

## Chương 05: QUẢN TRỊ HỆ THỐNG

- 5.1. KHỞI NẠP HỆ THỐNG, TIẾN TRÌNH, CÁC LOẠI TIẾN TRÌNH
- 5.2. QUẢN LÝ TIẾN TRÌNH-CÁC TẬP TIN VÀ LỆNH CƠ BẢN

### 5.1. Khởi nạp hệ thống, Tiến trình, các loại tiến trình.

#### 5.1.1. Chương trình nạp khởi động

- Trước khi Red Hat Linux có thể chạy được trên một máy tính, nó phải được khởi động thông qua chương trình nạp khởi động đặc biệt (boot loader).
- Boot loader có nhiệm vụ nạp mã nhân của hệ điều hành Linux cùng các tập tin được yêu cầu của nó vào bộ nhớ.
- Boot loader thường nằm trên ổ đĩa cứng chính (primary) của máy tính.

- Red Hat Linux có thể cài đặt một trong hai chương trình nạp khởi động là LILO (Linux LOader) hoặc GRUB (Grand Unified Bootloader)
- Quá trình LILO hay GRUB nạp bản thân nó vào trong bộ nhớ theo các giai đoạn sau:
  - **Giai đoạn 1:** Chương trình nạp khởi động chính được đọc vào trong bộ nhớ bởi BIOS từ MBR (Master Boot record). Nhiệm vụ duy nhất của nó là gọi nạp chương trình nạp khởi động thứ hai (của Linux).

- **Giai đoạn trung gian (đối với GRUB):**
  - Chương trình nạp khởi động được đọc vào bộ nhớ chỉ khi cần thiết.
  - Trong trường hợp partition **/boot** nằm trên 1025 cylinder của ổ đĩa cứng hoặc khi đĩa cứng sử dụng ở chế độ LBA (Logical Block Addressing) thì cần thực hiện bước trung gian này để có thể gọi được chương trình nạp khởi động thứ hai.

### – Giai đoạn 2:

- Bộ nạp giai đoạn này nằm trên phân vùng (partition) **/boot** và sẽ đọc thông số cấu hình ở **/boot/grub/grub.conf**.
- Chương trình nạp khởi động thứ hai hiển thị màn hình khởi nạp của Red Hat Linux, cho phép người sử dụng chọn hệ điều hành khởi động hay gửi đối số đến nhân hệ điều hành cũng như xem các thông số hệ thống (đối với GRUB).
- Chương trình nạp khởi động thứ hai đọc hệ điều hành hay nhân hệ thống và **initrd** vào bộ nhớ đồng thời chuyển điều khiển tới hệ điều hành được nạp

- Phương pháp được sử dụng để khởi động Red Hat Linux được gọi là phương pháp nạp trực tiếp bởi vì chương trình nạp khởi động hệ điều hành một cách trực tiếp, không có bước trung gian giữa chương trình nạp khởi động và mã nhân của hệ điều hành.
- Một số hệ điều hành khác ví dụ như Window có quá trình khởi động theo phương pháp nạp dây chuyền. Trong phương pháp này, MBR chỉ đơn giản là trỏ đến sector đầu tiên của partition có chứa hệ điều hành. Tại vị trí này, nó tìm các tập tin cần thiết để khởi động thực sự hệ điều hành đó.
- **LILO/GRUB** hỗ trợ cả hai phương pháp nạp trực tiếp và nạp dây chuyền, để cho phép nó khởi động bất kỳ hệ điều hành nào.

## LILO

- Linux cung cấp một chương trình quản lý khởi động là **LILLO** dùng để khởi động hệ điều hành và cho phép người dùng lựa chọn hệ điều hành muốn sử dụng, trong trường hợp máy tính có nhiều hệ điều hành. **LILLO** có một giới hạn là chỉ có thể quản lý được những phần chia khởi động nằm trong phạm vi 1024 cylinder đầu tiên của ổ đĩa cứng.

- **Thiết lập cấu hình LILLO**

- **LILLO** đọc thông tin chứa trong tập tin cấu hình **/etc/lilo.conf** để biết xem hệ thống máy có những hệ điều hành nào, và các thông tin khởi động đang nằm ở đâu.
- **LILLO** được lập cấu hình để khởi động một đoạn thông tin trong tập tin **/etc/lilo.conf** cho từng hệ điều hành.

```

lilo.conf.anaconda [----] 0 L:[ 1+13 14/ 14] *(225
prompt
timeout=50
default=linux
boot=/dev/sda1
map=/boot/map
install=/boot/boot.b
linear

image=/boot/vmlinuz-2.4.22-1.2115.nptl
    label=linux
    initrd=/boot/initrd-2.4.22-1.2115.nptl.img
    read-only
    append="rhgb root=LABEL=/"
-

```

Bài giảng môn học Nhập Môn Hệ Điều Hành

9

- Khoa Công nghệ Thông tin Trường Đại học Nông lâm TP. Hồ Chí Minh
- **prompt**: Xuất hiện thông báo lựa chọn hệ điều hành.
  - **timeout= dsec (1/10 giây)**. Thời gian LILO đợi người dùng lựa chọn hệ điều hành trước khi lựa chọn hệ điều hành mặc định.
  - **default= label**. Khởi động mặc định hệ điều hành có nhãn label được khai báo trong cấu hình LILO.
  - **boot=bootdev**. Xác định ổ đĩa khởi động (MBR trên đó)
- Bài giảng môn học Nhập Môn Hệ Điều Hành

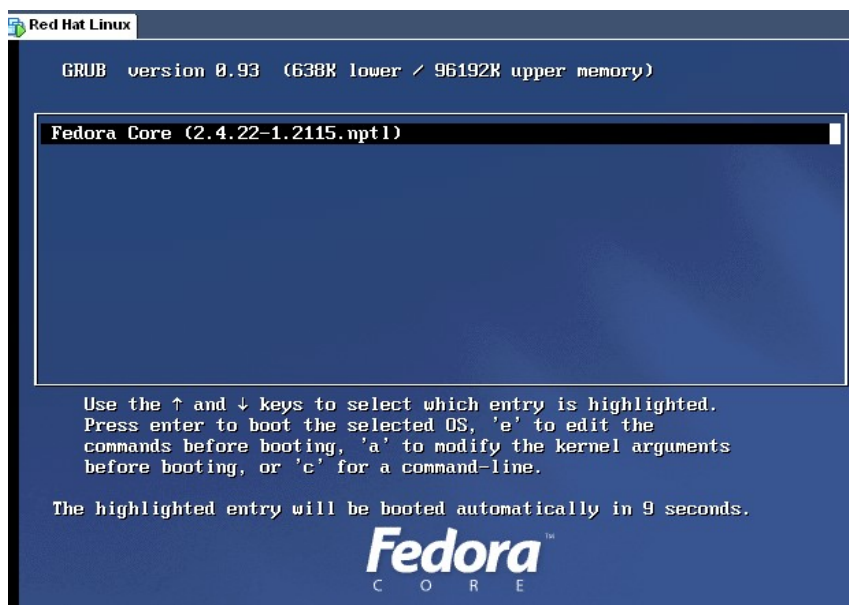
10

- **map=/boot/map**. Kiểm tra tập tin map trong thư mục boot.
- **install=bootsec**. Tập tin được sử dụng như một boot sector mới.
- **LBA32** cho biết cấu hình của đĩa cứng. Nếu có dòng này nghĩa là đĩa cứng hỗ trợ LBA32, thông thường dòng này có giá trị linear.
- **image=path**. Đường dẫn đến nhân hệ điều hành Linux.

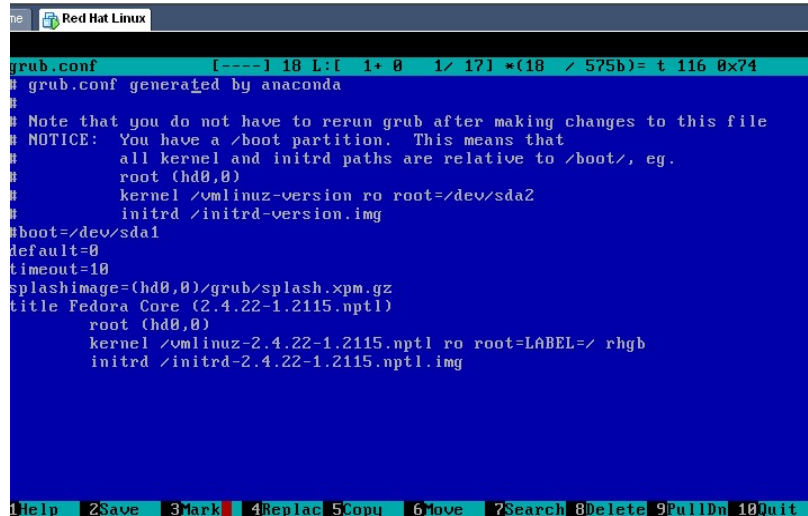
- **Label= Tên hệ điều hành**. Cho biết tên hệ điều hành nào sẽ xuất hiện trong giao diện của LILO.
- **initrd =filename**. File cung cấp thông tin khởi động hệ điều hành.
- **read only**: Hệ thống tập tin được mount với quyền chỉ đọc. Sau đó các thủ tục khởi động hệ thống sẽ mount lại với quyền read-write.

## GRUB

- GRUB (Grand Unified Bootloader) có chứa một số đặc tính quan trọng khác biệt so với LILO như sau:
  - GRUB cung cấp một môi trường trước hệ điều hành (Pre-OS). Điều này cho phép người dùng có khả năng linh động trong việc khởi nạp hệ điều hành với các thông số nào đó, cũng như nhận được thông tin về hệ thống.
  - GRUB hỗ trợ chế độ LBA (Logical Block Addressing). LBA chuyển giao quá trình chuyển đổi địa chỉ được dùng để tìm các tập tin cho firmware của ổ đĩa cứng, do đó nó sẽ không gặp phải vấn đề giới hạn dưới 1024-cylinder của BIOS.



- **GRUB** được cấu hình trong tập tin **/boot/grub/grub.conf**:



```

grub.conf      [----] 18 L: 1+ 0 1/ 171 *(18 / 575b)= t 116 0x74
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You have a /boot partition. This means that
# all kernel and initrd paths are relative to /boot/, eg.
# root (hd0,0)
# kernel /vmlinuz-version ro root=/dev/sda2
# initrd /initrd-version.img
#boot=/dev/sda1
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Fedora Core (2.4.22-1.2115.npt1)
    root (hd0,0)
    kernel /vmlinuz-2.4.22-1.2115.npt1 ro root=LABEL=/ rhgb
    initrd /initrd-2.4.22-1.2115.npt1.img

```

- Các lựa chọn gần như tương tự trong **/etc/lilo.conf**. Chú ý
  - **splashimage= file**. Xác định vị trí của tập tin ảnh màn hình giao diện menu của GRUB
  - **hiddenmenu**. Lệnh này sẽ không hiển thị giao diện menu của GRUB. Để hiển thị menu của GRUB phải nhấn **ESC** trong quá trình khởi động.



### 5.1.2. Tiến trình (Process)

- Một tiến trình là một “yêu cầu” hay “hoạt động” của một chương trình.
- Một chương trình là một danh sách các chỉ dẫn cho máy tính thực hiện. Một chương trình phần lớn được lưu dưới dạng tập tin nhị phân.
- Để chạy một chương trình, nó cần được sao chép (hoặc nạp) vào bộ nhớ chính của máy tính.
- Việc thi hành các chỉ dẫn của chương trình sẽ tạo ra các tiến trình.

- **Mỗi tiến trình có những thuộc tính sau:**
  - **PID:** Mã nhận diện tiến trình. Đây là một con số mà nhân hệ điều hành sử dụng để nhận diện một tiến trình.
  - **PPID:** Mã số nhận diện tiến trình cha, là tiến trình phát sinh ra tiến trình hiện hành.
  - **UID và GID:** Mã nhận diện tài khoản người dùng và nhóm đã tạo ra tiến trình.
  - **Độ ưu tiên tiến trình.** Độ ưu tiên của tiến trình có giá trị từ -20 (độ ưu tiên cao nhất) đến +19 (độ ưu tiên thấp nhất). **Giá trị này được biểu diễn bằng NI trong lệnh ps và top.**

- **Có 3 loại tiến trình chính trên Linux:**

- **Tiến trình tương tác** (Interactive processes): là tiến trình khởi động và quản lý bởi Shell, kể cả tiến trình foreground hoặc background.
- **Tiến trình thực hiện theo lô** (Batch processes): là tiến trình không gắn liền đến bàn điều khiển (terminal) và được nằm trong hàng đợi để lần lượt thực hiện.

- **Tiến trình ẩn trên bộ nhớ (Daemon processes)**  
Là các tiến trình chạy ẩn bên dưới hệ thống (background).

- Các tiến trình này thường được khởi tạo một cách tự động sau khi hệ thống khởi động.
- Đa số các chương trình Server cho các dịch vụ chạy theo phương thức này. Hầu hết các dịch vụ trên Internet như Mail, Web, Domain Name Service .... đều được thi hành theo nguyên tắc này.
- Các chương trình loại này được gọi là các chương trình daemon và tên của nó thường được kết thúc bằng ký tự “d” như **named**, **dhcpcd**, **httpd**...

### 5.1.3. Tiến trình init

- **Init** là một trong những chương trình thiết yếu đối với hoạt động của hệ thống Linux.
- **Init** có nhiều nhiệm vụ quan trọng như là khởi nạp các thẻ hiện của getty (để người dùng có thể đăng nhập), thực thi các mức thi hành chương trình và quản lý những tiến trình vô chủ (mất tiến trình cha).

- Khi nhân của hệ điều hành đã tự khởi động bản thân nó ( nạp vào bộ nhớ, đã chạy, và đã khởi nạp tất cả các trình điều khiển thiết bị,...) nó sẽ kết thúc quá trình khởi động hệ điều hành bằng cách khởi động một chương trình trong chế độ người dùng là chương trình **init**.
- **Init** là tiến trình cha của tất cả các tiến trình khác trên hệ thống Red Hat Linux. Vai trò của nó là gọi khởi động các tiến trình khác thông qua một script là **/etc/inittab**.

- Khi **init** khởi động, nó kết thúc quá trình khởi động hệ điều hành bằng một số tác vụ quản lý như là kiểm tra hệ thống tập tin, xóa các tập tin tạm thời trong /tmp, khởi động các dịch vụ khác nhau và khởi động một getty cho mỗi terminal và console ảo để người dùng có thể đăng nhập.
- Khi shutdown hệ thống, **init** điều khiển thứ tự và các script thực hiện tiến trình kết thúc.
- Chú ý: Không thể kết thúc tiến trình **init** khi hệ thống còn hoạt động.

### • Các bước khởi động của init

Trong quá trình khởi động của mình, chương trình **/sbin/init** sẽ thực hiện các thủ tục sau

1. Đầu tiên **init** gọi thi hành script **/etc/rc.d/rc.sysinit** để thiết lập đường dẫn, khởi động swap, kiểm tra hệ thống tập tin... Về cơ bản, **rc.sysinit** quản lý tất cả mọi thao tác mà hệ thống cần phải thực hiện tại thời điểm khởi động.

2. **Init** thực hiện script **/etc/inittab**. Script này mô tả cách thức hệ thống cần được thiết lập tại mỗi mức thi hành (run level) và thiết lập mức thi hành mặc định. Mỗi khi thay đổi mức thi hành, **init** sử dụng các script có trong thư mục **/etc/rc.d/init.d** để khởi động và dừng thi hành các dịch vụ khác nhau.
3. **Init** gọi thi hành script **/etc/rc.d/init.d/functions**. Script này cho biết cách thức khởi động hay ngừng một chương trình và cách thức xác định PID của một chương trình.

4. **Init** khởi động tất cả các tiến trình “ngầm” cần thiết cho hệ thống hoạt động bằng cách thi hành các script có trong thư mục **/etc/rc.d/rc0.d/**, **.../etc/rc.d/rc6.d/** tương ứng với mức thi hành được xác định.
- Các script trong thư mục **/etc/rc.d/rc[0-6].d/** là các symbolic link của các script có trong thư mục **/etc/rc.d/init.d/** được thi hành đối với mỗi trạng thái khởi động.
- Các script có tên bắt đầu là **S** là script khởi động, tên bắt đầu là **K** là script kết thúc. Những chữ số kế tiếp ký tự đầu tiên này chỉ ra thứ tự thi hành của script (giá trị thấp có độ ưu tiên cao)

5. Gọi thi hành script **/etc/rc.d/rc.local**.  
Script này dùng để bổ sung thêm các lệnh cần thiết cho môi trường hệ thống.
6. Sau khi **init** đã xử lý tất cả các mức thi hành, script **/etc/inittab** phát sinh một tiến trình getty cho mỗi giao tiếp ảo (virtual console) cho mỗi mức thi hành (mức 2-5 có sáu tiến trình getty, mức 1 chỉ có duy nhất một tiến trình, mức 0 và mức 6 không có tiến trình)

#### 5.1.4. Các mức thi hành và cấu hình **/etc/inittab**

- Ý tưởng của việc khởi động các dịch vụ khác nhau tại mỗi mức thi hành khác nhau nhằm để giải quyết thực tế rằng các hệ thống khác nhau có thể được sử dụng theo những cách thức khác nhau.
- Một số dịch vụ không thể được sử dụng trong khi hệ thống đang ở một trạng thái hay chế độ nào đó.
- Ví dụ khi muốn sửa lỗi đĩa ta cần khởi động hệ thống ở mức 1 để không người dùng nào có thể đăng nhập hệ thống khi sửa lỗi.

- Mức thi hành là một trạng thái của **init** và toàn thể hệ thống định nghĩa các dịch vụ hệ thống nào được khởi động.
- Red Hat Linux hỗ trợ 7 mức thi hành, mỗi mức thi hành được nhận diện bởi một con số như sau:
  - **0**: ngừng thi hành hệ thống (shutdown)
  - **1**: Chỉ một người dùng
  - **2**: Không sử dụng (người dùng có thể định nghĩa)
  - **3**: Chế độ đa người dùng đầy đủ.

- **4**: Không sử dụng (người dùng có thể định nghĩa).
- **5**: Chế độ đa người dùng trong môi trường X
- **6**: Khởi động lại máy.
- Thông thường Linux hoạt động ở mức **3** hoặc **5**
- Mức thi hành mặc định cho một hệ thống được cấu hình trong tập tin **/etc/inittab**. Tuy nhiên ta có thể chọn lại mức thi hành tại quá trình khởi động qua LILO hay GRUB

- Tập tin **inittab** chứa các mô tả cách thức tiến trình init cần thực hiện để thiết lập hệ thống tại mức thi hành nào đó.
- Khi **init** khởi động, nó đọc tập tin cấu hình **/etc/inittab**. Mỗi dòng **inittab** có cấu trúc gồm bốn trường, mỗi trường được phân cách nhau bằng ':' (nó cũng có thể chứa các dòng trắng và dòng nào được bắt đầu bởi dấu **#** là dòng ghi chú

**id: runlevels: action: process**

```

inittab [----] 4 L: 6+ 0 6/ 541 *(202 /1666b)= 32 0x20
#      Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit

```



- **Trong đó**

- **id**: Dùng nhận diện các dòng trong tập tin, nó phải có giá trị duy nhất và có chiều dài tối đa là 4 ký tự. Đối với các dòng của getty hay các tiến trình đăng nhập khác, nó phải là tên của terminal tương ứng. Ví dụ tty1.
- **runlevels**: Danh sách các mức thi hành mà tiến trình được thực hiện trong mức đó.
- **action**: Mô tả hành động sẽ được thực hiện
- **process**: Xác định tiến trình được thi hành.

- **Hành động thực hiện- action**

- **initdefault**: Xác định mức thi hành sẽ được thực thi sau khi khởi động hệ thống. Trường process không được xét đến trong hành động này.
- **sysinit**: Gọi thực hiện một tiến trình trong quá trình khởi động hệ thống. Trường runlevels phải được bỏ trống.
- **respawn**: Gọi thực hiện lại tiến trình ngay sau khi nó bị kết thúc.

- **wait**: Gọi thi hành một tiến trình tại mức đã định và init sẽ đợi nó kết thúc.
- **ctrlaltdel**: Gọi thực hiện một tiến trình init khi nhận được tín hiệu Ctrl+Alt+Del
- **once**: Tiến trình sẽ được thi hành một lần tại mức thi hành đã chỉ ra.

## 5.2. Quản lý tiến trình- Các tập tin và lệnh cơ bản

### 5.2.1. Hiển thị thông tin tiến trình- lệnh ps

- Với mỗi tác vụ trên hệ thống được Linux xử lý như một tiến trình và mỗi tiến trình và mỗi tiến trình đều có một mã nhận diện và một tên gọi. Lệnh **ps** cho biết các tiến trình hiện hành.
- **Cú pháp**  
**ps [option]**

- Lệnh này có một số lựa chọn và đối số, tuy nhiên đối với hầu hết nhà quản trị thì chỉ cần sử dụng một vài lựa chọn phổ biến và phổ biến nhất là thi hành **ps -aux** có tác dụng liệt kê danh sách tiến trình đang chạy cùng thông tin của nó bao gồm **chủ nhân (owner)**, **mã tiến trình (PID)**, **thời gian sử dụng CPU (%CPU)**, **mức chiếm dụng bộ nhớ (%MEM)**, **trạng thái của tiến trình (STAT)** và các thông tin khác cũng như **tên lệnh của bản thân tiến trình**.

- **Chú ý:**
  - Lệnh **ps -l**: Hiển thị thông tin đầy đủ về tiến trình.
  - Một số trạng thái tiến trình thường gặp bao gồm **R**-đang thi hành; **S**-đang chờ (sleeping); **Z** ngừng thi hành. **W**- Không đủ bộ nhớ cho tiến trình thi hành.

### 5.2.2. Hiển thị thông tin sử dụng tài nguyên

- Lệnh **top** hiển thị một danh sách các tiến trình hệ thống theo thời gian thực. Nó thống kê số lượng tiến trình cùng với trạng thái của chúng, tình trạng sử dụng CPU, sử dụng bộ nhớ...
- **Cú pháp**  
**top [option]**

- **Option**
  - d time**: Chỉ ra thời gian trễ giữa hai lần cập nhật thông tin trạng thái. Mặc định là 5 giây.
  - p [PID]**: chỉ theo dõi tiến trình có mã tiến trình là [PID]
  - c**: Hiển thị đầy đủ dòng lệnh thay vì chỉ hiển thị tên lệnh tạo tiến trình.

### 5.2.3. Kết thúc một tiến trình

- Lệnh **kill** cho phép kết thúc một tiến trình
- **Cú pháp**  
**kill -9 PID**

Trong đó PID là mã số nhận diện tiến trình muốn kết thúc.  
(man kill -> see more)

### 5.2.4. Thi hành lệnh ở chế độ ngầm (background)-tiến trình hậu cảnh

- Có thể điều khiển chế độ thi hành của một lệnh. Việc thi hành một lệnh chiếm nhiều thời gian ở chế độ background là hữu ích vì khi đang thực hiện một tác vụ ở chế độ ngầm thì terminal không bị chiếm dụng, do đó vẫn có thể thi hành các lệnh khác mà không cần chờ đến khi terminal được giải phóng như khi thực hiện lệnh ở chế độ bình thường (foreground)-tiến trình tiền cảnh

- Để thi hành một lệnh ở chế độ **background**, ta đặt ký tự **&** tại cuối dòng lệnh.
- Ta có thể thực thi nhiều lệnh ở chế độ ngầm. Mỗi lệnh chạy ở chế độ **background** sẽ được nhận diện như là một tác vụ độc lập và được cung cấp một tên và mã số tác vụ.
- Khi thực hiện lệnh ở **chế độ ngầm**, hệ thống sẽ thông báo mã số tác vụ và mã số tiến trình của lệnh.
- **Ví dụ**  
**top &**

- Lệnh **jobs** cho biết danh sách các tác vụ đang thi hành trong chế độ **background**.
- Mỗi tác vụ trong danh sách chứa mã số tác vụ, trạng thái của tác vụ là đang thi hành hay đã dừng và tên lệnh thi hành.
- Ký hiệu **+** chỉ ra tác vụ đang được xử lý và ký tự **-** chỉ ra tác vụ sẽ được xử lý kế tiếp.

- Ta có thể thực hiện chuyển một tác vụ đang thi hành ở chế độ **background** lên thực thi ở chế độ bình thường bằng lệnh **fg**.
- Trường hợp nếu có nhiều tác vụ đang thi hành ở chế độ **background** thì phải sử dụng lệnh **fg** với đối số là mã số tác vụ được đặt sau dấu **%**.
- Ví dụ  
**fg %3**

- Ta cũng có thể chuyển một tác vụ đang thi hành trong chế độ bình thường vào thi hành ở chế độ **background** bằng lệnh **bg**.
- Tuy nhiên không thể chuyển một cách trực tiếp tác vụ đang được thi hành sang chế độ **background**, mà trước tiên phải tạm dừng tác vụ đó bằng **CTRL+Z**, sau đó mới chuyển nó sang chế độ **background**.
- Ví dụ  
**top**  
**Ctrl+Z**  
**bg**

### 5.2.5. Lệnh service

- Dùng để xem trạng thái (status), dừng (stop), chạy (start), chạy lại (restart) một dịch vụ trên hệ thống
- **Cú pháp**  
`service <dịch vụ> status|stop|start|restart`
- **Ví dụ**  
`service sshd status`  
`service network restart`  
`service dhcpd stop`

### 5.2.6. Setting Priorities with nice

- The **nice utility** can be used on Linux to launch a program with a different priority level.
- Recall from our previous discussion of **top** and **ps** that each process running on your system has a **PR** and **NI** value associated with it.
- The **PR** value is the process' kernel priority. The higher the number, the higher the priority of the process. The lower the number, the lower the priority of the process.
- The **NI** value is the nice value of the process. The nice value is factored into the kernel calculations that determine the priority of the process. The nice value for any Linux process can range between -20 and +19. Again, the lower the number, the higher the priority of the process.



- You can't directly manipulate the **PR** of a process, but you can manipulate the **NI**
- The syntax for using nice

***nice -n nice\_level command***

- example: ***nice -n -2 vi***
- You can use the **renice** command to adjust the nice value of a process that is currently running on the system. The syntax for using this command

***renice nice\_value PID***

- example: ***renice -5 11***