig --add <service name>
    ```

  - Kích hoạt để service tự động chạy khi boot:

    ```
    # chkconfig <service name> on
    ```

    - Có thể dùng trình **setup**, mục System Service để kích hoạt service (đã trình bày ở phần trước).
  - Đặt lại runlevel cho service (nếu cần):

    ```
    # chkconfig --level <levels> <service>
    ```

Thiết lập các service (tt)

- Xem thiết lập của service:
  - **chkconfig --list <service name>**

```
# chkconfig --list sshd
sshd        0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

- Ngưng kích hoạt service:
  - **chkconfig <service name> off**
- Xóa bỏ service:
  - chkconfig --del <service name>

```
# chkconfig --del sshd
```

Cài đặt phần mềm trên RedHat

- Phần mềm dạng source:
  - Thường được đóng gói và nén dưới dạng **.tar.gz**

```
cd <software_directory>
```

configure => Cấu hình phần mềm

make => Biên dịch phần mềm

make install => Cài đặt phần mềm vào các thư mục hệ thống

  - Hệ thống phải có cài GCC (Gnome C Compiler)
  - Đọc file README, INSTALL của phần mềm để biết chính xác về cách cài đặt.

Cài đặt phần mềm trên RedHat (tt)

- Phần mềm dạng RPM binary:
  - Gói phần mềm có phần mở rộng tên file:
    - **. i386.rpm, .i686.rpm => Kiến trúc Intel**
    - **.sparc.rpm => Kiến trúc Sun**
  - Cài đặt mới:

    ```
    # rpm -ivh <RPM_file>
    ```
  - Nâng cấp:

    ```
    # rpm -Uvh <RPM_file>
    ```
  - Các tùy chọn quan trọng:
    - **--nodeps**: bỏ qua các gói RPM liên quan
    - **--force**: buộc phải cài đặt

Cài đặt phần mềm trên RedHat (tt)

- Phần mềm dạng RPM binary: (tt)
  - Xem thông tin phần mềm đã cài:

    # **rpm -q <software_name>**

    # **rpm -qi <software_name>**
  - Gỡ bỏ phần mềm (uninstall):

    # **rpm –e <software>**
    - <software> gồm tên và version của phần mềm, nếu không nhớ thì dùng rpm –q để xác định.
  - Một số lưu ý khi cài đặt phần mềm RPM:
    - Nếu có thông báo lỗi về dependancy giữa các gói RPM, dùng tùy chọn **--nodeps** để bỏ qua.

Cài đặt phần mềm trên RedHat (tt)

- Phần mềm dạng RPM source:
  - Gói phần mềm có phần mở rộng tên file:
    - **.src.rpm** => Dạng source code
  - Dịch RPM source sang RPM binary rồi cài đặt:

```
# rpm -ivh <RPM_source_file>
# cd /usr/src/redhat/SPECS
# rpmbuild -bb <specs_file.spec>
# cd /usr/src/redhat/RPMS/i386   ;Với Intel
# rpm -ivh <RPM_file>
```

  - Điều chỉnh **file specs** (nếu cần thêm tùy chọn biên dịch) trước khi gọi rpmbuild

Hệ điều hành Linux

Phần 4: Lập trình Shell

Chương trình Shell

- Chương trình Shell:
  - Là file văn bản chứa một hoặc nhiều lệnh Linux cần thực thi một lượt.
  - Công dụng:
    - Chạy nhiều lệnh được dùng thường xuyên bằng một lệnh đơn (chương trình shell)
    - Tự động hóa công đoạn cài đặt phần mềm
    - Viết các chương trình ứng dụng đơn giản
  - Chương trình Shell được viết bằng các trình soạn thảo văn bản thông thường như vi, ...

Tạo chương trình Shell

- Ví dụ: Chương trình backup nội dung thư mục làm việc vào đĩa mềm:
  - Soạn thảo chương trình shell:

```
# cat > backup
mount -t msdos /dev/fd0 /mnt/floppy
cp ~/* -R /mnt/floppy
```

  - Gán quyền thực thi:

```
# chmod a+x backup
```

  - Chạy chương trình backup:

```
# ./backup        hoặc # source backup
```

Sử dụng biến

- Biến (**variable**) là phần tử nhớ chứa dữ liệu dùng khi thực thi chương trình shell.
- Hai loại biến:
  - Biến do chương trình shell tự khai báo, biến môi trường, ví dụ PATH
  - Biến quy ước của shell, ví dụ: $1, $2, …
- Khai báo biến:
  - Không cần khai báo trước
  - Biến của shell không có kiểu dữ liệu: một biến có thể lưu hoặc chuỗi kí tự hoặc số nguyên, …

Sử dụng biến (tt)

- Định nghĩa biến và gán giá trị: ví dụ

  `COUNT=5` -> Khai báo biến COUNT giá trị 5

  - Lưu ý: với Bash shell, không được có khoảng trắng cạnh dấu "="
  - Một cách gán khác: **`set COUNT = 5`**
- Lấy giá trị của biến:
  - Dùng dấu $ trước tên biến
  - Ví dụ in giá trị biến COUNT ra màn hình:

  `echo` **`$COUNT`**

Sử dụng biến (tt)

- Biến tham số dòng lệnh:
  - Là các biến lưu các tham số được truyền trên dòng lệnh khi gọi chương trình shell.
  - Quy tắc đặt tên:
    - `1`: Tham số thứ 1
    - `2`: Tham số thứ 2
    - `3`: Tham số thứ 3
    - …
  - Lấy giá trị: cũng dùng dấu $ trước tên biến
    - `$1`, `$2`, `$3`, …

Sử dụng biến (tt)

- Biến tham số dòng lệnh (tt): ví dụ
  - Chương trình shell tạo 2 user account:

```
# vi create-two
useradd $1
useradd $2
passwd $1
passwd $2
```

  - Thực thi chương trình:

```
# create-two huyen hieu
```

    - => Tạo user **huyen** và **hieu**, sau đó đặt password

Sử dụng biến (tt)

- Một số biến quy ước trong Shell:
  - **$#**: Số lượng tham số dòng lệnh
  - **$?**: Mã thoát của lệnh vừa thực thi
  - **$0**: Tên chương trình shell (tham số 0)
  - **$***: Danh sách các tham số dòng lệnh ($1 $2 …)
  - **"$@"**: Danh sách các tham số dòng lệnh, đặt trong dấu nháy ("$1" "$2" …)

Sử dụng dấu nháy

- Dấu nháy kép (“”): biểu diễn chuỗi ký tự có chứa khoảng trắng

```
$ mystring=“Hello World”
$ echo $mystring        => In biến mystring
Hello World
```

  - Bash sẽ báo lỗi khi không dùng nháy kép cho chuỗi:

```
$ mystring=Hello World
-bash: World: command not found
```

  - => mystring vẫn không được gán giá trị

Sử dụng dấu nháy (tt)

- Dấu nháy đơn (‘’): biểu diễn chuỗi kí tự có chứa khoảng trắng hay ký tự đặc biệt khác

```
$ mystring='Hello World'
$ echo $mystring
Hello World
```

- Ý nghĩa của dấu nháy kép: che ký tự khoảng trắng trong chuỗi
- Ý nghĩa của dấu nháy đơn: che tất cả ký tự đặc biệt trong chuỗi (gồm cả khoảng trắng)

Sử dụng dấu nháy (tt)

- Khi nào sử dụng dấu nháy kép:

```
$ mystring="Hello, I am $LOGNAME"
$ echo $mystring
Hello, I am tuan
```

  - => Nháy kép không che ký tự $ nên được thay thế
- Khi nào sử dụng dấu nháy đơn:

```
$ mystring='Hello, I am $LOGNAME'
$ echo $mystring
Hello, I am $LOGNAME
```

  - => Nháy đơn che ký tự $ nên không thay thế

Sử dụng dấu nháy (tt)

- Dấu sổ ngược \ (backslash): Dùng để che một ký tự đặc biệt.

```
$ mystring=Hello\ World
$ echo $mystring
Hello World
```

  - => Dấu backslash che ký tự khoảng trắng

```
$ cost=\$2500
$ echo $cost
$2500
```

  - => Dấu backslash che ký tự $

Sử dụng dấu nháy (tt)

- Dấu nháy ngược (``): Dùng để lấy kết quả thực thi của lệnh đặt bên trong.

```
$ dir_content=`ls -la /home/tuan`
$ echo $dir_content
drwxr-x---  22 tuan tuan  4096 Jun  3 15:57 .
drwxr-xr-x  25 tuan tuan  4096 Jun  3 16:49 ..
-rw-------   1 tuan tuan 16595 Jun  4 19:44 .bash_history
-rw-r--r--   1 tuan tuan    24 Jun 11  2000 .bash_logout
-rw-r--r--   1 tuan tuan   271 Jun  3 10:09 .bash_profile
-rw-r--r--   1 tuan tuan   176 Aug 24  1995 .bashrc
```

  - => Thực thi lệnh ls và gán kết quả cho dir_content

Lệnh test

- Dùng để đánh giá một biểu thức điều kiện
- Cú pháp: 2 dạng
  - **`test` *`expression`*** hay
  - **`[` *`expression`* `]`**
- Có 4 nhóm toán tử dùng trong *expression*:
  - Toán tử số nguyên
  - Toán tử chuỗi ký tự
  - Toán tử file
  - Toán tử logic

Lệnh test (tt)

- Toán tử số nguyên:

expression	Ý nghĩa
`int1 -eq int2`	= true nếu int1 = int2
`int1 -ge int2`	= true nếu int1 ≥ int2
`int1 -gt int2`	= true nếu int1 > int2
`int1 -le int2`	= true nếu int1 ≤ int2
`int1 -lt int2`	= true nếu int1 < int2
`int1 -ne int2`	= true nếu int1 ≠ int2

Lệnh test (tt)

- Toán tử chuỗi:

expression	Ý nghĩa
`str1 = str2`	= true nếu str1 = str2
`str1 != str2`	= true nếu str1 ≠ str2
`str`	= true nếu str ≠ null
`-n str`	= true nếu str length > 0
`-z str`	= true nếu str length = 0

Lệnh test (tt)

- Toán tử file:

expression	Ý nghĩa
-d filename	= true nếu là thư mục
-f filename	= true nếu là file thường
-r filename	= true nếu file đọc được
-s filename	= true nếu kích thước khác 0
-w filename	= true nếu file ghi được
-x filename	= true nếu file thực thi được

Lệnh test (tt)

- Toán tử logic:

expression	Ý nghĩa
`! expr`	= true nếu expr = false
`expr1 -a expr2`	= true khi expr1 và expr2 true
`expr1 -o expr2`	=true khi expr1 hay expr2 true

Cấu trúc điều kiện

- Phát biểu **if**: dạng 1

```
if [ expression ]
then
  commands
fi
```

- Phát biểu **if**: dạng 2

```
if [ expression ]
then
  commands
else
  commands
fi
```

Cấu trúc điều kiện (tt)

- Phát biểu **if**: dạng 3

```
if [ expression ]
then
  commands
elif [ expression2 ]
then
  commands
else
  commands
fi
```

Cấu trúc điều kiện (tt)

- Ví dụ phát biểu **if**:
  - File **hello**: Hiển thị câu hello

```
if [ "$1" = "you" ]
then
  echo "Hi, how are you?"
else
  echo "Hello $1"
fi
```

  - Lưu ý:
    - Phải có khoảng trắng cạnh dấu [,], =
    - "$1": chuỗi ký tự tham số dòng lệnh thứ nhất
    - Từ khóa **then** phải viết xuống hàng

Cấu trúc điều kiện (tt)

- Ví dụ phát biểu **if**: (tt)
  - File **helpme**: kiểm tra và đọc README.TXT

```
if [ -r README.TXT ]
then
  less README.TXT
elif [ ! -f README.TXT ]
then
  echo "Sorry, no help file!"
else
  echo "Sorry, no read permission!"
fi
```

Cấu trúc điều kiện (tt)

- Phát biểu **case**:

```
case str1 in
  string1)
   commands;;
  string2)
   commands;;
  *)
   commands;;
esac
```

  - Phát biểu case trong shell mạnh hơn C, Pascal ở chỗ có thể so sánh chuỗi với các ký tự wild card.

Cấu trúc điều kiện (tt)

- Ví dụ phát biểu **case**:
  - File **readfile**: đọc nội dung một file

```
case $3 in
 first)
     head -n $2 $1;;
 last)
     tail -n $2 $1;;
 *)
     cat $1;;
esac
```

  - Ví dụ, đọc 10 dòng cuối file README.TXT:

```
$ readfile README.TXT 10 last
```

Cấu trúc lặp

- Phát biểu **for**: dạng 1

```
for var1 in list1
do
  commands
done
```

  - **list1**: danh sách các giá trị cách nhau bởi khoảng trắng
    - **list1** có thể là một biến hoặc một danh sách được nhập trực tiếp trong câu lệnh.
  - **var1**: Lần lượt nhận các giá trị trong **list1** tại mỗi lần lặp

Cấu trúc lặp (tt)

- Phát biểu **for**: dạng 2

```
for var1
do
  commands
done
```

  - **var1**: nhận giá trị từ danh sách tham số dòng lệnh
  - Dạng phát biểu for này tương đương với:

```
for var1 in "$@"
…
```

Cấu trúc lặp (tt)

- Ví dụ phát biểu **for**:
  - File **uppercase**: đổi sang chữ hoa nội dung các file văn bản (tên file nhập từ dòng lệnh)

```
for file
do
  tr a-z A-Z < $file > $file.caps
done
```

  - Lệnh **tr**: Chuyển đổi các ký tự (*a..z sang A..Z*) của chuỗi nhập từ bàn phím và in ra màn hình
  - Ví dụ: `$ uppercase vb1 vb2 vb3`
    - => **vb1.caps, vb2.caps, vb3.caps**

Cấu trúc lặp (tt)

- Ví dụ phát biểu **for**: (tt)
  - File **uppercase2**: version 2 của uppercase – danh sách file có sẵn (vb1, vb2, vb3)

```
for file in vb1 vb2 vb3
do
  tr a-z A-Z < $file > $file.caps
done
```

  - => Dùng dạng 1 của phát biểu for.
  - Lưu ý không có dấu nháy "" trong danh sách

Cấu trúc lặp (tt)

- Ví dụ phát biểu **for**: (tt)
  - File **uppercase3**: version 3 của uppercase – danh sách file là một biến

```
list="vb1 vb2 vb3"
for file in $list
do
  tr a-z A-Z < $file > $file.caps
done
```

  - => Cũng dùng dạng 1 của phát biểu for.

Cấu trúc lặp (tt)

- Phát biểu **while**:

```
while expression
do
   commands
done
```

  - **expression**: biểu thức luận lý (giá trị true hay false)
  - Vòng while lặp khi expression = **true**
- Lệnh **expr**: Đánh giá một biểu thức toán
  - Ví dụ: $ **expr 100 / 5 + 1** => In ra **21**
- Lệnh **shift**: Dịch chuyển tham số dòng lệnh qua trái ($2 => $1, $3 => $2, $4 => $3, ...)

Cấu trúc lặp (tt)

- Ví dụ phát biểu **while**:
  - File **param**: In thứ tự và giá trị các tham số trên dòng lệnh

```
count=1
while [ -n "$*" ]
do
  echo "Tham so thu $count = $1"
  shift
  count=`expr $count + 1`
done
```

  - Mỗi khi gọi **shift**, biến $1 sẽ lần lượt mang giá trị tham số kế tiếp

Cấu trúc lặp (tt)

- Phát biểu **until**:

```
until expression
do
  commands
done
```

  - **expression**: biểu thức luận lý (giá trị true hay false)
  - Vòng until lặp khi expression = **false**
  - Lệnh **break**: thoát khỏi vòng lặp for, while, until, select

Cấu trúc lặp (tt)

- Ví dụ phát biểu **until**:
  - File **param2**: version "until" của **param**

```
count=1
until [ -z "$*" ]
do
  echo "Tham so thu $count = $1"
  shift
  count=`expr $count + 1`
done
```

    - expression -z "$*" trả về true nếu danh sách tham số dòng lệnh rỗng (đã shift hết tham số)

Lệnh shift

- Tác dụng:
  - Dịch chuyển giá trị hiện tại của các tham số dòng lệnh sang trái một vị trí
- Ví dụ:
  - `$ command` **`par1 par2 par3`**
    - => **$1 = par1, $2 = par2, $3 = par3**
  - Sau khi shift:
    - => **$1 = par2, $2 = par3**, $3 = null
- Shift nhiều vị trí:
  - **shift *n***: Dịch sang trái ***n*** vị trí

Lệnh shift (tt)

- Ví dụ: version 4 của **uppercase**
  - $ **uppercase4 -i vidu.in -o vidu.out**

```
while [ "$1" ]
do
   if [ "$1" = "-i" ] then
    infile=$2
    shift 2
   elif [ "$1" = "-o" ]
   then
    outfile=$2
    shift 2
   else
    echo "Program $0 does not recognize option $1"
    break
   fi
done
tr a-z A-Z $infile $outfile
```

Phát biểu select

- Tạo hệ thống menu dòng lệnh:

```
select menuitem [in list_of_items]
do
  commands
done
```

  - Dấu ngoặc []: tùy chọn của phát biểu select
  - **list_of_items**: biến hoặc danh sách chứa nhiều hơn 1 phần tử (*menu item*)
  - Nếu không chỉ định list_of_item, select sẽ dùng danh sách tham số dòng lệnh (như **for**)

Phát biểu select (tt)

- Ví dụ phát biểu **select**:
  - Hiển thị 3 menu pick1, pick2, pick3:

```
select menuitem in pick1 pick2 pick3
do
  echo "Ban muon chon muc $menuitem?"
  read res
  if [ $res = "y" -o $res = "Y" ]
  then
       break
  fi
done
```

  - Lệnh **read**: Đọc dữ liệu user nhập từ bàn phím

Định nghĩa thủ tục

- Khai báo hàm:

```
function_name () {
  commands
}
```

- Gọi hàm:

```
function_name [param1 param2 …]
```

  - Hàm không hạn chế số tham số
  - Tham số hàm có thể xem như tham số dòng lệnh (truy xuất bằng **$1**, **$2**, dùng **shift**, … như với lệnh)
- Mã trả về:
  - Gọi lệnh **return [*n*]**: ***n*** chứa giá trị trả về (1 byte)

Định nghĩa thủ tục (tt)

- Ví dụ sử dụng hàm:
  - File **tinhgiaithua**: Chương trình tính giai thừa

```
#!/bin/bash
# Ham giaithua
giaithua () {
        if [ $1 -gt 1 ] ; then
                PREV=`expr $1 - 1`
                giaithua $PREV
                RESULT=`expr $RESULT \* $1`
        fi
}
# Chuong trinh chinh
RESULT=1
giaithua $1
echo "Giai thua cua $1 = $RESULT"
```

  - **Lưu ý**: Dấu nhân (*) phải thay bằng ***** để shell không hiểu lầm

Một số quy ước khác

- Mỗi file chương trình shell nên bắt đầu với:

  `#!/bin/bash`

  - => Thông báo đây là Bash shellscript để hệ thống nhận biết khi thực thi. Ngoài ra các trình soạn thảo văn bản như **vi** cũng hỗ trợ tô màu nội dung.
- Dấu chấm phẩy (**;**):
  - Dùng để phân cách nhiều lệnh trên cùng một dòng
- Thoát chương trình shell:
  - Gọi lệnh **`exit n`** với **n** là mã thoát
- Lệnh **continue**, **break**:
  - **`continue`**: bỏ qua lệnh còn lại và sang bước lặp kế
  - **`continue n`**: tiếp tục từ n vòng lặp gần nhất
  - **`break n`**: thoát khỏi n vòng lặp gần lệnh break nhất

Một số quy ước khác (tt)

- Lệnh **export**: Truyền giá trị biến xuống các shell con

```
$ DISPLAY=10.1.1.12:0.0; export DISPLAY
```

- Lệnh **sed** (stream editor): Hiệu chỉnh dữ liệu đầu vào, ví dụ:

  sed '1,2d' vb.txt => xóa dòng 1 và dòng 2 từ vb.txt

  sed '/bye/d' vb.txt => xóa dòng có chữ bye

  sed 's/bad/good' vb.txt => thay từ bad thành good

  sed -n '/hello/p' vb.txt => in những dòng có từ hello

  sed 's/^M//g' vb.dos > vb.unix => đổi văn bản DOS -> UNIX

Một số quy ước khác (tt)

- Cấu trúc **&&** (và):

  cmd1 **&&** ***cmd2*** tương đương với

```
if cmd1 ; then
  cmd2 ; fi
```

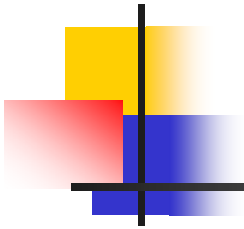
- Cấu trúc **||** (hay):

  cmd1 **||** ***cmd2*** tương đương với

```
if cmd1 ; then
  :
else
  cmd2 ; fi
```

Hệ điều hành Linux

Phần 5: Lập trình Linux



Ngôn ngữ C

- Lịch sử:
  - Phát triển bởi Dennis Ritchie tại Bell Laboratories khi xây dựng hệ thống UNIX
  - Các phiên bản UNIX đầu tiên được viết bằng Assembly và ngôn ngữ B.
  - C ra đời nhằm khắc phục các yếu điểm của B và trở thành ngôn ngữ thông dụng nhất.
- Ưu điểm:
  - Chuẩn hóa trên nhiều nền tảng, hệ điều hành
  - Chương trình thực thi nhanh

GNU C Compiler

- GNU C Compiler (GCC):
  - Là trình biên dịch C thông dụng trên Linux
  - Tương thích với chuẩn ANSI C
- Cú pháp tổng quát:

  `gcc [options] [filenames]`
  - Các `options` sẽ được áp dụng cho từng file trong `filenames`.

GNU C Compiler (tt)

- Lưu ý về options:
  - > 100 options trong GCC
  - Có nhiều options chứa từ 2 kí tự trở lên
  - => Không thể nhóm nhiều options sau một dấu –
  - => Mỗi option phải đi với một dấu – riêng
  - Ví dụ: 2 lệnh sau có ý nghĩa khác nhau:

```
gcc -p -g test.c
gcc -pg test.c
```

GNU C Compiler (tt)

- Biên dịch chương trình:

```
gcc test.c
```

  - => File thực thi tên **a.out**

```
gcc -o thunghiem test.c
```

  - => File thực thi tên **thunghiem**

```
gcc -c test.c
```

  - => File mã đối tượng tên **test.o**. Sau đó liên kết các file object thành file thực thi.
  - => Dùng option này khi liên kết nhiều file

GNU C Compiler (tt)

- Option biên dịch tối ưu:
  - Mặc định: Chương trình dịch nhanh và dễ debug.
  - `-O` và `-O2`: Chương trình nhỏ hơn và chạy nhanh hơn.
- Option debug và profile:
  - `-g`: Tạo thông tin debug để trình GNU debugger (**gdb**) sử dụng.
  - `-pg`: Tạo profile để trình **gprof** hiển thị thông tin timing khi chạy chương trình.

GNU Debugger (gdb)

- Cú pháp chung:

  `gdb [filename]`

  - => Bắt đầu debug một file chương trình
  - Trước đó, file chương trình cần được biên dịch với tùy chọn -g.
- Các lệnh cơ bản trong **gdb**:
  - file: nạp file chương trình cần debug
  - kill: dừng debug chương trình
  - list: xem các phần source code

GNU Debugger (gdb)

- Các lệnh cơ bản trong **gdb**: (tt)
  - break: đặt điểm dừng (breakpoint) trong code
  - run: thực thi chương trình cần debug
  - next: chạy 1 dòng code, không chạy vào trong các hàm.
  - step: chạy 1 dòng code, chạy vào trong các hàm trên dòng đó.
  - watch: hiển thị giá trị của biến khi bị thay đổi
  - quit: thoát khỏi gdb

Ví dụ chương trình C

- Soạn thảo file **test.c**:

```
$ cat > test.c
#include <stdio.h>
int main() {
  int n = 10;
  n += 2;
  printf("n = %i\n", n);
  return 0;
}
```

Ctrl-Z Enter

Ví dụ chương trình C (tt)

- Dịch chương trình **test.c**:

```
$ gcc -g -o test test.c
```

  - => file chương trình test nằm trong cùng một thư mục với test.c
  - Thêm tùy chọn -g để có thể debug chương trình

- Thực thi chương trình **test**:

```
$ ./test
n = 12
```

Ví dụ chương trình C (tt)

- Debug chương trình **test**:

```
$ gdb test
GNU gdb Red Hat Linux (…)
Copyright 2003 Free Software Foundation,
  Inc.
(gdb) list
1       #include <stdio.h>
2       int main() {
3             int n = 10;
4             n += 2;
5             printf("n = %i\n", n);
6             return 0;
7       }
```

Ví dụ chương trình C (tt)

- Debug chương trình **test**: (tt)

```
(gdb) break 4
Breakpoint 1 at 0x804833f: test.c, line 4
(gdb) run
Starting program: /home/tuan/test
Breakpoint 1, main () at test.c:4
4               n += 2;
(gdb) watch n
Hardware watchpoint 2: n
```

Ví dụ chương trình C (tt)

- Debug chương trình **test**: (tt)

```
(gdb) next
Hardware watchpoint 2: n
Old value = 10
New value = 12
main () at test.c:5
5                printf("n = %i\n", n);
(gdb) next
n = 12
7                return 0;
(gdb) kill
(gdb) quit
```

Một số tiện ích lập trình C

- xxgdb:
  - Phiên bản giao diện đồ họa (X Window) của gdb
- indent:
  - Định dạng mã nguồn theo quy tắt thống nhất
- gprof:
  - Cho biết mỗi hàm được gọi bao nhiêu lần và phần trăm thời gian thực thi của hàm.
  - Cần biên dịch chương trình với tùy chọn -pg
- p2c:
  - Chuyển mã nguồn Pascal thành mã nguồn C

Source Code Control

- Mục đích:
  - Quản lý các phiên bản source code của phần mềm.
  - Giúp tìm kiếm, cập nhật, theo dõi các phiên bản khác nhau của một file source dễ dàng.
- Tiện ích make:
  - Tự động biên dịch, liên kết các file source trong một project phần mềm.
  - Thể hiện mối quan hệ phụ thuộc giữa các file source.

Source Code Control (tt)

- Tiện ích make: (tt)
  - Chỉ cập nhật các file có thay đổi sau lần biên dịch sau cùng.
  - Sử dụng một file mô tả (*Makefile*) chứa các luật cần thực hiện khi biên dịch phần mềm.
  - Mỗi luật sẽ sinh ra các lệnh cần thiết cho quá trình biên dịch. Các lệnh đó được thực thi bởi shell.

Source Code Control (tt)

- Ví dụ một Makefile:
  - Project someonehappy bao gồm các file sau:
    - 2 file source: **`main.c`**, **`dothis.c`**
    - 3 file header: **`yes.h, no.h, maybe.h`**
    - 1 file thư viện: **`/usr/happy/lib/likeatree.a`**
    - 1 file assembly: **`itquick.s`**
  - => file chương trình **`someonehappy`**

Source Code Control (tt)

- Ví dụ một Makefile: (tt)
  - Mội dung Makefile:

```
someonehappy: main.o dothis.o
  /usr/happy/lib/likeatree.a
  gcc –o someonehappy main.o dothis.o
  itquick.o /usr/happy/lib/likeatree.a
main.o: main.c
  gcc -c main.c
dothis.o: dothis.c
  gcc -c dothis.c
itquick.o: itquick.s
  as -o itquick.o itquick.s
```

Source Code Control (tt)

- Ví dụ một Makefile: (tt)
  - Mội dung Makefile: (tt)

```
fresh:
  rm *.o
maybe.h: yes.h no.h
  cp yes.h no.h /user/sue/
```

  - Thực thi make:

```
$ make someonehappy
```

hoặc

```
$ make
```

Source Code Control (tt)

- Định dạng của Makefile
  - Bao gồm nhiều mục, mỗi mục có dạng:

```
<target>: [dependents]
    <command list>
```

    - => Nếu file <target> cũ hơn so với các file [dependents] thì make sẽ thực thi <command list>.
    - [dependents] là danh sách các file
    - Các lệnh trong <command list> cách nhau bởi dấu chấm phẩy (;) và không có ký tự xuống dòng
    - Trước <command list> phải là dấu cách TAB

Source Code Control (tt)

- Forcing recompiles:
  - Không muốn **`make`** biên dịch lại mỗi khi copy file từ nơi này qua nơi khác
  - => Sử dụng tiện ích **`touch`** hay gọi **`make`** với tùy chọn **`-t`**
- Kiểm tra Makefile:
  - Gọi **`make`** với tùy chọn **`-n`**
  - => **`make`** sẽ in ra các thông báo nhưng không biên dịch thật chương trình

Source Code Control (tt)

- Macro:
  - Tương tự như các biến trong lập trình shell
  - Ví dụ:

```
LIBFILES=/usr/happy/lib/likeatree.a
objects = main.o dothis.o
CC = /usr/bin/cc
1version="This is one version of
  someonehappy"
OPTIONS =
```

  - Makefile cũng coi các biến môi trường shell như macro.

Source Code Control (tt)

- Macro: (tt)
  - Các macro quan trọng trong Makefile:

    CC: Trình C Compiler

    CFLAGS: Các cờ biên dịch cho **CC**
  - Macro có thể tham khảo các macro khác:

```
LIB_DIR = /usr/happy/lib
LIB_FILES = ${LIB_DIR}/likeatree.a
OBJS = main.o dothis.o itquick.o
sohappy: ${OBJS} ${LIB_FILES}
  ${CC} -o sohappy ${OBJS} ${LIB_FILES}
```

Source Code Control (tt)

- Luật hậu tố:
  - Áp dụng chung một mục của Makefile cho các target hay dependent có phần mở rộng cho trước => Không cần tạo cho mỗi file một mục riêng trong Makefile.
  - Một số luật hậu tố mặc định của make:

```
.SUFFIXES: .o .c .s
.c.o:
  $(CC) $(CFLAGS) -c $<
.s.o:
  $(AS) $(ASFLAGS) -o $@ $<
```

Source Code Control (tt)

- Luật hậu tố:
  - Một số luật hậu tố mặc định của make: (tt)

```
.SUFFIXES: .o .c .s
```

    - => Các hậu tố cần áp dụng luật

```
.c.o:
    $(CC) $(CFLAGS) -c $<
```

    - => Dịch các file *.c nếu các file *.o của chúng chưa được cập nhật
    - $< đại diện cho mỗi dependent *.c (tương tự $?)
    - $@ đại diện cho mỗi target *.o