Công ty An ninh mạng Viettel

Báo cáo tổng quan về Linux

Sinh viên: Nguyễn Đan Trường

*MỤC LỤC*

[*Chương 1. Device 3*](#_Toc151134951)

[*1. Các khái niệm cơ bản 3*](#_Toc151134952)

[*2. Quy ước tên device file 4*](#_Toc151134953)

[*Chương 2. Disk và partition 6*](#_Toc151134954)

[*1. Cấu trúc vật lý của ổ đĩa cứng 6*](#_Toc151134955)

[*2. Cấu trúc Logic của ổ đĩa cứng với kiến trúc phân vùng MBR 7*](#_Toc151134956)

[*3. Cấu trúc logic của ổ đĩa cứng với kiến trúc phân vùng GPT 8*](#_Toc151134957)

[*Chương 3. Linux File System 10*](#_Toc151134958)

[*1. Cấu trúc file system 10*](#_Toc151134959)

[*2. Tạo Filesystem 12*](#_Toc151134960)

[*Chương 4. LVM – Logical Volume Management 16*](#_Toc151134961)

[*1. Logical volume management 16*](#_Toc151134962)

[*2. Các bước để quản lý và tạo LVM 17*](#_Toc151134963)

# Device

## Các khái niệm cơ bản

Trong hệ điều hành Linux, thiết bị chung tâm của máy tính là CPU và RAM, ngoài 2 bộ phận này thì tất cả những bộ phận khác được gọi chung là thiết bị ngoại vi (peripheral device). Bộ vi xử lý, bộ nhớ và các thiết bị ngoại vi được nối với nhau bằng các dây điện rất nhỏ gọi là **bus**, tất cả được gắn trên một bảng mạch điện tử gọi là mainboard.

**Phân loại Device**

Dựa trên cách thức đọc/ghi dữ liệu device được chia thành 2 loại:

* **Block device**: dữ liệu trên các device này mỗi lần được đọc/ghi theo từng block (1 block = 512 bytes to 32KB) chứ không phải theo từng byte. Dữ liệu được kiểu này thường dùng để lưu trữ lâu dài.
* **Character device**: dữ liệu trên các device này được đọc/ghi theo từng character (1 byte). Các device này chẳng hạn như mouse làm nhiệm vụ chuyển thao tác di chuột thành dữ liệu về vị trí con trỏ trên màn hình, luôn có dữ liệu thay đổi liên tục theo thời gian và không lưu trữ để sau này dùng lại.

*# ls -l /dev/ | grep "^b"*

*# ls -l /dev/ | grep "^c"*

**Số major và minor của device**

Mỗi device được đặc trưng bởi 2 chỉ số này:

* Số **major** trỏ tới loại driver tương ứng với thiết bị trong linux
* Số **minor** xác định thiết bị cụ thể

File /proc/devices cho phép xem các thông tin về số hiệu *major* và tên module thiết bị đã đăng ký với số major đó.

|  |  |
| --- | --- |
| *# cat /proc/devices*  *Character devices:*  *136 pts*  *Block devices:*  *8 sd* | # ls -l /dev/pts/0 /dev/sda  crw--w---- 1 root tty 136, 0 Aug 21 11:22 /dev/pts/0  brw-r----- 1 root disk 8, 0 Aug 21 10:52 /dev/sda |

**Driver**

Driver là chương trình điều khiển device. Khi hệ điều hành muốn tương tác với device nó phải thông qua driver. Các driver phổ biến thường được cài sẵn trong nhân Linux, hoặc có thể được cài đặt như một module tách dời của nhân.

**Device file**

Việc truy cập một device nếu phải thông qua driver thì rất phức tạp, đòi hỏi người dùng phải có kiến thức sau về phần cứng. Trong linux, việc này được đơn giản bằng các “device file”.

Device file là các file nằm trong thư mục /dev, mỗi file này đại diện cho một driver, người dùng thay vì tương tác với driver có thể tương tác với các device file này như nó là các file dữ liệu thông thường.

Việc kết nối giữa file thiết bị và driver được thực hiện thông qua 2 bước sau:

* Đăng ký số hiệu <major, minor> cho file thiết bị.
* Kết nối các thao tác thiết bị với các hàm tương ứng trong driver.

Ví dụ: /dev/audio là device file đến driver của sound card. Người dùng có thể sử dụng các lệnh làm việc với file thông thường như lệnh cat với device file.

# cat sample.au >/dev/audio

Nhân Linux sẽ redirect mọi thứ mà /dev/audio nhận được vào driver của card sound. Kết quả là đoạn âm thanh sample.au chạy, lệnh này thường được dùng để kiểm tra card sound.

## Quy ước tên device file

**Block device:**

/dev/fd\_: floppy disk.

/dev/hd\_: HDD hoặc CDROM chuẩn IDE

/dev/hd\_ \_: partition trên HDD chuẩn IDE.

/dev/sd\_: HDD chuẩn SCSI/SATA hoặc USB removable disk

/dev/sd\_ \_: partition trên HDD chuẩn SCSI, SATA hoặc USB removable disk

/dev/ram\_: ramdisk

**Character device**

Mouse: /dev/input/mice

Keyboard: /dev/input

Monitor: /dev/tty\_, /dev/vcs\_, /dev/console

Sound card: /dev/audio

Serial port (các cổng COM1, COM2, ...): /dev/ttyS\_

Parallel port, cũng còn gọi là line port (các cổng LPT1, LPT2): /dev/par\_ hoặc /dev/lp\_

**Virtual device** do Linux tự định nghĩa

/dev/urandom hoặc /dev/random: device sinh byte ngẫu nhiên

/dev/zero: device sinh các ký tự null

/dev/null: khi có dữ liệu vào, device này sẽ hủy bỏ và cũng không trả về dữ liệu ra

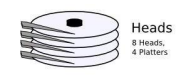
**Loop device** là cơ chế cho phép một file có chứa filesystem (điển hình là các file image như \*.iso, \*.img) có thể được mount như thể nó là một storage device. Khi đang làm việc với console mà cắm thêm một device vào máy thì có thể thông tin về device đó được hiển thị trên màn hình, nếu không hiển thị có thể dùng lệnh dmesg để xem. Lệnh **dmesg** (viết tắt của display message) cho biết các thông báo của nhân khi nó detect các thiết bị kề từ lúc nhân được tải vào RAM.

# Disk và partition

## Cấu trúc vật lý của ổ đĩa cứng

Mỗi ổ đĩa cứng gồm:

* Nhiều đĩa (plate) xếp chồng lên nhau
* Các đầu đọc (head) đọc/ghi trên mặt các đĩa. Mỗi mặt đĩa có một đầu đọc/ghi vì vậy nếu có 4 đĩa thì có 8 đầu đọc/ghi.



* Mạch điều khiển là mặc điện nằm phía sau ổ cứng, mạch này có các chức năng:
  + Điều khiển tốc độ quay đĩa
  + Điều khiển dịch chuyển các đầu từ
  + Mã hóa và giải mã các tín hiệu đọc ghi

**Cylinder – Head – Sector**

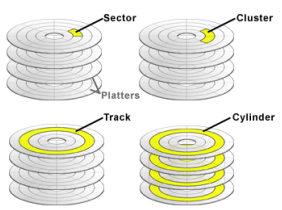
Mỗi một đĩa (plate) được chia thành nhiều rãnh (track) là các đường tròn đồng tâm

Tập hợp các track cùng bán kính ở các mặt đĩa khác nhau thành cylinder

Mỗi rãnh (track) được chia thành nhiều cung (sector), 1 Sector = 512 byte

Sector được xác định bằng bộ 3 chỉ số (Cylinder, Head, Sector)

→ Tọa độ CHS của một sector



Để tăng dung lượng của đĩa thì trong các đĩa cứng ngày nay, các Track ở ngoài được chia thành nhiều Sector hơn và mỗi mặt đĩa cũng được chia thành nhiều track hơn và như vậy đòi hỏi thiết bị phải có độ chính xác rất cao.

**Tính tổng dung lượng ổ đĩa cứng**

Dung lượng ổ đĩa cứng được tính = (số cylinder) × (số đầu đọc/ghi) × (số sector/track) × (số byte/sector)

Capacity = nC x nH x nS x 512

Ví dụ: Western Digital WD Caviar 26400 6.4GB Hard Drive

Drive Parameters: 13328 cyl – 15 heads – 63 spt – 6448.6 MB

» 13328 cylinders

» 15 heads

» 63 sectors per track

» 512 B/sector

Capacity = 13328 × 15 × 63 × 512 = 6,448,619,520 B

## Cấu trúc Logic của ổ đĩa cứng với kiến trúc phân vùng MBR

Master boot record (MBR) là kiểu đặc biệt cảu boot sector đặt tại sector đầu tiên trong Track zero của ổ cứng (Cylinder 0, head 0, sector 1), có dung lượng 512 byte gồm 3 thành phần chính:

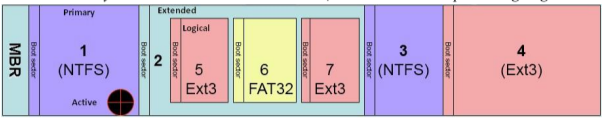
* Đoạn mã Bootloader code: nàm ở đầu MBR, có kích thước 446 byte, dùng để chứa chương trình bootrap khởi động Hệ điều hành
* Bảng phân hoạch đĩa (partition table): có kích thước 64 byte, chữa thông tin về partition như số thứ tự, tên ổ đĩa logic … Mỗi MBR có thể quản lý tối đã 4 primary partition, mỗi primary partition có kích thước 16 bytes.
* Signature Bytes 2 byte chỉ dấu hợp lệ ở cuối MBR có giá trị 0xAA55

Để khắc phục vấn đề mỗi ổ cứng MBR chỉ chứa được 4 phân vùng người ta khắc phục bằng cách dùng sector đầu tiên của partition thứ 4 để quản lý các phân vùng tiếp theo tương tự như 1 MBR.

Boot sector: là sector đầu tiên của một partition trong ổ cứng

Partition:

* HDD được chia thành các Partition với thuộc tính: primary, extended, logical
* Có tối đa 4 phân vùng Primary trên một ổ cứng
* Primary partition được đánh dấu Active để làm partition khởi động
* 1 Primary biến thành Extended để có thể được chia thành nhiều phân vùng logical.



**Đọc/ghi theo Sector/Cluster**

Việc đọc/ghi trên ổ đĩa cứng thực hiện trên sector ( = khối 512 byte = ½ KB) chứ không phải đọc/ghi từng bit (khác với RAM)

VD: file 800 bytes => cần 2 sector để lưu

Các hệ điều hành thường điều khiển việc đọc/ghi lên ổ đĩa cứng theo Cluster (block) = n sector

VD: Window XP/7/8

*1 cluster = 8 sectors = 4KB*

1 file có size 10 byte => phải dùng 1 cluster = 4KB để lưu

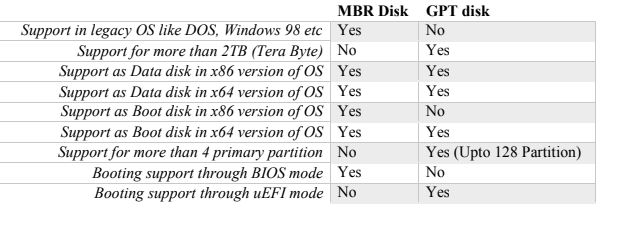
* Lãng phí space nhưng tăng performance

## Cấu trúc logic của ổ đĩa cứng với kiến trúc phân vùng GPT

Những hệ điều hành cũ với BIOS legacy và kiểu phân vùng MBR bị giới hạn bới kích thước phân vùng nhỏ hơn 2TB

**UEFI** là một chuẩn BIOS mới của Intel, có trên các bo mạch chủ đời mới được lưu trử ở thư mục EFI trong bộ nhớ non-volatile (bộ nhớ đảm bảo cho dữ liệu không bị hỏng mỗi khi mất điện). Ưu điểm của UEFI là nó giúp các hệ thống khởi động nhanh hơn, UEFI sử dụng bảng phân vùng GPT có khả năng quản lý các thiết bị có dung lượng lớn hơn 2TB.

**GPT** (GUID Partition Table) có thể xem là một miêu tả về các sắp xếp các phân vùng trên ổ đĩa.



**Cấu trúc ổ đĩa định dạng GPT**

Một ổ đĩa GPT được chia ra làm nhiều LBA (Logical Block Addressing). Thông thường, một LBA có kích thước là 512 byte, tuy nhiên kích thước có thể thay đổi lên đến 1024 byte hoặc 2048 byte.

* **LBA 0** sẽ có cấu trúc giống một ổ đĩa dạng MBR, nhằm giúp các phần mềm dựa trên MBR có thể hiểu được GPT nhằm tránh ghi đè.
* **LBA 1** sẽ gồm các header chứa GUID và thông tin về dung lượng, vị trí phân vùng.
* Các **LBA 2-33** chứa các GUID tương ứng với các phân vùng.

Một phiên bản của các LBA 1-33 sẽ được sao lưu ở vùng dữ liệu cuối của ổ đĩa.

Các phân vùng sẽ nằm sau LBA 33. Mỗi phân cùng sẽ được gán một GUID để đảm bảo tính duy nhất của phân vùng. Có thể chuyển một ổ đĩa dạng thô (raw) hoặc dạng MBR sang dạng GPT nhưng không thể giữ lại dữ liệu trước đó.

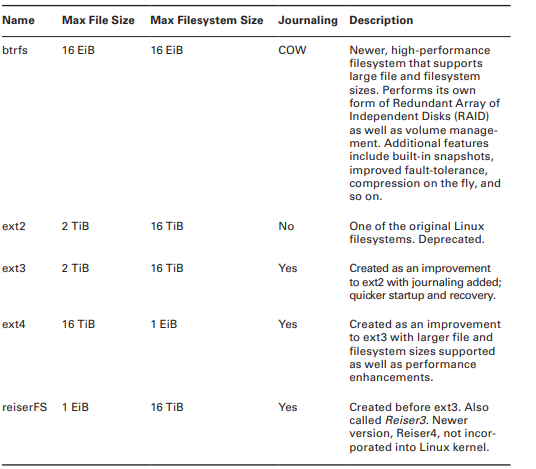
**Ưu điểm của ổ đĩa dạng GPT**

* Cho phép tạo đến 128 phân vùng primary.
* Cho phép chia phân vùng lớn hơn 2 TB (mức giới hạn đối với ổ đĩa dạng MBR)
* Tích hợp CRC32, một cơ chế kiểm tra lỗi, tăng tính ổn định cho bảng phân vùng.
* Mỗi phân vùng sẽ tương ứng với một GUID, thuộc tính và kiểu phân vùng sẽ do GUID quyết định.

# Linux File System

## Cấu trúc file system

File system được dùng để quản lý các dữ liệu được đọc/ghi vào các thiết bị lưu trữ. Mỗi loại file system có các phương pháp riêng để truy cập, cật nhật và bảo vệ tính toàn vẹn của dữ liệu. Các file system thường gặp là EXT2/3/4, NTFS…



**EXT2** hệ thống file system trên các bản phân phối linux cũ hiện tại không còn được sử dụng.

**EXT3 = EXT2 + Journaling** *(hệ thống file nhật ký => Một hệ thống file có tính năng log giúp đảm bảo toàn vẹn dữ liệu và tránh các trường hợp mất dữ liệu đột ngột. EXT3 có độ tin cậy cao vì có khả năng ghi nhớ quá trình thao tác dữ liệu, khôi phục nhanh hệ thống file khi có sự cố)*

Nâng cấp từ ext2 lên ext3: ***tune2fs –j /dev/sda1 ext3***

**EXT4** kế thừa từ ext3, kích thước file tối đa lên đến 16TB, kích thước file system tối đa đến 1EB (exabytes). Khả năng mở rộng thư mục: ext3 hỗ trợ 32000 thư mục con, ext4 không giới hạn việc tạo các thư mục con.

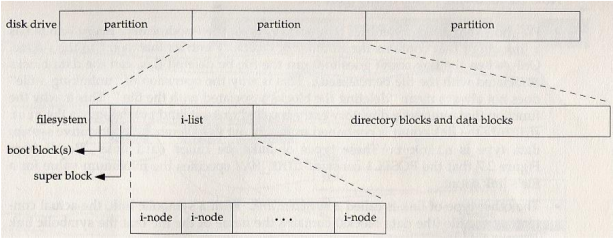
**BTRFS** hệ thống tệp mở rộng cho hệ điều hành linux, thiết kế để cung cấp nhiều tính năng hiện đại, linh hoạt. Được phát triển như một giải pháp thay thế ext4. Sử dụng cơ chế COW (Copy-On-Write) – khi một tệp hoặc một phần của tệp được thay đổi, btrfs tạo ra một bản sao mới của dữ liệu đó và chỉ thay đổi bản sao đó. Điều này giúp giảm thiếu rủi ro mất dữ liệu trong trường hợp lỗi đột suất.

**SWAP** hệ thống file dùng làm vùng đệm cho bộ nhớ chính. Thông thường, nó chỉ được sử dụng khi hệ thống thiếu hụt RAM hoặc chuyển trạng thái của máy tính về chế độ Hibernate.

**NTFS** định dang file trên window nhưng linux cũng hỗ trợ đọc/ghi.

Hệ thống file của linux có các thành phần sau:

* Boot Block
* Super Block
* I-nodes
* Data Block



Super Block là dùng để lưu trữ các thông tin:

* *Kích thước và cấu trúc file system*
* *Thời gian cật nhật file system cuối cùng*
* *Thông tin trạng thái*

**Inode**: lưu những thông tin về tập tin và thư mục được tạo trong file system. Mỗi tập tin tạo ra sẽ được phân bổ một inode lưu thông tin sau:

* *Loại tập tin và quyền hạn truy cập*
* *Chủ sở hữu tập tin*
* *Kích thước và số hard link đến tập tin*
* *Ngày và giờ chỉnh sửa tập tin lần cuối cùng*
* *Vị trí lưu nội dung tập tin trong file system*

*# tune2fs -l /dev/sda1 | grep Inode*

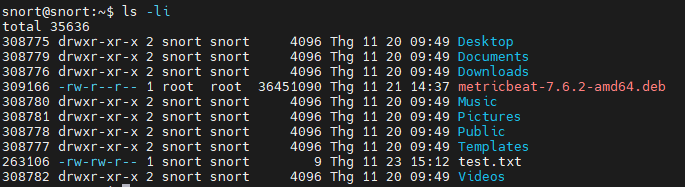
*Inode count: 26104*

*Inodes per group: 2008*

*Inode blocks per group: 251*

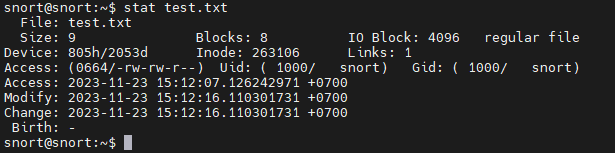
*Inode size: 128*

Mỗi inode được xác định bởi một số inode duy nhất bên trong filesystem



Từ kết quả ta thấy file test.txt có inode là 263106.

Ngoài ra, ta có thể dùng lệnh *stat <file\_name>* để kiểm tra số inode và thuộc tính của nó:



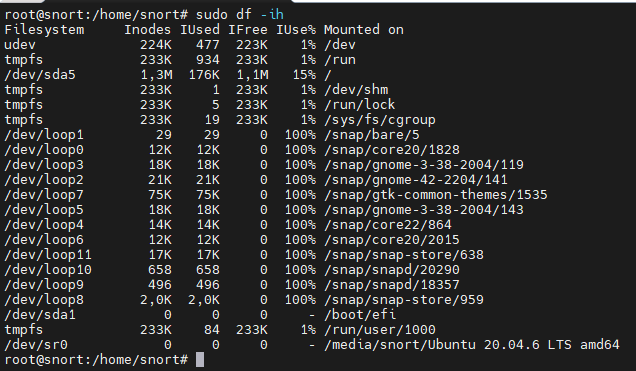
* Size 9: cho bieets kích thước tập tin là 9byte
* Blocks 8: số block trên đĩa cứng được sử dụng bới tập tin
* IO Block: 4096: kích thước block I/O, tức là dung lượng mà hệ điều hành sử dụng khi thực hiện các hoặc động I/O
* Device: 805h/2053d: thông tin về thiết bị lưu trữ mà tệp tin thuộc về
* Inode: 263106: số inode của tệp tin
* Links 1: số liên kết đến tệp tin
* Access: 2023-11-23 15:12:07.126242971 +0700: Thời điểm cuối cùng mà tệp tin được truy cập.
* Modify: 2023-11-23 15:12:16.110301731 +0700: Thời điểm cuối cùng mà nội dung của tệp tin được sửa đổi
* Change: 2023-11-23 15:12:16.110301731 +0700: Thời điểm cuối cùng mà thông tin metadata của tệp tin (ví dụ: quyền truy cập) được sửa đổi.
* Birth: - : thông tin về thời điểm tập tin được tạo

Trong thực tế, những file được tạo ra với những ký tự đặc biêt như ?,^… thì không dễ dàng xóa đi được, lúc này để xóa được những file đặc biệt , chúng ta cần tìm số inode của file đó và thực hiện xóa.





Một cách để chạy hết dụng lượng trong hệ thông file là sử dụng hết các inodes. Ngay cả khi có đủ dung lượng trống trên đĩa, bạn sẽ không thể tạo file mới.



**Data block**: là vùng lưu trữ dữ liệu thực sự của tập tin. Mỗi block thường chứa 1024 byte.

*# tune2fs -l /dev/sda1 | grep -i “block size”*

*Block size: 4096*

## Tạo Filesystem

Để có thể đọc ghi file trong linux cần làm các bước sau:

B1: tạo phân vùng (partition) phân hoạch trên đĩa cứng.

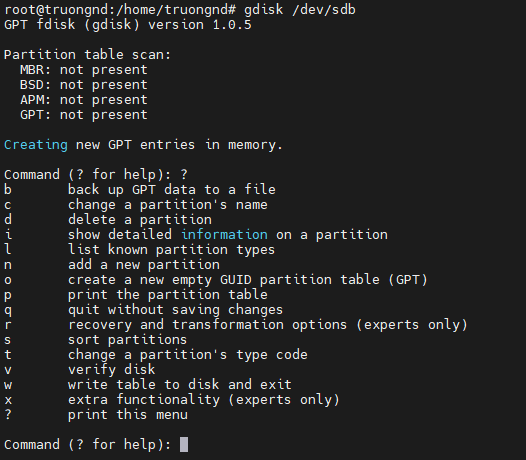
B2: tạo file system trên các partition đã tạo – format partition theo định dạng nào đó.

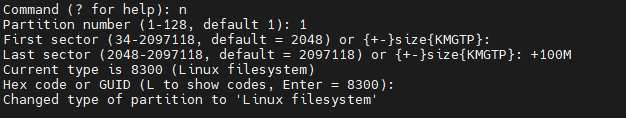
B3: mount filesystem đã được format vào một vị trí trên cây thư mục.

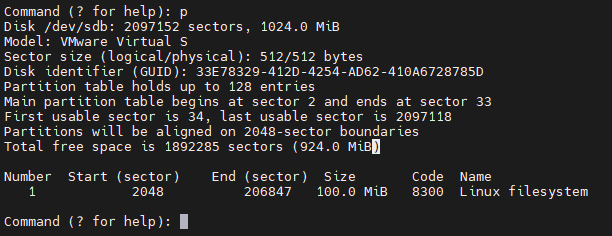
**Tạo phân vùng (Partition)**

Sử dụng công cụ *gdisk* để tạo partition với chuẩn GPT, *fdisk* với chuẩn MBR

Ngoài ra còn có 2 công cụ *parted* và *gparted*: tương tự với 2 công cụ trên nhưng cho phép thay đổi kích thước partition.

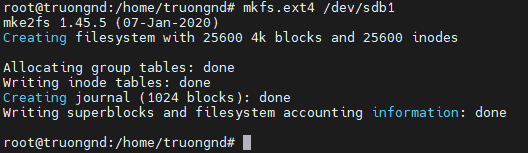




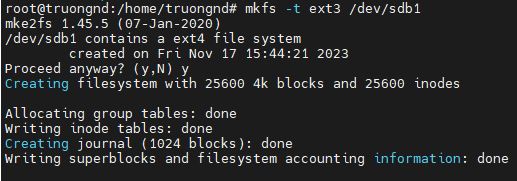


**Thực hiện định dạng parttion**

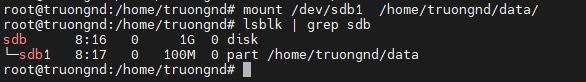
Công cụ chính để định dạng partition là *mkfs*.



Format lại sang ext3



**Thực hiện mount filesystem**

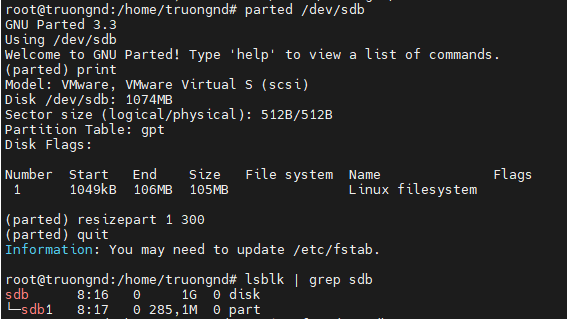
****

Nếu không sử dụng hoặc muốn mount vào thư mục khác cần umount trước sau đó mount lại vào thư mục mới.

**Thay đổi kích thước partition:**

Nếu phân vùng đang được mount vào một thư mục nào đó thì cần umount trước.

Sau đó sử dụng các công cụ như *parted, gparted, fdisk* để thay đổi kích thước phân vùng.



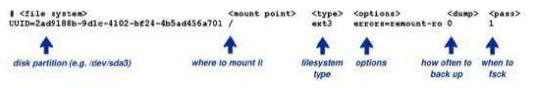
Sau khi thay đổi phân kích thước phân vùng cần sử dụng lênh *resize2fs* để cật nhật lại kích thước phân vùng

Cấu hình file /etc/fstab để quản lý mount các device lúc khởi động hệ thống.

File **/etc/fstab** có thể có nhưng device mount không thành công do ghi không đúng qui tắc, không tìm thấy mount point hay đường dẫn device bị thay đổi… Những partition đã mount thành công được ghi trong file **/etc/mtab**.

Dùng lệnh mount không tham số hay lệnh **cat /etc/mtab** để xem danh sách những device đã mount được, nếu chỉ quan tâm đến những storage device được mount thì dùng lệnh **df –h** (hiển thị thông tin về không gian đĩa đang sử dụng của hệ thông)

**Cấu trúc file /etc/fstab**

****

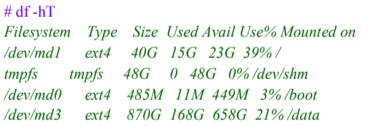
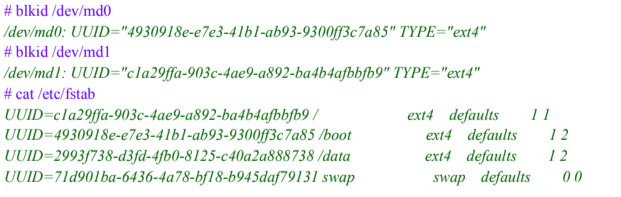
Mỗi dòng trong /etc/fstab chứa thông tin về một device. Các cột ở mỗi hàng được phân cách bởi khoảng trắng.

* **Cột 1**: cho biết tên device (phân vùng ổ cứng, CD/DVD, USB, ISO image...). Đồng thời cũng cho biết đường dẫn tới file đại diện cho thiết bị (device file) . Trong Linux, mọi tài nguyên phần cứng lẫn phần mềm đều được xem là file, các device file thường nằm ở thư mục /dev

Có thể sử dụng UUID "Universally Unique IDentifier" thay cho tên device. Kiểm tra UUID gắn với device bằng lệnh **blkid**



* **Cột 2**: đường dẫn thư mục (mount point)
* **Cột 3**: kiểu filesystem của device: ext, swap, …
* **Cột 4**: các tùy chọn khi mount. Nếu có nhiều tùy chọn thì chúng được phân các nhau bởi dấu phẩy “,”.
  + Auto: tự động mount khi hệ thống khởi động
  + Noauto: không tự động mount, nếu muốn sử dụng thiết bị thì sau khi khởi động vào hệ thống cần chạy lệnh mount
  + User: cho phép người dùng thông thường được quyền mount
  + Nousser: chỉ cho ngường dùng root mới có quyền mount
  + Exec: cho phép chạy các file nhị phân trên thiết bị
  + Noexec: không cho phép chạy các file nhị phân trên thiết bị
  + Ro (read-only): chỉ cho phép quyền đọc trên thiết bị
  + Rw (read-write): cho phép quyền đọc/ghi trên thiết bị
  + Defaults: tương đương với tập các tùy chọn *rw, exec, auto, nouser*
* **Cột 5**: tùy chọn cho công cụ sao lưu dump, 0: bỏ qua việc sao lưu, 1: thực hiện sao lưu
* **Cột 6**: tùy chọn cho công cụ dò lỗi fsck, 0: bỏ qua việc kiểm tra, 1: thực hiện kiểm tra



Fsck là một tiện ích được sử dụng để kiểm tra và sửa chữa các lỗi trên hệ thống tệp.

*sudo fsck [tùy chọn] <đường dẫn đến thiết bị hoặc hệ thống tệp>*

Các tùy chọn bao gồm:

* “-a”: tự dộng sửa tất cả các lỗi mà không yêu cầu xác nhận từ người dùng.
* “-r”: sửa tất cả các lỗi có thể tuwjd ộng và đặt hệ thống vào chế độ “read-only” nếu cần

# LVM – Logical Volume Management

## Logical volume management

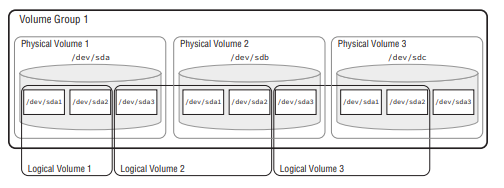
LVM cho phép nhiều partition được nhóm lại với nhau và sử dụng như một phân vùng duy nhất. Việc nhóm này cũng giúp chúng ta dễ dàng tăng hoặc giảm kích thước của phân vùng theo mong muốn.

LVM bao gồm 3 phần chính, mỗi phần đóng vai trò quan trọng trong việc tạo và lưu trữ các logical volume:

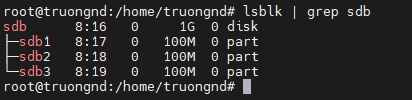
* **Physical Volume**: là các ổ đĩa vật lý hoặc phân vùng trên ổ đĩa vật lý. Mỗi Physical Volume được đăng ký với LVM và cung cấp không gian lưu trữ cho các logical volume
* **Volume Group**: là tập hợp của một hoặc nhiều Physical Volume. Một Volume Group là đơn vị lớn nhất mà người quản trị có thể quản lý trong LVM. Nó cung cấp không gian lưu trữ cho các Logical Volume và quản lý cách dữ liệu được phân phối trên các Physical Volume.
* **Logical Volume**: là phân vùng được tạo ra từ Volume Group. Logical Volume tương tự như các phân vùng trên ổ đĩa thông thường, nhưng chúng có thể được thay đổi kích thức một cách linh hoạt mà không làm mất dữ liệu.

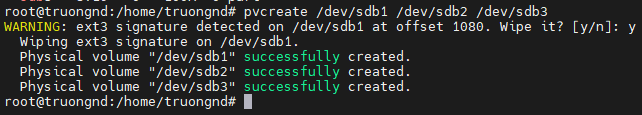
Chúng ta có thể thay đổi kích thước của Logical Volume mà không cần umount.

## Các bước để quản lý và tạo LVM

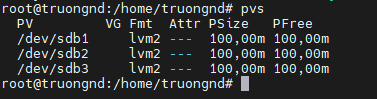


**Tạo Physical Volume**





Liệt kê các PV:

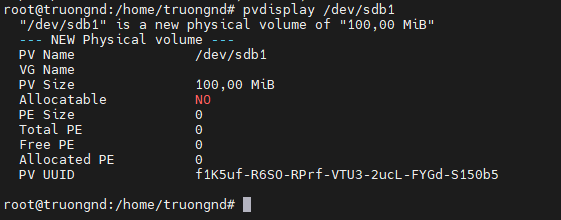


Ý nghĩa các trường của pvs:

PV: Đĩa được sử dụng

PFree: Kích thước đĩa vật lý (Kích thước PV)

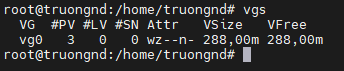
Để có thêm thông tin chi tiết từng PV có thể sử dụng lênh *pvdisplay:*

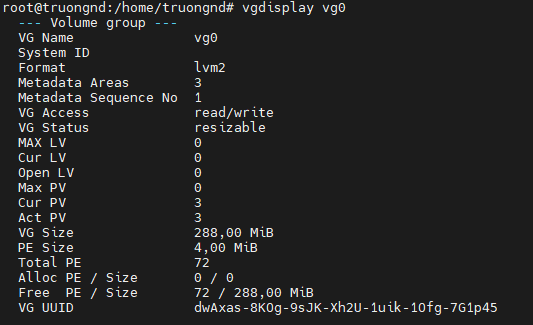


**Tạo Volume Group**

Để tạo volume group sử dụng lênh *vgcreate*:





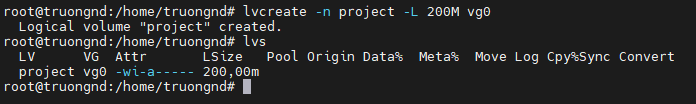


Ý nghĩa các thông tin của Volume group khi chạy lệnh vgdisplay:

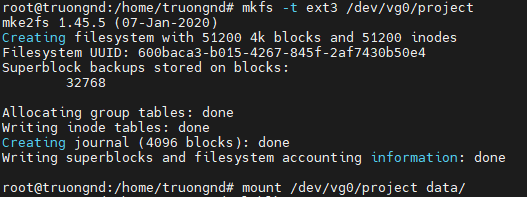
* VG Name: Tên Volume Group.
* Format: Kiến trúc LVM được sử dụng.
* VG Access: Volume Group có thể đọc và viết và sẵn sàng để sử dụng.
* VG Status: Volume Group có thể được định cỡ lại, chúng ta có thể mở rộng thêm nếu cần thêm dung lượng.
* PE Size: Mở rộng Physical, Kích thước cho đĩa có thể được xác định bằng kích thước PE hoặc GB, 4MB là kích thước PE mặc định của LVM
* Total PE: Dung lượng Volume Group có
* Alloc PE: Tổng PE đã sử dụng
* Free PE: Tổng PE chưa được sử dụng

**Tạo Logical Volume**

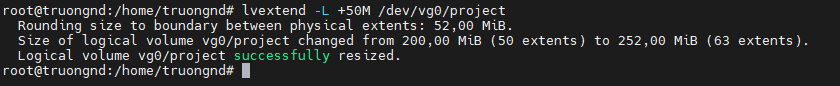
Để tạo logical volume chúng ta sử dụng lệnh *lvcreate:*

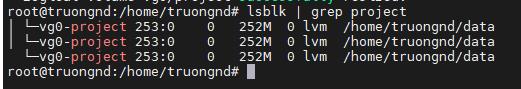


Sau đó chúng ta thực hiện format và mount như một phân vùng bình thường để sử dụng.

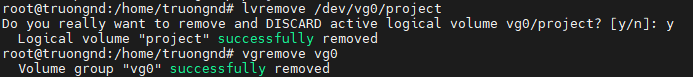


Thực hiện mở rộng phân vùng lvm:





**Xóa các LV đã tạo**

****

