

CHƯƠNG 6: QUẢN LÝ TIẾN TRÌNH – TÀI NGUYÊN

GV: LƯƠNG MINH HUẤN

NỘI DUNG

- I. Định nghĩa tiến trình
- II. Quản lý tiến trình
- III. Lập lịch
- IV. Quản lý tài nguyên với Quota

I. ĐỊNH NGHĨA TIẾN TRÌNH

- Tiến trình là một thực thể điều khiển đoạn mã lệnh cho chương trình hay dịch vụ trong hệ thống.
- Một tiến trình bao gồm: Thành phần văn bản (mã của chương trình), thành phần dữ liệu (những biến toàn cục)
- Mỗi tiến trình mang một định danh gọi là **PID** (Process Identification). Process ID là một con số lớn hơn 0 và là duy nhất. Hệ thống dựa vào các PID này để quản lý các tiến trình.
- Một tiến trình khi thực hiện nếu sinh ra nhiều tiến trình con thì được gọi là tiến trình cha (parent process).
- Khi tiến trình cha bị dừng thì tất cả tiến trình con cũng sẽ bị dừng.

I. ĐỊNH NGHĨA TIẾN TRÌNH

➤ Số trong dấu () là PID của tiến trình

```
[root@localhost ~]# pstree -np
init(1)
├── udevd(503)
├── auditd(1605)
│   ├── {auditd}(1606)
│   └── audispd(1607) ── {audispd}(1608)
├── restorecond(1621)
├── rpcbind(1641)
├── rpc.statd(1660)
├── rpc.idmapd(1697)
├── pcscd(1746)
│   ├── {pcscd}(1756)
│   ├── {pcscd}(2410)
│   └── {pcscd}(2571)
├── rsyslogd(1755) ── {rsyslogd}(1757)
├── rklogd(1760)
├── dbus-daemon(1775) ── {dbus-daemon}(1777)
├── named(1801)
│   ├── {named}(1802)
│   ├── {named}(1803)
│   └── {named}(1804)
├── automount(1821)
│   ├── {automount}(1822)
│   ├── {automount}(1823)
│   ├── {automount}(1826)
│   └── {automount}(1829)
├── setroubleshootd(1840)
│   ├── {setroubleshootd}(1981)
│   └── {setroubleshootd}(1982)
└── acpid(1850)
```


I. ĐỊNH NGHĨA TIẾN TRÌNH

➤ Có ba loại tiến trình chính trên Linux:

- Tiến trình tương tác (Interactive processes): là tiến trình khởi động và quản lý bởi shell
- Tiến trình thực hiện theo lô (Batch processes): là tiến trình không nằm ở terminal mà nằm ở hàng đợi để lần lượt thực hiện.
- Tiến trình ẩn trên bộ nhớ (Daemon processes): là tiến trình nằm ẩn dưới hệ thống. Các tiến trình thường khởi tạo lúc khởi động một cách tự động. Đa số các chương trình server chạy dưới hình thức này. Các chương trình loại này được gọi là chương trình daemond và tên của nó thường được kết thúc bằng chữ “d”. Ví dụ: named,...

I. ĐỊNH NGHĨA TIẾN TRÌNH

➤ Các trạng thái của tiến trình:

- Running: các lệnh của tiến trình đang được thực hiện
- Sleeping: tiến trình có trong bộ nhớ nhưng không làm gì cả
- Uninterruptable Sleep: tiến trình đang chờ đợi tài nguyên
- Terminated: sự thực thi của tiến trình kết thúc
- Zombie: tiến trình dừng nhưng chưa kết thúc hẳn vì còn đang chờ phản hồi của tiến trình cha

➤ Chỉ có một tiến trình ở trạng thái running tại một thời điểm

➤ Có thể có nhiều tiến trình ở trạng thái sleeping

I. ĐỊNH NGHĨA TIẾN TRÌNH

TIẾN TRÌNH TIỀN CẢNH

- Khi thực hiện một chương trình từ dấu đợi lệnh (\$ hoặc #), chương trình sẽ thực hiện và hệ thống không xuất hiện dấu đợi lệnh cho đến khi thực hiện xong chương trình.
- Do đó, chúng ta không thể thực hiện các công việc khác trong khi chương trình này đang thực hiện.

Ví dụ:

```
#find / -name pro -print
```

```
#find / -name pro -print > timkiem.txt
```

I. ĐỊNH NGHĨA TIỀN TRÌNH

TIỀN TRÌNH HẬU CẢNH

- Là tiến trình sinh ra độc lập với tiến trình cha. Khi thực hiện tiến trình hậu cảnh, chúng ta vẫn có thể thực hiện các công việc khác.
- Để tiến trình chạy dưới chế độ hậu cảnh, chúng ta thêm dấu & vào sau lệnh.

Ví dụ:

```
#find / -name pro -print > ketqua.txt &
```


I. ĐỊNH NGHĨA TIỀN TRÌNH

- Chúng ta có thể kiểm tra chương trình này có hoạt động không bằng lệnh: `#ps -aux | grep find`
- Đơn giản hơn, chúng ta dùng lệnh `#jobs` để xem các tiến trình đang có ở hậu cảnh.

[1] + Running `find / -name pro -print > results.txt &`

[1] Done `find / -name pro -print`

- Việc cho phép các tiến trình chạy hậu cảnh giúp chúng ta cho phép nhiều tiến trình hoạt động đồng thời.

I. ĐỊNH NGHĨA TIẾN TRÌNH

Một số khái niệm khác:

➤ Tín hiệu (signal)

- Là những thông điệp đơn giản được sử dụng để thông báo cho tiến trình về một sự kiện nào đó xảy ra mà không cần sự tác động của user
- Tín hiệu được sử dụng thông qua tên hay số thứ tự, vd:
 - Signal 15, TERM: terminal cleanly
 - Signal 9, KILL: terminal immediately
 - Signal 1, HUP: Re-read configuration file
- Xem tất cả các tín hiệu: man 7 signal

I. ĐỊNH NGHĨA TIẾN TRÌNH

➤ Thứ tự ưu tiên (Scheduling Priority)

- Quy định trình tự tiến trình được CPU xử lý
- Được gán thông qua giá trị: nice
- Giá trị nice chạy từ -20 đến 19, mặc định là 0
- Giá trị nice càng nhỏ thì độ ưu tiên của tiến trình càng cao.

II. QUẢN LÝ TIẾN TRÌNH

1. Xem thông tin tiến trình
2. Tìm kiếm tiến trình
3. Tạm dừng tiến trình
4. Đánh thức tiến trình
5. Hủy tiến trình
6. Định độ ưu tiên cho tiến trình
7. Lệnh khác

I.1 XEM THÔNG TIN TIỀN TRÌNH

- Ta có thể kiểm tra các tiến trình đang chạy bằng câu lệnh ps (process status). Lệnh ps có nhiều tùy chọn và phụ thuộc vào user đăng nhập.
- Cú pháp: **#ps <tùy chọn> <tham số>**
- Một số tùy chọn
 - ux: hiển thị tất cả các tiến trình mà user kích hoạt
 - T : xem tiến trình được chạy tại terminal hiện tại của user
 - aux : xem tất cả các tiến trình trong hệ thống
 - u username : xem tất cả các tiến trình của user nào đó

I.1 XEM THÔNG TIN TIỀN TRÌNH

- Ví dụ: Lệnh #ps

PID TTY STAT TIME COMMAND

41 v01S 0:00 -bash

134 v01 R 0:00 ps

```
[root@test ~]# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.3	2192	560	?	S	18:27	0:00	init [3]
root	2	0.0	0.0	0	0	?	SN	18:27	0:00	[ksoftirqd/0]
root	3	0.0	0.0	0	0	?	S<	18:27	0:00	[events/0]
root	4	0.0	0.0	0	0	?	S<	18:27	0:00	[khelper]
root	5	0.0	0.0	0	0	?	S<	18:27	0:00	[kacpid]
root	16	0.0	0.0	0	0	?	S<	18:27	0:00	[kblockd/0]
root	26	0.0	0.0	0	0	?	S	18:27	0:00	[pdflush]
root	27	0.0	0.0	0	0	?	S	18:27	0:00	[pdflush]
root	29	0.0	0.0	0	0	?	S<	18:27	0:00	[aio/0]
root	17	0.0	0.0	0	0	?	S	18:27	0:00	[khubd]

I.2 TÌM KIẾM TIỀN TRÌNH

- **Lệnh pgrep:** lệnh này cho phép tìm PID của một tiến trình hệ thống.
 - Ví dụ: tìm PID của firefox: `pgrep firefox`

```
[root@Server ~]# chkconfig httpd on
[root@Server ~]# service httpd start
Starting httpd:
[root@Server ~]# service httpd status
httpd (pid 5465 5464 5463 5462 5461 5460 5459 5458 5456) is running
[root@Server ~]# pgrep httpd
5456
5458
5459
5460
5461
5462
5463
5464
5465
[root@Server ~]# _
```

II.3 TẠM DỪNG TIẾN TRÌNH

- Đưa tiến trình đang chạy vào hậu cảnh bằng dấu &
- Xem tiến trình trong hậu cảnh bằng lệnh: **#jobs**

[1] + Stopped find / -name pro –print > results.txt

- **Lệnh bg:** dùng thi hành tiến trình trong hậu cảnh

Cú pháp: **#bg** <số thứ tự tiến trình>

#bg 1

#find / -name pro –print > results.txt

#jobs

[1] + Running find / -name pro –print > results.txt

II.3 TẠM DỪNG TIẾN TRÌNH

```
aaronkilik@tecmint ~ $ tar -czf home.tar.gz . &  
[1] 6016  
aaronkilik@tecmint ~ $  
aaronkilik@tecmint ~ $ jobs  
[1]+  Running                  tar -czf home.tar.gz . &
```

Annotations:

- Red box around `tar -czf home.tar.gz . &` with an arrow pointing to "Job/Command".
- Blue box around `[1]` with an arrow pointing to "Job ID Running".
- Blue box around `6016` with an arrow pointing to "Process ID".
- Red box around `jobs` with an arrow pointing to "List Running Jobs".

Tecmint.com
Linux Howto's Guide

Start Linux Command in Background and Detach Process in Terminal

II.3 TẠM DỪNG TIẾN TRÌNH

- Ta cũng có thể dùng lệnh service:
- Cú pháp: service process_name stop
- VD: service httpd stop

```
[root@build ~]# service iptables stop
iptables: Flushing firewall rules: [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules: [ OK ]
[root@build ~]# service iptables stop
iptables: Flushing firewall rules: [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules: [ OK ]
[root@build ~]# service iptables status
iptables: Firewall is not running.
[root@build ~]# /sbin/iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source destination
[root@build ~]#
```

II.4 ĐÁNH THỨC TIẾN TRÌNH

- **Lệnh fg:** dùng để đưa một tiến trình từ hậu cảnh sang tiền cảnh

Cú pháp: **#fg <số thứ tự tiến trình>**

- Ngoài ra, ta cũng có thể dùng lệnh service:

Cú pháp: **service process_name start**

VD: **service httpd start**

```
[root@node2 ~]# service httpd start
Starting httpd: [ OK ]
[root@node2 ~]# nmap localhost

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2006-07-29 13:32 PDT
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1657 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https

Nmap run completed -- 1 IP address (1 host up) scanned in 1.219 seconds
[root@node2 ~]#
```

II.5 HỦY TIẾN TRÌNH

➤ **Lệnh kill:** dùng để hủy một tiến trình

Cú pháp: **#kill -9 <PID tiến trình>**

-9 : tính hiệu dừng tiến trình không điều kiện.

Lưu ý: Chỉ có người dùng root mới có quyền dừng tất cả các tiến trình. Đối với những người dùng khác chỉ được dừng các tiến trình do mình tạo ra.

➤ **Bằng PID:** *kill [signal] PID*

▪ **VD:** *kill -9 3428*

➤ **Bằng tên:** *pkill [signal] comm*

▪ **VD:** *kill -TERM cupsd*

II.5 HỦY TIẾN TRÌNH

- Có 3 mức tín hiệu phổ biến được sử dụng với câu lệnh KILL đó là các mức tín hiệu **1, 9 và 15**.
 - **Mức tín hiệu 1 (SIGHUP)** có chức năng dừng process và khởi động lại process. Như vậy nếu chúng ta muốn khởi động lại một process nào đó chúng ta sử dụng câu lệnh kill -1 process.
 - **Mức tín hiệu 9 (SIGKILL)** là mức tín hiệu cao nhất có chức năng dừng một process ngay lập tức.
 - **Mức tín hiệu 15 (SIGTERM)** có chức năng dừng một process. Đây là mức tín hiệu mặc định khi kill một process.

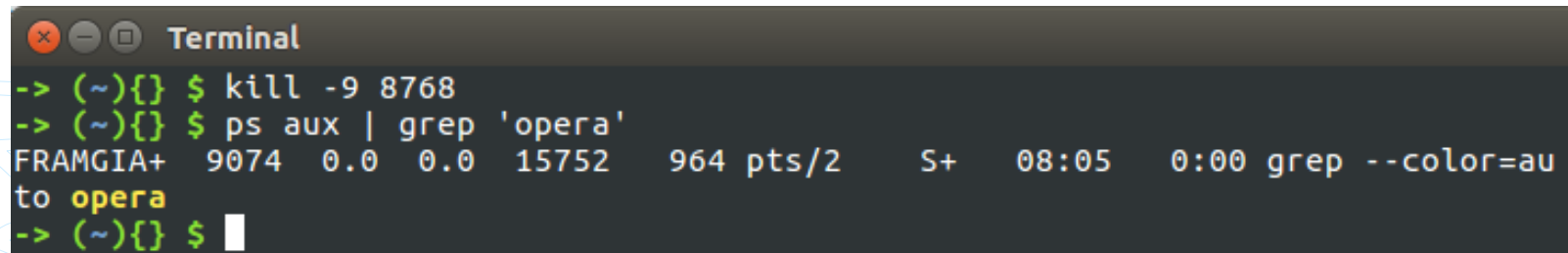
II.5 HỦY TIẾN TRÌNH

- Ví dụ: ta cần tắt đi tiến trình của trình duyệt Opera.
 - Để xác định PID, ta sử dụng lệnh `ps aux | grep 'opera'`

```
Terminal
-> (~){} $ ps aux | grep 'opera'
FRAMGIA+  8768 22.7  1.3 1087944 112552 ?        Sll  08:02   0:00 /usr/lib/x86_64
-linux-gnu/opera/opera
FRAMGIA+  8776  0.3  0.5 463568 40768 ?        S    08:02   0:00 /usr/lib/x86_64
-linux-gnu/opera/opera --type=zygote
FRAMGIA+  8778  0.0  0.0 463568 7624 ?        S    08:02   0:00 /usr/lib/x86_64
-linux-gnu/opera/opera --type=zygote
FRAMGIA+  8822  6.3  0.9 542588 78772 ?        Sl   08:02   0:00 /usr/lib/x86_64
-linux-gnu/opera/opera --type=gpu-process --field-trial-handle=17135449986909921
984,6086614955924888561,131072 --supports-dual-gpus=false --gpu-driver-bug-worka
rounds=1,9,10,27,34,70,84 --disable-gl-extensions=GL_ARB_timer_query GL_EXT_time
r_query GL_KHR_blend_equation_advanced GL_KHR_blend_equation_advanced_coherent -
--disable-accelerated-video-decode --gpu-vendor-id=0x8086 --gpu-device-id=0x5912
--gpu-driver-vendor=Mesa --gpu-driver-version=17.2.4 --gpu-driver-date --service
-request-channel-token=2C40C7AB2BE08EF05FC8C9EFF050BED0
FRAMGIA+  8828  0.0  0.1 471124 10116 ?        S    08:02   0:00 /usr/lib/x86_64
-linux-gnu/opera/opera --type=gpu-broker
FRAMGIA+  8843 26.0  1.4 899500 116376 ?        Sl   08:02   0:00 /usr/lib/x86_64
-linux-gnu/opera/opera --type=renderer --field-trial-handle=17135449986909921984
,6086614955924888561,131072 --service-pipe-token=A7EB8F474960DE5B22CB703F6BBECB5
A --lang=en-US --disable-client-side-phishing-detection --enable-offline-auto-re
load --enable-offline-auto-reload-visible-only --enable-pinch --num-raster-threa
ds=2 --enable-main-frame-before-activation --content-image-texture-target=0,0,35
53;0,1,3553;0,2,3553;0,3,3553;0,4,3553;0,5,3553;0,6,3553;0,7,3553;0,8,3553;0,9,3
```

II.5 HỦY TIẾN TRÌNH

- Trình duyệt Opera chạy rất nhiều tiến trình, vậy ta thử tắt chúng đi, sử dụng lệnh **kill -9 PID**, , ở trên hình ta sẽ thử tắt tiến trình có PID = 8768



```
Terminal
-> (~){} $ kill -9 8768
-> (~){} $ ps aux | grep 'opera'
FRAMGIA+ 9074  0.0  0.0  15752   964 pts/2    S+   08:05   0:00 grep --color=au
to opera
-> (~){} $
```

II.6 ĐỊNH ĐỘ ƯU TIÊN CHO TIẾN TRÌNH

- Xem độ ưu tiên của tiến trình:
 - `ps -o comm, nice`
- Thiết lập giá trị ban đầu cho độ ưu tiên:
 - `nice -n 5 comm`
- Thay đổi giá trị của độ ưu tiên:
 - `renice 5 PID`
- Process với độ ưu tiên thấp hơn sẽ chỉ chạy khi nó được yêu cầu (nếu CPU power hết mức sử dụng). Giá trị từ -20 đến 19. Giá trị càng thấp, thì độ ưu tiên càng cao. Mặc định **tất cả** process là **0**.

II.7 MỘT SỐ LỆNH KHÁC

- Lệnh top: in ra những tiến trình đang chạy trên hệ thống, update thông tin sau mỗi 5s
- `gnome-system-monitor`
- Lệnh watch:
 - VD: `watch -n 2 ps -ef`: thực hiện lại lệnh xem tất cả các tiến trình đang chạy trên hệ thống sau mỗi 2s

II.7 MỘT SỐ LỆNH KHÁC

```
top - 10:38:39 up 33 min,  2 users,  load average: 0.00, 0.01, 0.05
Tasks:  85 total,   2 running,  83 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.
KiB Mem:  1017936 total,  294112 used,  723824 free,   11744 buffers
KiB Swap: 2826236 total,    0 used, 2826236 free.  117820 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1055	teamspe+	8	-12	763192	18292	7320	S	0.7	1.8	0:07.86	ts3ser+
2148	root	20	0	123528	1496	1096	S	0.3	0.1	0:04.03	top
1	root	20	0	46144	6652	3932	S	0.0	0.7	0:01.25	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthrea+
3	root	20	0	0	0	0	S	0.0	0.0	0:00.04	ksofti+
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworke+
6	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworke+
7	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migrat+
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/0
10	root	20	0	0	0	0	S	0.0	0.0	0:00.24	rcu_sc+

III. LẬP LỊCH

➤ Quản lý một công việc thường bao gồm:

- Chạy 1 tiến trình dưới nền HDH:
 - VD: `gedit test.txt &`
- Tạm thời dừng 1 tiến trình: `Ctrl-Z`
- Liệt kê các công việc: **jobs**
- Chuyển 1 công việc từ foreground sang background: `bg [%jobnum]`
- Chuyển 1 công việc từ background sang foreground : `fg [%jobnum]`

III. LẬP LỊCH

- Có những công việc lặp đi lặp lại nhiều lần hoặc dự định thực hiện ở một khoảng thời gian sắp tới. Do đó, ta cần thực hiện lập lịch để tự động thực hiện công việc. Ví dụ: backup, đồng bộ dữ liệu.
- **Chương trình at:** thực hiện các công việc ở thời điểm định trước.

Cú pháp: **#at [time]**

<các lệnh thực hiện>

<Ctrl + D>

III. LẬP LỊCH

Hoặc sử dụng lệnh **#at [time] < tập_lệnh** để có thể thực hiện nhiều lệnh cùng một lúc.

- Kiểm tra các tiến trình đã nhập vào: **#at -l**
- Hủy bỏ các công việc đã nhập vào: **#at -r [job-number]**
- VD: **at 0200 → Nhập công việc → Hoàn tất với Ctrl-D**
- Một số format của time:
 - at 8:00pm December 7
 - at midnight + 23 minutes
 - at 7 am Thursday
 - at now + 5 minutes

III. LẬP LỊCH

- **Chương trình lập lịch crontab:** cho phép lập lịch có tính chu kỳ. Những công việc lập lịch được định nghĩa trong một tập tin văn bản được tạo theo cú pháp sau:

phút giờ ngày_của_tháng tháng_của_năm ngày_của_tuần lệnh

Dùng lệnh sau để cài đặt tập tin lệnh:

#crontab [filename]

- Mỗi người dùng sẽ có 1 crontab trùng với tên username của mình để lưu tất cả lệnh cần thực hiện theo chu kỳ

III. LẬP LỊCH

➤ Các giá trị cho các trường:

Phút (0 – 59)

Giờ (0 – 23)

Ngày_của_tháng (1 – 31)

Tháng_của_năm (1-12)

Ngày_của_tuần (0 – 6, 0 is Sunday)

Lệnh (rest of line)

III. LẬP LỊCH

- Ví dụ: chúng ta muốn lập lịch cho kịch bản chạy vào 1 giờ sáng mỗi Thứ Sáu, chúng ta cần dùng lệnh sau:
- **crontab -e**: tạo lịch
- Cronjobs được viết theo định dạng sau:

```
0 1 * * 5 /bin/execute/this/script.sh
```


III. LẬP LỊCH

➤ Kịch bản này sẽ được thực thi khi giờ hệ thống:

- minute - phút: 0
- of hour - của giờ: 1
- of day of month - Của ngày trong tháng: (every day of month)
- of month - Của tháng: * (every month)
- and weekdays - Và ngày trong tuần: 5 (=Friday)

III. LẬP LỊCH

➤ **Chương trình lập lịch batch:** được thi hành khi mức tải của hệ thống dưới 20%.

Cú pháp: **#batch <câu lệnh>**

IV. QUẢN LÝ TÀI NGUYÊN VỚI QUOTA

- Quota được dùng để thiết lập hạn ngạch đĩa cho người dùng. Ta chỉ thiết lập quota trên những filesystem lưu trữ thông tin cho người dùng hoặc nhóm người dùng.
- MỘT SỐ KHÁI NIỆM
 - **Giới hạn cứng:** chỉ định dung lượng đĩa cứng tối đa mà người dùng có thể sử dụng.
 - **Giới hạn mềm:** cho phép người dùng vượt quá dung lượng cho phép trong một khoảng thời gian nào đó. Mặc định hệ thống cho phép thời gian 7 ngày.
 - **Thời gian gia hạn:** là thời gian cho phép người dùng vượt quá dung lượng đĩa cứng cho phép trong giới hạn mềm.

IV. QUẢN LÝ TÀI NGUYÊN VỚI QUOTA

➤ CÁC BƯỚC THỰC HIỆN

- Thiết lập tùy chọn quota trên file `/etc/fstab`.
- Kiểm tra hạn ngạch thông qua lệnh `quotacheck`.
- Phân bổ hạn ngạch thông qua lệnh `edquota`.

➤ CẤU HÌNH TẬP TIN `/etc/fstab`

- Mở tập tin `/etc/fstab` để thêm một số thông số giới hạn `usrquota` (cho người dùng), `grpquota` (cho nhóm).

IV. QUẢN LÝ TÀI NGUYÊN VỚI QUOTA

- Trong ví dụ bên dưới, ta đặt cấu hình hạn ngạch trên /home
- Khởi động lại hệ thống để remount lại file system /home

```
/dev/md0      /   ext3  defaults          1 1
LABEL=/boot   /boot ext3  defaults          1 2
none dev/pts   devpts  gid=5,mode=620    0 0
LABEL=/home   /home ext3
defaults,usrquota,grpquota 1 2
none          /proc  proc  defaults        0 0
none         /dev/shm tmpfs  defaults        0 0
/dev/md1      swap  swap   defaults          0 0
```

IV. QUẢN LÝ TÀI NGUYÊN VỚI QUOTA

➤ KIỂM TRA QUOTA

- Cú pháp: **#quotacheck -avug**

Ví dụ: *#quotacheck -avug*

Scanning /dev/sda9 [/home] done

Checked 236 directories and 695 files

Using quotafile /home/quota.user

Using quotafile /home/quota.group

- Thông tin cấu hình quota của người dùng được lưu trong file */home/aquota.user*, cấu hình của nhóm được lưu trong file */home/aquota.group*.

IV. QUẢN LÝ TÀI NGUYÊN VỚI QUOTA

➤ PHÂN PHỐI QUOTA

- Cú pháp: **#edquota** <option> <username>

Ví dụ: #edquota -u hv

Disk quotas for user mp3user (uid 503):

Filesystem	Block	Soft	Hard	Inode	Soft	Hard
/dev/hdc3	24	0	0	7	0	0

IV. QUẢN LÝ TÀI NGUYÊN VỚI QUOTA

➤ KIỂM TRA VÀ THỐNG KÊ QUOTA

▪ Kiểm tra Quota

Cú pháp: #quota [options] <user/group>

▪ Thống kê Quota

Cú pháp: #repquota [options] <filesystem>

