Công ty An ninh mạng Viettel

Báo cáo về Process monitoring and scheduling

Sinh viên: Nguyễn Đan Trường

*MỤC LỤC*

[*Chương 1. Process monitoring 3*](#_Toc152076501)

[*1. process 3*](#_Toc152076502)

[*a. Foreground and background process 7*](#_Toc152076503)

[*2. Kill process 9*](#_Toc152076504)

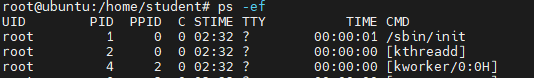
[*Chương 2. Lập lịch trong hệ thống 14*](#_Toc152076505)

# Process monitoring

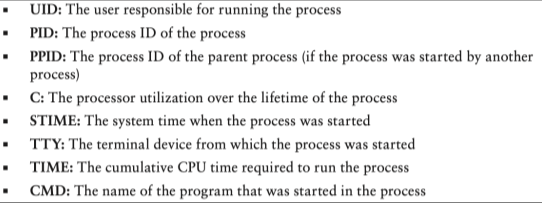
## process

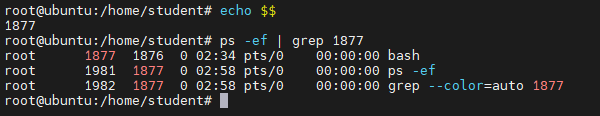
Trong hệ điều hành Linux, "process" (tiến trình) đề cập đến một chương trình hoặc một tác vụ đang chạy trên hệ thống.

**PS**



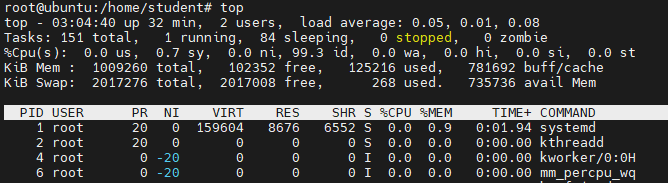
Công cụ ps giúp chúng ta xem thông tin về các tiến trình đan chạy trong hệ thống





**TOP**

top là công cụ giúp theo dõi hiệu suất hệ thống theo thời gian thực (3s), các thông tin mà top cung cấp bao gồm thông tin chi tiết về: CPU, bộ nhớ, swap, tải hệ thống, danh sách các tiến trình đang chạy.



Hàng đầu tiên:

* Load Average: thông tin tải trung bình của hệ thống trong 1/5/15 phút.
* Thông tin thời gian: hiển thị thời gian hệ thống đã chạy và thời gian tương đối.

Hàng thứ 2:

* Số lượng tiến trình và thread: Tổng số tiến trình, số tiến trình đang chạy, số tiến trình đang chờ (stopped), số tiến trình đang ngủ (sleeping), và số thread.

Hàng thứ 3:

* Thông tin tải CPU: % chiếm dụng do user, % chiếm dụng do system

Hàng thứ 4:

* Thông tin về bộ nhớ RAM: tổng dụng lượng RAM (Kib), lượng RAM đang free, lượng RAM đang được sử dụng, buff/cache – dung lượng bộ nhớ được sử dụng để lưu trữ đệm và bộ nhớ cache. Dữ liệu đệm thường chứa các dữ liệu được sử dụng thường xuyên để giảm thời gian truy xuất từ đĩa cứng.

Hàng thứ 5:

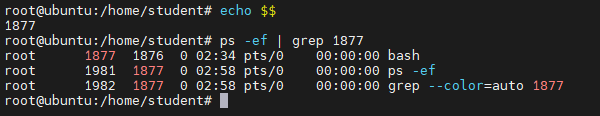
* Thông tin về việc sử dụng phân vùng swap: tổng dung lượng swap trên hệ thống (KiB), free bao nhiêu, đang sử dụng bao nhiêu, **avail Mem** - số lượng bộ nhớ RAM có sẵn để sử dụng ngay lập tức.

Hàng thứ 6: hiển thị thông tin các tiến trình đang chạy trong hệ thống

* PID: ID của tiến trình
* USER: người sở hữu tiến trình
* PR (priority): độ ưu tiên của tiến trình
* NI (nice value): giá trị nice
* VIRT (virtual mem): tổng lượng bộ nhớ ảo sử dụng bởi tiến trình
* RES (Resident mem): lượng bộ nhớ vật lý đang được sử dụng
* SHR (Shared mem): lượng bộ nhớ được chia sẻ
* S (%CPU): % CPU đang sử dụng
* MEM (%MEM): % bộ nhớ RAM đang sử dụng
* TIME+: tổng thời gian CPU đã sử dụng bởi tiến trình

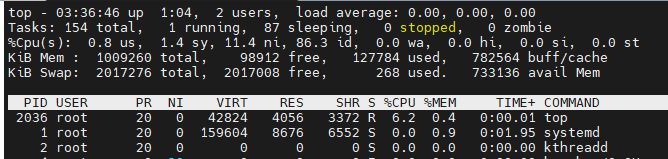
**Screen và tmux**

Screen và tmux là 2 công cụ quản lý cửa sổ làm việc trong terminal. Việc này giúp người dùng có thể chia màn hình thành nhiều cửa sổ làm việc hơn.

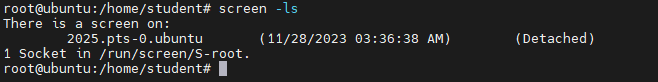


Screen



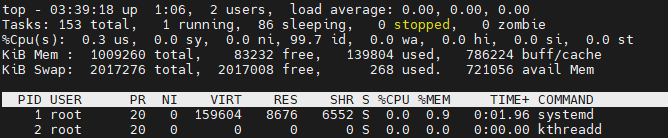


Bấm Ctrl + A + D để thoát khỏi màn hình làm việc



Kết nối lại:

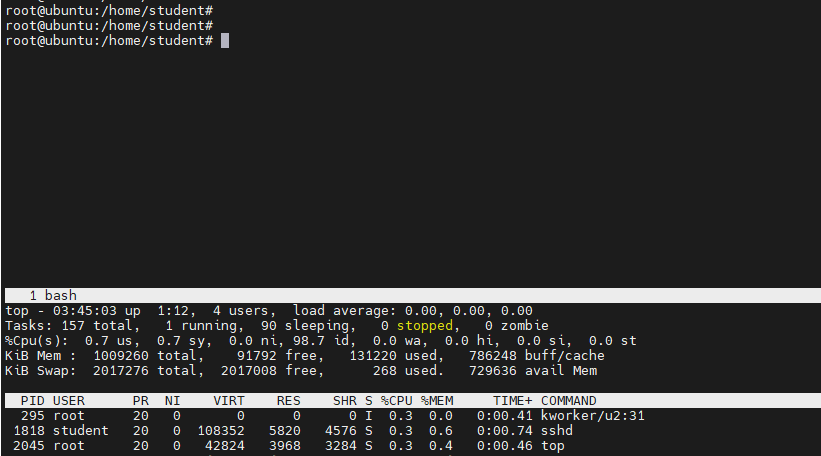




* Nếu thoát ra khỏi phiên làm việc thì screen vẫn còn

Ctrl a + S : thêm một cửa sổ => sau đó Ctrl a + c

Chuyển qua lại giữa các cửa sổ Ctrl a + tab



Một nhược điểm của screen là nếu thoát ra khỏi screen và reconnect lại thì screen chỉ giữ lại được 1 cửa sổ cuối cùng được tạo ra

Tmux khắc phục được nhược điểm này của screen, còn về chức năng thì tương tự như nhau.

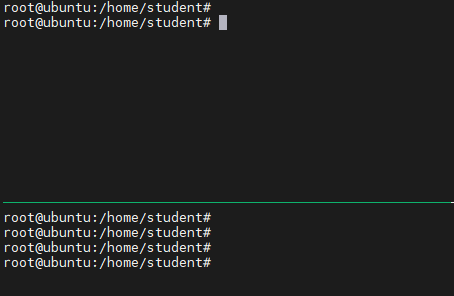
Ctrl-b c: Tạo một cửa sổ mới.

Ctrl-b % và Ctrl-b ": Chia màn hình thành các panel.

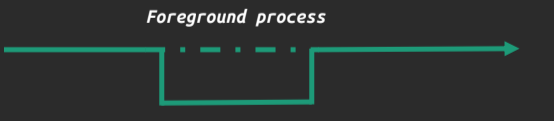
Ctrl-b +d: thoát khỏi màn hình

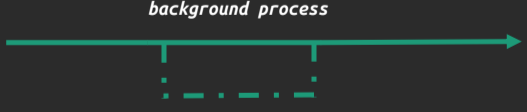
Ctrl-b +&: thoát khỏi và xóa session





### Foreground and background process



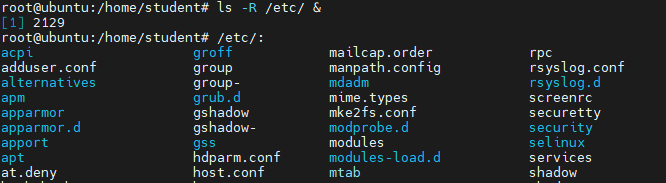


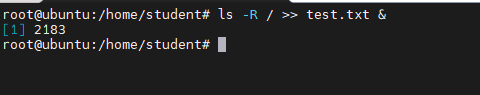
Trong hệ điều hành Linux, các tiến trình được chạy ở 2 chế độ là Foreground và background.

2 kiểu chạy này chủ yếu được phân biệt bằng việc chiếm dụng shell:

* Foreground: process chạy ở chế độ này thì người dùng phải chờ cho tới khi process chạy xong thì mới có thể có lại shell prompt
* Background: process chạy ở chế độ này sẽ chạy ngầm và lập tức trả lại shell promtp cho người dùng có thể tương tác tiếp với hệ thống

Chú ý: mặc dù background process sẽ trả lại shell promtp cho người dùng nhưng nếu process đó có những output thì mặc định nó vẫn sẽ hiển thì ra màn hình vì vậy nếu có output đối với background process thì nên redirect ra một file.





# Chạy tiến trình trong Foreground

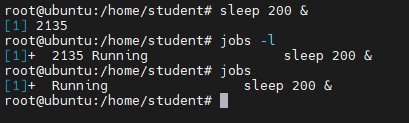
*./my\_program*

# Chạy tiến trình trong Background

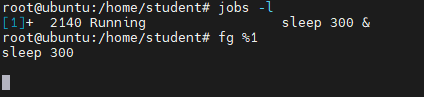
*./my\_program &*

**Quản lý backgound process**

*jobs [option] [job-id]*

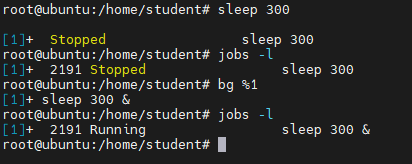
**

Chuyển process từ backgroup sang foreground sử dụng lênh *fg %id*



Chuyển process từ foreground sang background:

Sẽ không thể chuyển trực tiếp từ foreground sang background được mà đầu tiên phải stop process đó trước (Ctrl + Z) sau đó mới chuyển nó sang background



Bản chất khi tra sử dụng phím Ctrl +Z là đã đưa process về background nhưng ở trạng thái stopped. Để process tiếp tục chạy ở background thì ta cần dùng lệnh:

*bg %id*

1 jobs ở trạng thái stopped thì ta có 2 lựa chọn: đưa về lại foreground (fg %id) hoặc running ở background (bg %id)

## Kill process

**Process signal**

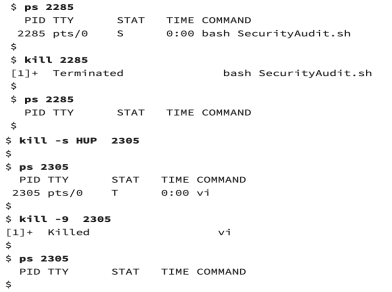
Trong hệ điều hành linux, process signal là cơ chế thông điệp gửi từ một tiến trình hoặc hệ thống đến một hoặc nhiều tiến trình khác. Signal thường được sử dụng để thông báo về sự kiện hoặc yêu cầu một hành động cụ thể từ một tiến trình.

Một số signal quan trọng:

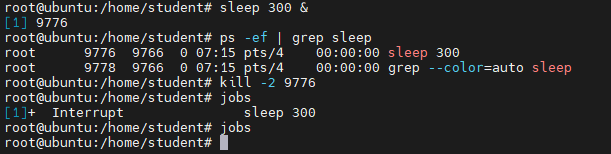
|  |  |  |
| --- | --- | --- |
| Number | Name | Description |
| 1 | HUP | thông báo rằng terminal đã đóng. Nó có thể được sử dụng để yêu cầu tiến trình làm mới hoặc chấm dứt. |
| 2 | INT | Ctrl+C, yều cầu tiến trình kết thúc |
| 3 | QUIT | Tương tự INT nhưng thường kèm theo thông điệp |
| 9 | KILL | Buộc kết thúc ngay lập tức mà không có cơ hộ xử lý hay lưu trạng thái. |
| 11 | SEGV | Thông báo rằng tiến trình đã cố gắng truy cập một phần của bộ nhớ mà nó không có quyền truy cập |
| 15 | TERM | Yêu cầu một tiến trình kết thúc, có thể thực hiện công việc dọn dẹp trước khi kết thúc |
| 17 | STOP | Thông báo răng một hoặc nhiều tiến trình con của một tiến trình đã thay đổi trạng thái. |
| 18 | TSTP | Sử dụng để tiếp tục một tiến trình đã bị tạm dừng |
| 19 | CONT | Tạm dừng một tiến trình. |

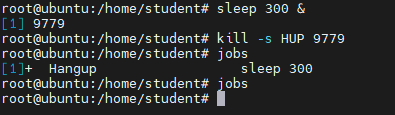
Truyền các tín hiệu tới process:

*kill [option] process*

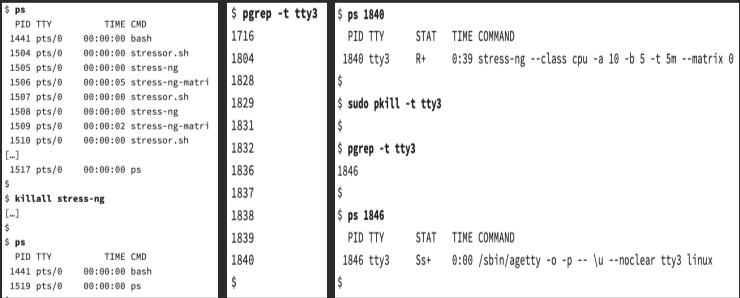
**

Mặc định kill không gửi signal thì sẽ là 15 – TERM

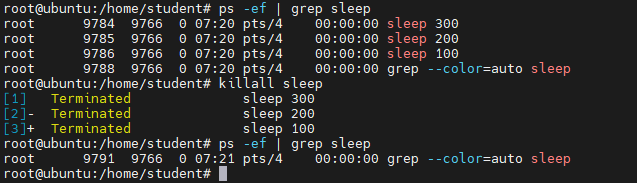




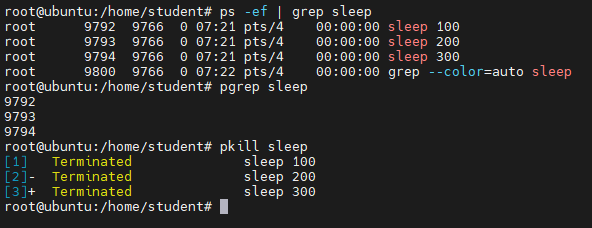
Ngoài công cụ kill để gửi signal tới process thì còn thêm các công cụ khác như killall, pkill:

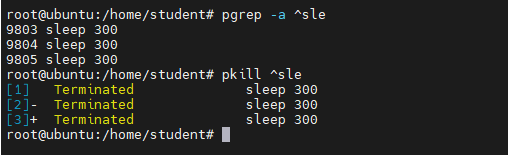


*killall* cho phép truyền tín hiệu tới 1 hoặc 1 nhóm các tiến trình cùng chạy bằng 1 command. (mặc định nếu không chỉ signal thì sẽ là 15)



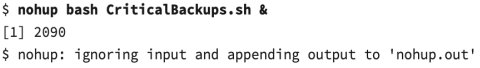
*pkill* cho phép gứi tín hiệu tới 1 nhóm process thỏa mãn điều kiện nào đó và được dùng kèm theo *pgrep*





Khi người dùng thoát khỏi shell (logout) thì shell process sẽ gửi signal 1 HUP tới cho các process con của nó và các process con có thể sẽ kết thúc. Trong những tình huống khi ta không muốn một process con kết thúc sau khi logout, công cụ nohup hỗ trợ việc này:

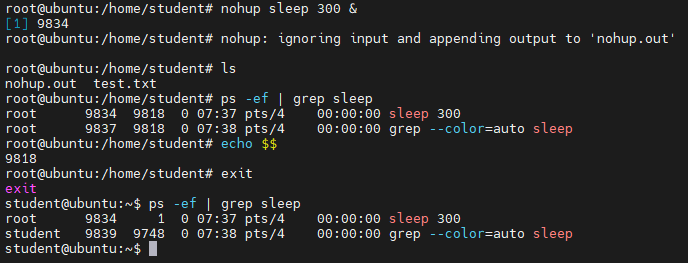
*nohup program/script &*



Bản chất của nohup là khi thoát khỏi shell làm việc thì process sẽ lấy process có ID là 1 (init process) làm PPID.

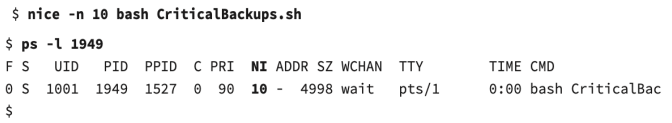
* Process chỉ down nếu init process kết thúc - ứng với việc hệ thống down

Nohup sẽ redirect phần standard output và error sang file nohup.out. Mặc định file nohup.out sinh ra tại vị chí đang đứng chạy lệnh nohup, nếu không có quyền ghi tại thư mục thì sẽ tạo file nohup.out ở thư mục /home



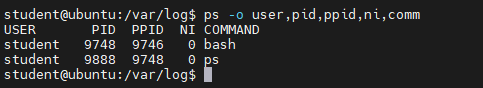
Trong hệ điều hành, sẽ có trường hợp những process được ưu tiên xử lý trước và việc này dựa trên priority của process

*nice –n VALUE Command*

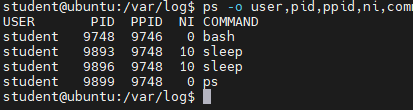
**

Nice value là chỉ số các định độ ưu tiên của process ( -20 – 19) càng nhỏ càng ưu tiên.

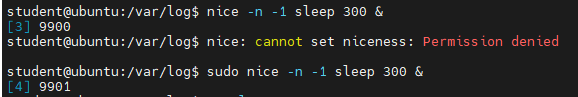
Mặc định các process được start lên thì đều có nice value = 0

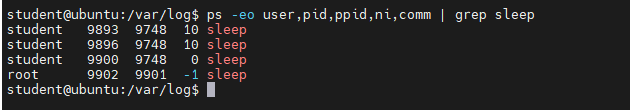


Công cụ nice giúp thay đổi độ ưu tiên của process



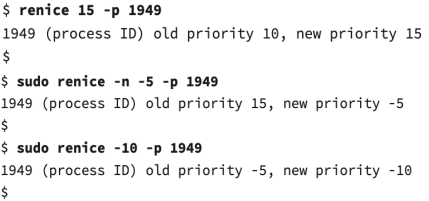
Với quyền user thông thường thì chỉ có thể gán nice value >0, chỉ có account root mới có thể gán nice value <0.

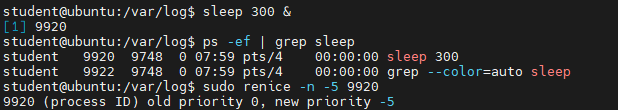




Để set lại nice value ho các process ta sử dụng công cụ renice:

*renice [–n] VALUE [PID] [-u Users] [-g Groups]*

**





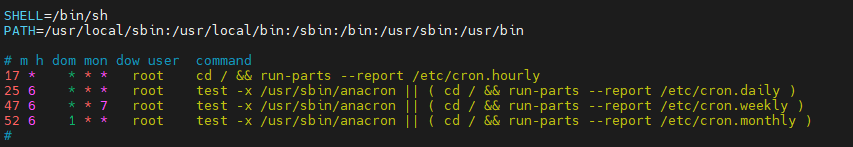
# Lập lịch trong hệ thống

Trong hệ điều hành Linux, cron là một công cụ quản lý thời gian được sử dụng để thực hiện các công việc định kỳ tự động theo lịch trình. Có hai loại crontab chính: system crontab và user crontab.

System crontab là crontab của hệ thống, được sử dụng để thực hiện các tác vụ hệ thống được lên lịch sẵn. Các tác vụ này thường được thực hiện bởi root hoặc các dịch vụ hệ thống. Ví dụ, hệ thống crontab có thể được sử dụng để thực hiện các tác vụ sau:

* Cập nhật các bản tin
* Tạo bản sao lưu
* Chạy các ứng dụng định kỳ
* Thực hiện các tác vụ bảo trì hệ thống

System contab được đặt lịch tại file */etc/contab* hoặc các tập tin trong thư mục */etc/cont.d/*



User crontab là crontab của người dùng, được sử dụng để thực hiện các tác vụ của người dùng được lên lịch sẵn. Các tác vụ này thường được thực hiện bởi người dùng đang hoạt động. Ví dụ, user crontab có thể được sử dụng để thực hiện các tác vụ sau:

* Cập nhật các ứng dụng
* Gửi email
* Chạy các tác vụ tự động

User contab được lưu vào file trong thư mục */var/spool/cron/crontabs*

## Cấu trúc file crontab

Cả system crontab và user crontab đều có cấu trúc giống nhau, bao gồm các trường sau:

* Minute: Xác định phút trong ngày. Giá trị có thể từ 0 đến 59.
* Hour: Xác định giờ trong ngày. Giá trị có thể từ 0 đến 23.
* Day of month: Xác định ngày trong tháng. Giá trị có thể từ 1 đến 31.
* Month: Xác định tháng trong năm. Giá trị có thể từ 1 đến 12.
* Day of week: Xác định ngày trong tuần. Giá trị có thể là:
  + SUN: Chủ nhật
  + MON: Thứ hai
  + TUE: Thứ ba
  + WED: Thứ tư
  + THU: Thứ năm
  + FRI: Thứ sáu
  + SAT: Thứ bảy
* Command: lệnh sẽ được thực thi

Ví dụ, dòng crontab sau sẽ thực thi lệnh */path/to/script* vào lúc 10 giờ sáng thứ ba hàng tuần:

*30 10 \* \* 2 /path/to/script*

Để sử dụng crontab, bạn cần sử dụng các lệnh sau:

* crontab -e: Mở file crontab để chỉnh sửa.
* crontab -l: Xem nội dung của file crontab.
* crontab -r: Xóa file crontab.

Việc crontab chỉ cho phép thực thi lệnh đến từng phút mà không phải từng giây là vì:

* hiệu suất: Chạy một tác vụ every second có thể gây ra tắc nghẽn hệ thống, đặc biệt là trên các hệ thống có nhiều người dùng và tác vụ.
* Sự cần thiết: Trong hầu hết các trường hợp, không cần thiết phải chạy một tác vụ every second. Hầu hết các tác vụ crontab có thể được lên lịch chạy every minute hoặc thậm chí every hour mà không ảnh hưởng đến chức năng của hệ thống.
* Hầu hết các tác vụ chạy mỗi giây thường là những tác vụ liên quan tới monitring và việc này thì không cần tới crontab.

Nếu cần thực hiện chạy lệnh mỗi giây 1 lần ta có thể thực hiện tạo 1 đoạn mã:

\* \* \* \* \* /path/to/your/script.sh

*#!/bin/bash*

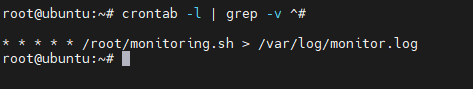
*while true; do*

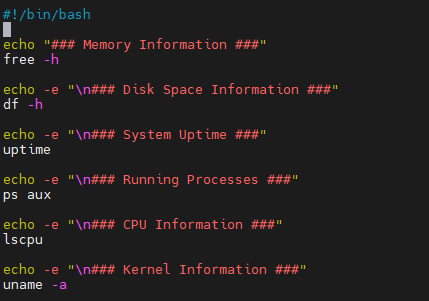
*# Your task here*

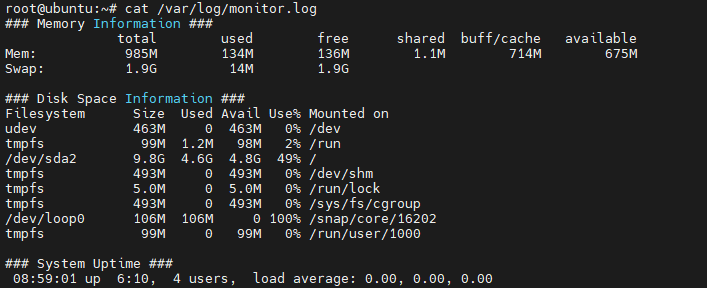
*sleep 1*

*done*

**

**

**

**

Ngoài ra ta có thể đặt chạy script 5 phút 1 lần:

*\*/5 \* \* \* \* command*

*\*/ number :*tương ứng với chu kỳ phút/giờ/ngày.



