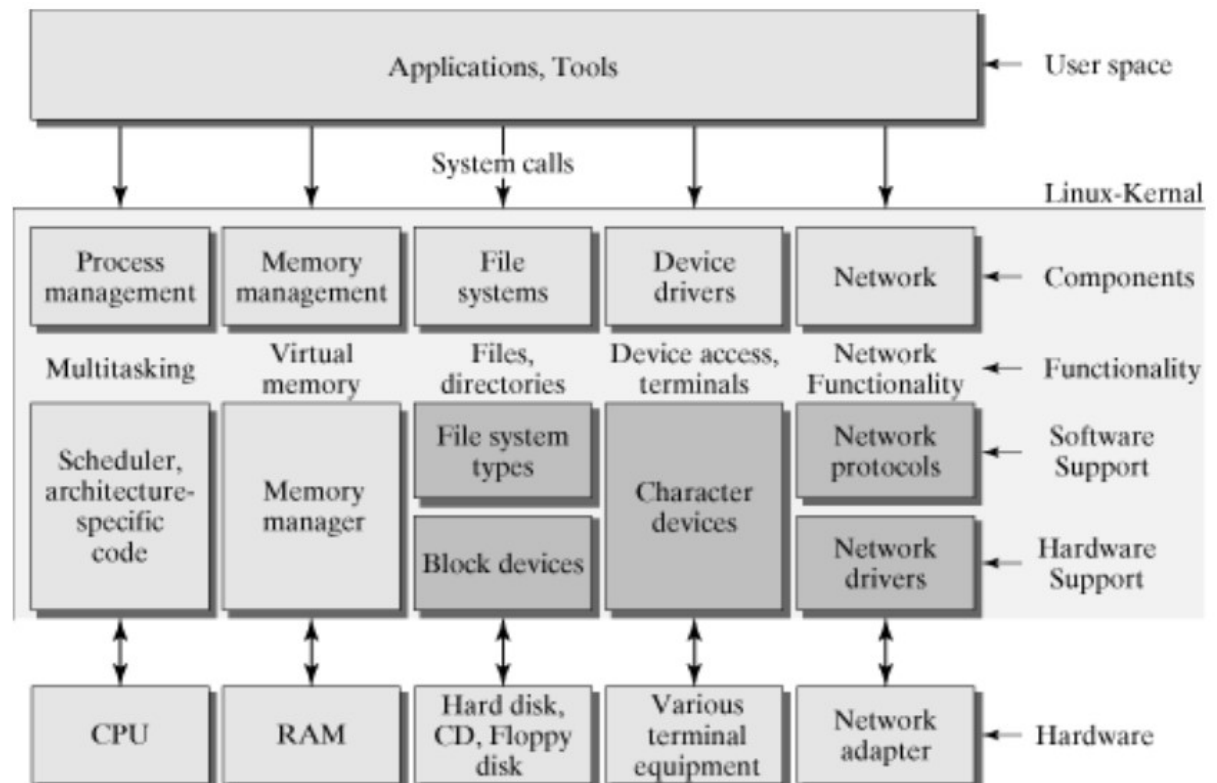


LINUX Kernel



# Cấu trúc chung của Linux

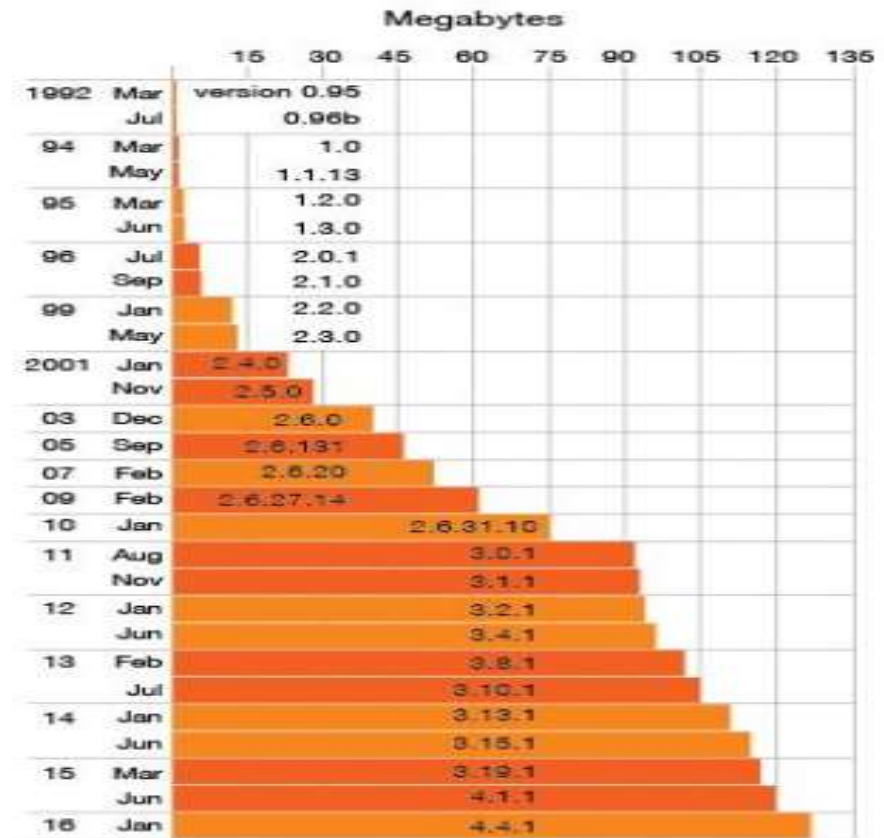


# Một số khái niệm cơ bản trong Linux



# Nhân Linux: Linux kernel

- Thành phần chính điều phối mọi hoạt động của hệ thống: tiến trình, CPU, bộ nhớ, file, thiết bị, mạng,...
- Android OS thực chất là một phiên bản linux tối ưu cho thiết bị di động



# Linux distro

- Nghĩa tiếng Việt là “bản phân phối linux”
- Nhân linux chỉ giúp quản trị tài nguyên máy tính, chưa phải là một hệ thống hoàn chỉnh dành cho người sử dụng

Linux distro = nhân linux + các phần mềm bổ sung

- Do mã nguồn mở, nên mỗi một công ty có thể tùy chọn các phần mềm bổ sung theo mục tiêu riêng của mình. Một phần mềm thường tối ưu cho distro cụ thể nào đó
- Mỗi cách xây dựng hệ thống như vậy gọi là một distro
- Có hàng nghìn linux distro khác nhau

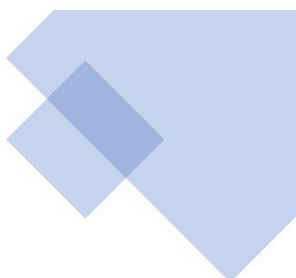

# Linux distro





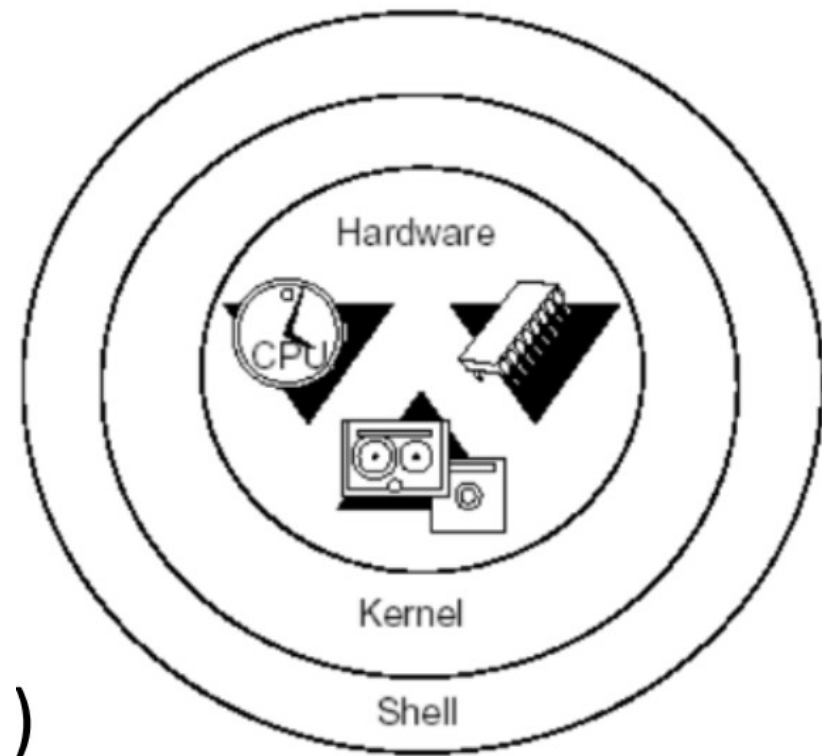
# Linux distro

Linux distro thường gồm:

1. Nhân linux (linux kernel)
  2. Tập hợp các gói phần mềm (software packages)
  3. Chương trình cài đặt (installer)
  4. Các cấu hình của riêng nhà sản xuất (re-configure)
  5. Trình quản lý và cập nhật gói (update/patch)
  6. Tài liệu hướng dẫn sử dụng (user guide)
- 
- 

# Linux shell

- Linux shell là bộ diễn dịch các câu lệnh thành các yêu cầu cho hệ thống
- Trong linux có nhiều shell
  - Bourne shell (bash)
  - Korn shell
  - C shell
- Bash là shell mặc định
- Dùng giao diện text (console)
- Giao diện đồ họa (x-window) thực chất chỉ là ứng dụng chạy trên shell



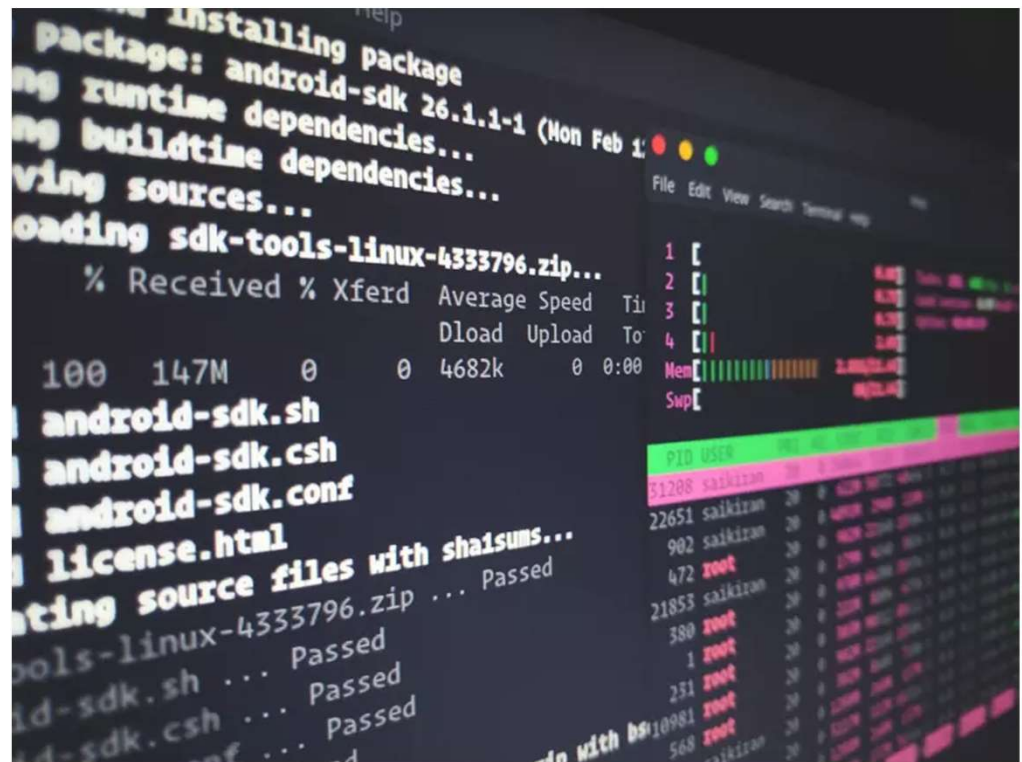


# Một số câu lệnh hệ thống

- Các lệnh linux thường gồm 3 khối  
**<lệnh>** **<lựa chọn>** **<tham số>**

Trong đó:

- <lệnh>**: cố định, phải học và nhớ
- <lựa chọn>** và **<tham số>**: tùy vào từng lệnh



# Một số câu lệnh hệ thống

- **passwd**: đổi mật khẩu người dùng
- **whoami**: xem tên người dùng
- **who am i**: xem chi tiết người dùng
- **last**: xem các phiên làm việc gần đây
- **uname -a**: xem thông tin hệ thống
- **free**: xem bộ nhớ còn trống
- **df**: xem dung lượng lưu trữ còn trống
- **ps -l**: xem thông tin các tiến trình
- **uptime**: xem thời gian hoạt động của máy
- **w**: uptime + who
- **whereis X**: xem ứng dụng X ở đâu
- **date**: xem ngày giờ (ở đồng hồ của máy)
- **sudo X**: chạy ứng dụng X với “quyền root”

# Lệnh liệt kê: `ls`

- Cú pháp:  
`ls [tùy chọn] [thư mục]`
- Một số tùy chọn:
  - `-x` hiển thị trên nhiều cột
  - `-l` hiển thị chi tiết các thông tin của tập tin
  - `-a` hiển thị tất cả các tập tin kể cả tập tin ẩn
- Nếu không chỉ tên thư mục, thì lệnh sẽ liệt kê các file trong thư mục hiện tại  
Lệnh sau làm gì? `ls */*`

# Lệnh liệt kê **ls**

Phân loại file

các quyền truy cập cho chủ

Các quyền truy cập cho nhóm chủ

Các quyền truy cập cho người dùng bên ngoài

Kích thước của file

Thời điểm file được tạo ra

1 /home/user # ls -l

2 total 36

3	drwxr-xr-x	3	root	root	4096	2006-06-29 04:21	.
4	drwxr-xr-x	7	root	root	4096	2006-06-23 02:13	..
5	-rwxr-xr-x	1	root	root	6096	2006-06-22 09:26	functions
6	-rw-r--r--	2	anon	users	651	2006-06-23 05:23	hardlink
7	-rw-r--r--	2	anon	users	651	2006-06-23 05:23	mark.txt
8	drwxr-xr-x	2	root	root	4096	2006-06-22 09:27	mydir
9	brw-rw----	1	root	disk	8, 192	2005-05-24 08:09	sdm
10	-rwsr-sr-x	1	root	root	6096	2006-06-22 09:29	share
11	lrwxrwxrwx	1	root	root	9	2006-06-22 09:28	softlink -> functions
12	-rw-r--r--	1	root	root	0	2006-06-29 04:21	zerobyte.txt

Tổng số liên kết  
đến cùng 1 file

Tên nhóm chủ của tập tin

Tài khoản cá nhân chủ tập tin

Tên tập tin

Kiểm tra thuộc tính của các tập tin bằng lệnh `ls -l`

# Sao chép tập tin / thư mục

- Cú pháp:

`cp [tùy chọn] <nguồn> <đích>`

- Một số tùy chọn:

`-f` ghi đè không cần hỏi (force)

`-i` hỏi trước khi ghi đè (interactive)

`-r` sao chép toàn bộ thư mục kể cả con

- Ví dụ:

`cp -r dir1 dir5`

`cp file1 file5`

# Xóa tập tin và thư mục

- Cú pháp:  
`rm [tùy chọn] <tập tin>`
- Một số tùy chọn:
  - `f` xóa không cần hỏi
  - `i` hỏi trước khi xóa
  - `r` xóa toàn bộ thư mục kể cả con
- Lưu ý: KHÔNG dùng lệnh `rm -rf /`

# Đổi tên hoặc dịch chuyển tập tin

- Cú pháp:  
`mv [tùy chọn] <nguồn> <đích>`
- Một số tùy chọn :
  - `f` ghi đè không cần hỏi (force)
  - `I` hỏi trước khi ghi đè (interactive)
- Ví dụ :  
`mv file5 file6`  
`mv file1 dir5`

# Tạo thư mục

- Cú pháp:

`mkdir [tùy chọn] <thư mục> ...`

- Tùy chọn:

`-p` tạo thư mục cha nếu chưa tồn tại

- Ví dụ:

`mkdir dir1`

`mkdir dir1 dir2`

`mkdir -p dir3/dir4`





# Xóa thư mục rỗng

- Cú pháp:

`rmdir [tùy chọn] <thư mục> ...`

- Tùy chọn :

`-p` xóa tất cả các thư mục tạo nên đường dẫn

- Ví dụ :

`rmdir dir1`

`rmdir dir1 dir2`

`rmdir -p dir3/dir4`

`rmdir dir3/dir4 dir3`

# Lệnh **wc**

- Cho biết thông tin về số dòng, số từ, kích thước (byte) của tập tin
- Cú pháp:  
**wc [tùy chọn] [tập tin 1] ... [tập tin n]**
- Một số tùy chọn:
  - c** kích thước tập tin (byte) gồm cả ký tự CR và EOF
  - m** số lượng ký tự có trong tập tin
  - w** số lượng từ có trong tập tin
  - l** số dòng trong tập tin
  - L** chiều dài của dòng dài nhất

# Lệnh **touch** và **cat**

- Lệnh “**touch filename**”: tạo tập tin rỗng (hoặc xóa nội dung nếu file đã có từ trước)
- Lệnh “**cat**” dùng để hiển thị nội dung tập tin
- Cú pháp:

**cat [tùy chọn] [tập tin 1] ... [tập tin n]**

- Một số tùy chọn :
  - s** xóa các dòng trắng chỉ để lại 1 dòng duy nhất
  - n** đánh số thứ tự các dòng, kể cả dòng trắng
  - b** đánh số thứ tự các dòng, ngoại trừ dòng trắng

# Lệnh **more**

- Xem nội dung của tập tin theo từng trang màn hình
- Cú pháp:

**more [tùy chọn] [tập tin 1] ... [tập tin n]**

- Một số tùy chọn:
  - **n** xác định kích thước của màn hình n dòng
  - + **n** dòng bắt đầu hiển thị
  - **s** xóa bớt các dòng trắng
- Bấm **space** để xem trang tiếp
- Bấm **b** để xem trang trước

# Lệnh **head**

- Xem nội dung đầu tập tin
- Cú pháp:

**head** [tùy chọn] [tập tin 1] ... [tập tin n]

- Một số tùy chọn:
  - n** in ra màn hình n dòng đầu tiên (mặc định lệnh **head** sẽ hiển thị 10 dòng đầu)
  - q** không hiển thị ra màn hình phần đầu đề chứa tên tập tin trong trường hợp mở nhiều tập tin cùng lúc

# Lệnh **tail**

- Xem nội dung cuối tập tin
- Cú pháp:

**tail** [tùy chọn] [tập tin 1] ... [tập tin n]

- Một số tùy chọn:
  - n** in ra màn hình n dòng cuối cùng (mặc định lệnh tail sẽ hiển thị 10 dòng cuối)
  - q** không hiển thị ra màn hình phần đầu đề chứa tên tập tin trong trường hợp mở nhiều tập tin cùng lúc
  - f** cập nhật liên tục (mỗi khi nội dung file thay đổi)

# Lệnh **find**

- Tìm kiếm tập tin
- Cú pháp:

```
find [path ... ] [expression]
```

- Một số tùy chọn:

- name "pattern"** tìm các tập tin có tên chứa chuỗi **pattern**
- group name** tìm các tập tin thuộc nhóm **name**
- user name** tìm các tập tin tạo bởi user có tên **name**
- size [+/-]n[bck]** tìm các tập tin kích thước lớn hơn/nhỏ hơn **n**  
**block** (512 bytes/block). Kích thước là **block** nếu ký tự theo sau là b, c là byte, **k** là kilobytes.
- type filetype** tìm các tập tin có kiểu là **filetype**

# Lệnh **grep**

- Tìm kiếm một chuỗi nào đó trong nội dung tập tin
- Cú pháp:

```
grep [options] pattern [file] ...
```

- Một số tùy chọn:
  - I** không phân biệt hoa thường
  - n** kèm theo số thứ tự dòng khi xuất kết quả
  - r** tìm lặp lại trong thư mục con
  - v** tìm nghịch đảo
  - a** xử lý tập tin nhị phân như là một tập tin văn bản



# Regular expression và ống lệnh

- Một số regular expression:

`"^xxxx"` begin of line: Tìm các dòng bắt đầu bởi xâu `xxxx`

`.` ký tự bất kỳ

`"xxxx$"` end of line: Tìm các dòng kết thúc bởi xâu `xxxx`.

- Ví dụ:

- Liệt kê tất cả các file trong `/etc` bắt đầu bằng `b, k, n`

```
ls /etc | grep "^[bkn]"
```

- Liệt kê tất cả các file trong `/etc` có ký tự kế cuối là `a`

```
ls /etc | grep "a$"
```

# Lệnh **cmp**

- So sánh hai tập tin có kiểu bất kỳ
- Cú pháp:

```
cmp [-l] file1 file2
```

- Trong đó **-l** cho phép xuất ra danh sách các vị trí khác nhau giữa hai tập tin
- Ví dụ:

```
cmp myfile m1
```

# Lệnh **diff**

- Tìm sự khác nhau giữa hai tập tin
- Cú pháp:

**diff [tùy chọn] from-file to-file**

- Một số tùy chọn:
  - I** so sánh không phân biệt hoa thường
  - s** hiển thị thông báo nếu hai tập tin giống nhau
  - w** bỏ qua khoảng trắng giữa các từ
  - r** so sánh tất cả các tập tin trong các thư mục con

# PHÂN QUYỀN TRONG LINUX

# User

- Linux phân chia người dùng thành 2 loại chính:
  - Người quản trị (root hay superuser): có thể thực hiện mọi thứ với máy tính
  - Người dùng thường (user hay normal user): bị hạn chế một số chức năng (ví dụ như thay đổi thiết lập khởi động, cài đặt phần mềm, cập nhật hệ thống,...)
  - Người dùng được hệ thống cấp tên (username), mật khẩu (password) và thư mục con dành riêng trong /home để lưu trữ dữ liệu cá nhân (download, ảnh, tài liệu, desktop,...)

# Một số lệnh về phân quyền

- Tạo người dùng mới: `useradd username`
- Xóa người dùng cũ: `userdel username`
- Thiết lập mật khẩu: `passwd username`
- Tạo nhóm mới: `groupadd groupname`
- Xóa nhóm mới: `groupdel groupname`
- Thêm người dùng vào nhóm:  
`gpasswd -a username groupname`
- Thay đổi nhóm chính của một người dùng:  
`usermod -g groupname username`

# Nhắc lại: lệnh ls -l

**Phân loại file**  
**các quyền truy cập cho chủ**  
**Các quyền truy cập cho nhóm chủ**  
**Các quyền truy cập cho người dùng bên ngoài**  
**Kích thước của file**  
**Thời điểm file được tạo ra**

```
1 /home/user # ls -l
2 total 36
3 drwxr-xr-x 3 root root 4096 2006-06-29 04:21 .
4 drwxr-xr-x 7 root root 4096 2006-06-23 02:13 ..
5 -rwxr-xr-x 1 root root 6096 2006-06-22 09:26 functions
6 -rw-r--r-- 2 anon users 651 2006-06-23 05:23 hardlink
7 -rw-r--r-- 2 anon users 651 2006-06-23 05:23 mark.txt
8 drwxr-xr-x 2 root root 4096 2006-06-22 09:27 mydir
9 brw-rw-r-- 1 root disk 8, 192 2005-05-24 08:09 sdm
10 -rwsr-sr-x 1 root root 6096 2006-06-22 09:29 share
11 lrwxrwxrwx 1 root root 9 2006-06-22 09:28 softlink -> functions
12 -rw-r--r-- 1 root root 0 2006-06-29 04:21 zerobyte.txt
```

**Tổng số liên kết đến cùng 1 file**  
**Tên nhóm chủ của tập tin**  
**Tài khoản cá nhân chủ tập tin**  
**Tên tập tin**

Kiểm tra thuộc tính của các tập tin bằng lệnh `ls -l`

Kiểu tập tin	Quyền tập tin	Số liên kết	Chủ nhân	Tên nhóm	Kích thước (byte)	Thời điểm sửa đổi sau cùng	Tên tập tin
	-rw-r--r--	1	chris	weather	207	Feb 20 11:55	mydata

# Ý nghĩa

- Ý nghĩa phân quyền
  - Quyền đọc (read) – r mã quyền là 4
  - Quyền ghi (write) – w mã quyền là 2
  - Quyền chạy (execute) – x mã quyền là 1
- Chú ý: với thư mục, quyền chạy là quyền vào xem nội dung



# Một số lệnh khác

- Phân quyền cho group:

```
chgrp groupname file
```

- Phân quyền cho user:

```
chown username file
```

- Phân quyền tổng quát:

```
chmod code file
```

- Code ở đây ứng với quyền của user/group/other

```
chmod 755 abc
```

```
chmod -R 777
```

# LUỒNG VÀO RA DỮ LIỆU

# Các luồng vào ra dữ liệu chuẩn

- Khái niệm “luồng”: dãy dữ liệu được xử lý tuần tự, Tương tự như khái niệm stream trong lập trình C++
  - “luồng vào”: dãy dữ liệu được gửi vào chương trình
  - “luồng ra”: dữ liệu kết quả, được chương trình gửi trả lại từng thành phần cho chương trình gọi
- Khi thực thi một chương trình trên linux, hệ thống mặc định tạo 3 luồng cho chương trình đó
  - Luồng 0 (luồng vào chuẩn): thường là bàn phím
  - Luồng 1 (luồng ra chuẩn): thường là màn hình console
  - Luồng 2 (luồng lỗi chuẩn): thường là màn hình console

# Các luồng vào ra dữ liệu chuẩn

- Chương trình luôn hoạt động theo nguyên tắc:
  - Đọc dữ liệu đầu vào từ luồng 0
  - Nếu có kết quả thì ghi ra luồng 1
  - Nếu có báo lỗi thì ghi ra luồng 2
- Chính vì hoạt động mặc định trên, thông thường ta luôn nhập liệu từ bàn phím vào chương trình, và khi hoạt động, chương trình in ra màn hình kết quả hoạt động hoặc báo lỗi
- Người dùng có thể thay đổi các luồng vào/ra chuẩn để phục vụ những ý đồ riêng của mình

# Đổi hướng xuất (output redirection)

- Sử dụng ký tự “>” để đổi hướng việc xuất dữ liệu
- Cú pháp:

**lệnh > tập-tin**

- Cách làm việc:
  - Thay vì ghi dữ liệu ra màn hình, kết quả thực hiện câu lệnh sẽ được ghi vào tập-tin chỉ định (ghi đè nội dung đã có)
  - Nếu muốn ghi dữ liệu vào tập-tin chỉ định, nhưng giữ nguyên dữ liệu cũ, ghi thêm vào cuối tập-tin, dùng “>>”

- Ví dụ :

**ls -l /tmp/ > /home/txnam/t1.out**

**ls -l /etc/ >> t1.out**

# Đổi hướng nhập (input redirection)

- Sử dụng “<” hoặc “0<” để đổi hướng việc nhập liệu
- Cú pháp:  
    `lệnh < tập-tin`  
    `lệnh 0< tập-tin`
- Cách làm việc: thay vì nhận dữ liệu từ bàn phím, câu lệnh sẽ nhận dữ liệu từ tập-tin chỉ định
- Ví dụ :  
    `cat < /etc/passwd`  
    `more 0< /etc/passwd`

# Đổi hướng lỗi (error redirection)

- Sử dụng “2>” để đổi hướng thông báo lỗi
- Cú pháp: `lệnh 2> tập-tin`
- Cách làm việc: Những thông báo lỗi sẽ ghi vào tập-tin thay vì màn hình. Nội dung cũ trong tập-tin sẽ bị xóa
- Trường hợp muốn giữ lại nội dung ban đầu của tập-tin và chèn thông tin lỗi vào cuối tập tin, dùng “2>>”
- Ví dụ:

```
ls -l /temp/ > t1.out 2> log.err
```

```
ls -l /etc/ >> t1.out 2>> log.err
```

# QUẢN LÝ TIẾN TRÌNH



# Các khái niệm

- Chương trình (*program*) là một file thực thi trong hệ thống, ví dụ: **/sbin/shutdown**
- Tiến trình (*process*) là một chương trình đã được nạp vào bộ nhớ và được cấp CPU để hoạt động
  - Có thể mở nhiều cửa sổ terminal để thử nghiệm các lệnh, mỗi cửa sổ là một tiến trình
  - Tiến trình đôi khi được gọi là tác vụ (*task*)
- Khi khởi chạy, mỗi tiến trình được cấp một chỉ số PID (*process id*) duy nhất. Hệ thống dùng PID để quản lý tiến trình

# Các khái niệm

- Tiến trình cũng có phân quyền sở hữu và truy cập (như với tập tin)
- Linux cho phép nhiều tiến trình chạy cùng lúc
  - Nhân linux có một module riêng lập lịch phân phối CPU cho từng tiến trình để đảm bảo các tiến trình đều được hoạt động hợp lý
  - Mỗi tiến trình có một chỉ số ưu tiên (**priority**) tương ứng
  - Chỉ số ưu tiên càng thấp thì hệ thống càng ưu tiên phân phối nhiều thời gian sử dụng CPU cho tiến trình đó
  - Có thể chỉnh lại chỉ số ưu tiên này bằng lệnh **nice** hoặc **renice**

# Các loại tiến trình

- Một tiến trình có thể yêu cầu hệ thống khởi chạy một tiến trình khác (ví dụ: trình duyệt có thể tạo một process riêng khi người dùng mở một link)
  - Khi đó tiến trình được tạo ra gọi là **child process**
  - Tiến trình ban đầu được gọi là **parent process**
- Một tiến trình con đang chạy nhưng tiến trình cha của nó đã kết thúc thì được gọi là **orphan process**
- Một tiến trình đã hoàn tất nhưng vì một lý do gì đó vẫn được giữ trong bộ nhớ thì được gọi là **zombie process** hoặc **defunct process**

# Các loại tiến trình

- Các tiến trình nhận tương tác từ người dùng thì hoạt động ở chế độ mặt trước (**foreground**)
- Các tiến trình không nhận tương tác thì hoạt động ở chế độ nền (**background**)
- Các tiến trình thường chuyển qua chuyển lại giữa hai trạng thái này trong quá trình hoạt động, việc chuyển trạng thái có thể thực hiện do người dùng, do lệnh từ shell hoặc do lập trình
- Tiến trình ở chế độ mặt trước thường nhận được nhiều CPU hơn một chút so với chế độ nền

# Các loại tiến trình

- Hệ thống linux có một số các tiến trình đặc biệt gọi là các **daemon process**
  - Thường cung cấp các chức năng quan trọng của hệ thống, đặc biệt là các dịch vụ mạng
  - Thường thuộc về quyền root
  - Thường không gắn với shell cụ thể nào, không truy xuất vào/ra bàn phím, màn hình
  - Khi sử dụng câu lệnh liệt kê tiến trình sẽ thấy kí hiệu **?** ở trường TTY
  - Đa số **daemon process** không chiếm CPU, chúng chỉ hoạt động khi có yêu cầu

# Liệt kê các tiến trình

- Cú pháp: **ps [options]**
- Một số tùy chọn:
  - **a** tất cả proc của các user khác
  - **x** các proc không gắn với terminal (daemon)
  - **u** user-format
  - **l** long-format
  - **w** wide output
  - **-U user** xem proc của một user cụ thể
- Ví dụ:

```
ps aux
```

```
ps aux | grep httpd
```

# Trạng thái tiến trình

- Cho ở cột **STAT**
  - **R** Đang thi hành
  - **S** Đang bị đóng
  - **Z** Ngừng thi hành
  - **W** Không đủ bộ nhớ cho tiến trình thi hành
- Ví dụ:

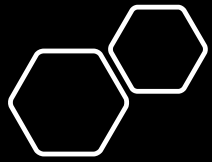
# Thông tin chi tiết tiến trình

- Cú pháp: **top [options]**

Liên tục hiển thị thông tin về các tiến trình

- Một số tùy chọn:
  - **-d delay** Khoảng thời gian trễ giữa hai lần cập nhật
  - **-p [pid]** Chỉ theo dõi tiến trình có mã là pid
- Một số phím lệnh trong sử dụng trong top:
  - **q** Thoát khỏi lệnh top
  - **space** Cập nhật thông tin tiến trình ngay lập tức
  - **k** Ngừng một tiến trình
  - **f** Lựa chọn thông tin tiến trình





Ví dụ:



# Thông tin chi tiết tiến trình (tiếp)

- Đây là công cụ mà quản trị hệ thống linux nào cũng cần biết và sử dụng thành thạo
- Cung cấp thông tin về tiến trình, có thể thực hiện luôn các thao tác với tiến trình (ví dụ: kết thúc tiến trình, thay đổi mức độ ưu tiên,...)
- Cung cấp các chỉ số quan trọng của hệ thống:
  - Thời gian hiện tại, thời gian từ lần khởi động mới nhất
  - Mức độ tải CPU trung bình trong 1, 5, 15 phút gần đây
  - Mức độ chiếm dụng CPU hiện tại
  - Các chỉ số quan trọng của từng tiến trình

# Ngừng tiến trình

- Cú pháp: **kill [-s signal] pid**  
**kill -l [signal]**
- Mặc dù có nhiều tín hiệu khả dụng khác nhau, hầu như chúng ta chỉ dùng **SIGKILL (9)** và **SIGTERM (15)**.
  - **SIGKILL** được sử dụng để kết thúc chương trình ngay lập tức. Nó không thể bị bỏ qua hoặc chặn lại, và do đó luôn luôn gây kết thúc tiến trình.
  - **SIGTERM** cũng được sử dụng để kết thúc chương trình. Không giống như **SIGKILL**, tín hiệu này có thể bị chặn, xử lý hoặc bỏ qua.

# Ngừng tiến trình (tiếp)

- Cú pháp: **killall [-s signal] name**
- Quyền hủy tiến trình (cả **kill** và **killall**) thuộc về người sở hữu tiến trình hoặc quyền root
- Lệnh **killall** kết thúc mọi tiến trình cùng tên, vì thế cẩn thận khi sử dụng lệnh này
  - Tiến trình xử lý web bị lỗi, nếu hủy bằng **killall** có thể dẫn đến hủy mọi giao dịch web đang thực hiện
- Ví dụ:

```
killal -HUP syslogd
```

```
killall -9 man
```

# Điều khiển tác vụ

- Một tác vụ (job) là một tiến trình đang thực thi
- Một số cách điều khiển tác vụ:
  - **^C** thoát ngang
  - **^Z** chuyển sang chế độ nền
  - **"jobs"** liệt kê các tác vụ đang thực thi
  - **&** thực hiện tác vụ ở chế độ nền
- Tiến trình bị tạm ngừng bởi **ctrl-Z** có thể được tiếp tục bằng lệnh fg hoặc bg
  - **fg %x** tiếp tục tác vụ **x** ở foreground
  - **bg %x** tiếp tục tác vụ **x** ở background

# Điều khiển tác vụ

- Để tiến trình chạy ở chế độ background, chúng ta thêm dấu **&** vào sau lệnh thực hiện chương trình

- Ví dụ :

```
find / -name pro -print > results.txt &
```

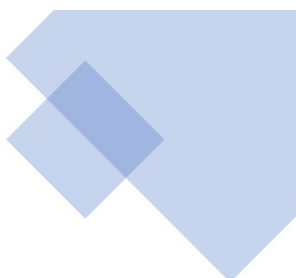

- Để kiểm tra, ta có thể dùng lệnh:

```
ps -aux | grep find
```

- “**jobs**”: để xem các tiến trình đang có ở background



# Theo dõi hệ thống

- **w**: xem các user còn đang login đang làm gì
  - **free**: hiển thị thông tin bộ nhớ.
  - **uptime**: thời gian sống của hệ thống
  - **ps tree**: hiển thị cây tiến trình
  - **pgrep, pkill**: tìm hoặc gửi signal đến tiến trình dựa theo tên và các thuộc tính khác
  - **nice, renice, snice**: thay đổi priority của tiến trình
- 
- 

# LẬP TRÌNH SHELL TRONG LINUX



# Shell script

- **Shell script** là một chuỗi các lệnh được viết trong plain text file. Shell script thì giống như batch file trong MS-DOS nhưng mạnh hơn.

# Shell script

- Shell script có thể nhận input từ user, file hoặc output từ màn hình.
- Tiện lợi để tạo nhóm lệnh riêng.
- Tiết kiệm thời gian.
- Tự động làm một vài công việc thường xuyên.

# Tạo file script và chạy

- Tạo file có đuôi **\*.sh**

**touch myScript.sh**

- Dùng **nano** hoặc bất kỳ trình soạn thảo nào để viết nội dung cho file:
  - Dòng đầu tiên: bắt buộc phải là **#!/bin/bash**
  - Sau đó là các câu lệnh
  - Muốn comment một dòng, dùng dấu **#**

# Chạy file script

- Chạy file bằng cách:
  - Nếu muốn chạy trực tiếp:
    - Cấp quyền chạy cho người dùng  
**chmod xxx myScript.sh**
    - Chạy:  
**./myScript.sh**
  - Nếu không muốn cấp quyền cho file thì dùng câu lệnh:  
**bash myScript.sh**



Ví dụ:

Viết chương trình tạo ra file **ps.txt** và ghi vào file danh sách file của thư mục hiện tại

# Biến trong shell

- **Biến hệ thống :**
  - Tạo ra và quản lý bởi Linux.
  - Tên biến là CHỮ HOA
- **Biến do người dùng định nghĩa:**
  - Tạo ra và quản lý bởi người dùng
  - Tên biến là chữ thường

## Ví dụ

- `echo $BASH_VERSION`
- `echo $BASH`
- `echo $HOME`
- `echo $PATH`

<lệnh **echo** để in giá trị biến ra màn hình>

# Biến người dùng

- Tên biến bắt đầu bởi ký tự.
- Tên biến có phân biệt chữ hoa, thường
- Phép gán:

**Tên biến=Giá trị**

- Không có dấu cách 2 bên toán tử = khi gán giá trị cho biến



# Lệnh **echo**

- Để in ra giá trị biến

```
echo "Hello"
echo $a
```
- In một số ký tự đặc biệt trong tham số với tùy chọn -e:

```
\a alert (bell)
\b backspace
\c suppress trailing new      line
\n new line
\r carriage return
\t horizontal tab
\\ backslash
```

# Các phép toán số học

- Cú pháp:

**expr toán\_hạng\_1 toán\_tử toán\_hạng\_2**

- Ví dụ:

# phép cộng

**\$expr 1 + 2**

# phép trừ

**\$expr 5 - 1**

# phép chia

**\$expr 8 / 3** # output =2 phép chia chỉ lấy phần nguyên

**\$expr 8 % 5** # output =3 phép chia lấy phần dư

**\$expr 10 \\* 2** # output = 20 phép nhân

# Các phép toán số học

- Chú ý: Phải có dấu cách trước và sau toán tử.

`$expr 1+ 2`

# Cấu trúc điều khiển trong shell script

- Cú pháp rẽ nhánh If

```
if [điều_kiện]  
then  
    câu lệnh 1  
    ...  
fi
```

# Lệnh `if`

```
if điều_kiện then  
    câu_lệnh_1  
    ...  
else  
    câu_lệnh_2  
fi
```

# Lệnh `for`

```
for { tên biến } in { danh sách }  
do  
    # Khởi lệnh  
    # Thực hiện từng mục trong danh sách cho đến cho đến hết  
    # (Và lặp lại tất cả các lệnh nằm trong "do" và "done")  
done  
  
#hoặc sử dụng for  
for (( expr1; expr2; expr3 ))  
do  
    # Lặp cho đến khi biểu thức expr2 trả về giá trị TRUE  
done
```

# Lệnh **while**

Cú pháp:

```
while <điều kiện>  
do  
    # câu lệnh 1  
    # câu lệnh 2  
done
```

# Các phép so sánh số học

<b>-eq</b>	is equal to
<b>-ne</b>	is not equal to
<b>-lt</b>	is less than
<b>-le</b>	is less than or equal to
<b>-gt</b>	is greater than
<b>-ge</b>	is greater than or equal to



# So sánh xâu

<code>string1 = string2</code>	string1 is equal to string2
<code>string1 != string2</code>	string1 is NOT equal to string2
<code>string1</code>	string1 is NOT NULL or not defined
<code>-n string1</code>	string1 is NOT NULL and does exist
<code>-z string1</code>	string1 is NULL and does exist

# Tham số khi gọi

- Để truyền tham số khi gọi file script:

`./myScr.sh 100`

- Trong file `*.sh`, người ta sử dụng `$1`, `$2`, `..$n` để truy cập giá trị các tham số truyền vào.
- Ví dụ:

```
#!/bin/sh
#Check whether they are equal
if [ $1 = $2 ]
then
    echo "$1 and $2 are equal"
else
    echo "$1 and $2 are not equal"
fi
```

```
./myScr.sh 10 20
```

# File, thư mục

**-s file**

Non empty file

**-f file**

Is File exist or normal file and not a directory

**-d dir**

Is Directory exist and not a file

**-w file**

Is writeable file

**-r file**

Is read-only file

**-x file**

Is file is executable

# Giao tiếp với người dùng

```
echo "nhập tên file"  
read n
```

# Bài tập

- Viết Chương trình tính tổng các số chẵn từ 1 đến  $n$ , với  $n$  truyền vào như một tham số
- Viết Chương trình tính tổng các số lẻ từ 1 đến  $n$ , với  $n$  truyền vào như một tham số
- Viết Chương trình tính tổng các số từ 1 đến  $n$ , với  $n$  truyền vào như một tham số