Nguyễn Thế Đức - Bkacad

Quản lý File System Linux

I. Device

Thiết bị trung tâm của máy tính là bộ vi xử lý CPU và bộ nhớ trong RAM, ngoài 2 bộ phận này thì tất cả những bộ phận khác như bàn phím, con chuột, màn hình, ổ đĩa cứng 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. Nếu tháo vỏ case ra quan sát mainboard, sẽ thấy có rất nhiều *"đường kẻ màu xanh (hoặc vàng)"* → đó chính là đường bus

Phân loại device

Các device được phân chia thành 2 loại tùy thuộc vào cách thức đọc/ghi dữ liệu

- ***Block device*** *hay Random Access Device như: HDD, CDROM, USB… các thiết bị dùng để lưu trữ dữ liệu lâu dài (storage device). Dữ liệu trên các device này mỗi lần được đọc ghi theo từng block (1 block at a time, 1 block = 512 bytes to 32KB), chứ không phải theo từng byte. Các khối dữ liệu trên các device này được đánh địa chỉ và có thể truy cập đến một địa chỉ bất kì.*
- ***Character device*** *như: mouse, modem, printer, terminal… Dữ liệu trên các device này được đọc/ghi theo từng character (1 byte at a time). Các device này, chả hạn như mouse làm nhiệm vụ chuyển thao tác di chuột thành dữ liệu về vị trí của con trỏ trên màn hình, luôn có dữ liệu thay đổi liên tục theo thời gian và không được 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 con số này

- Số *major* trỏ tới loại driver tương ứng với thiết bị trong linux kernel
- 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

```
# cat /proc/devices
Character devices:
136 pts
Block devices:
  8 sd
# tty
/dev/pts/0
# 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 đặt sẵn trong nhân Linux, hoặc có thể được cài đặt như một module tách rời của nhân.

Device file --> Driver --> iu khin Device

Device file

Việc truy cập một device nếu phải thông qua driver thì rất phức tạp, đòi hỏi người dùng phải có kiến thức sâu về phần cứng. Trong linux, việc này được đơn giản hóa bằng các "device file". *Device file (cũng còn gọi là special file) là các file nằm trong thư mục **/dev/** của Linux, 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 thể 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 file 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

Qui ước tên device file

Block device hay storage 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

✍ CDROM hay USB chỉ có 1 partition

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

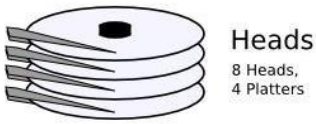
/dev/null: khi có dữ liệu vào, device này sẽ hủy bỏ và cũng không trả về dữ liệu ra

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 kể từ lúc nhân được tải vào RAM.

II. Disk và partition

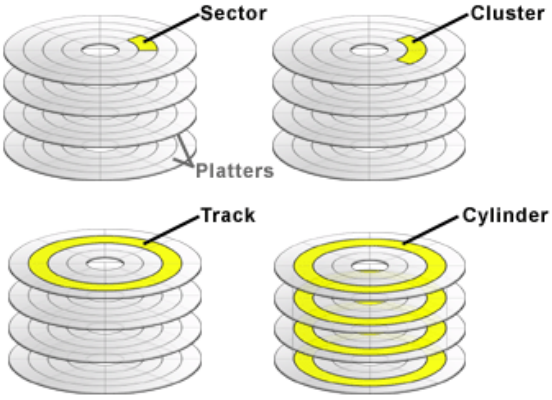
1. Cấu trúc vật lý của ổ đĩa cứng

Một 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 một ổ có 4 đĩa thì có 8 đầu đọc/ghi
- Mạch điều khiển: Là mạch 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 hoá và giải mã các tín hiệu ghi và đọc

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

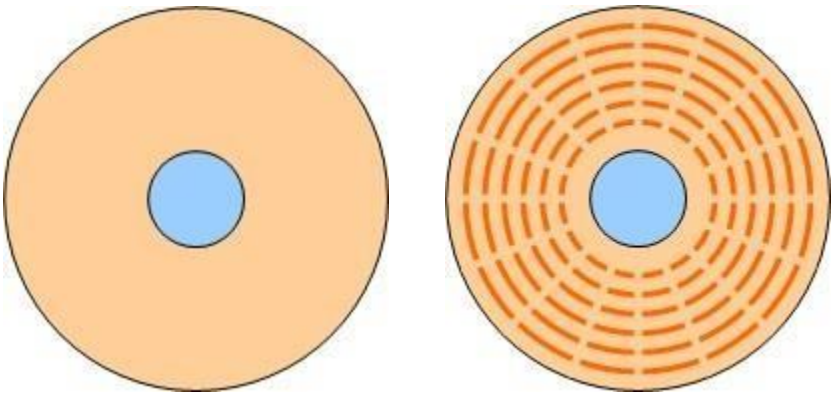
Capacity = 13328 × 15 × 63 × 512 = 6,448,619,520 B

Định dạng vật lý cấp thấp

Đây là công việc của nhà sản xuất ổ đĩa, quá trình được thực hiện như sau :

- *Sử dụng chương trình định dạng để tạo các đường Track*
- *Chia các Track thành các Sector và điền các thông tin bắt đầu và kết thúc cho mỗi Sector*

Format cấp thấp



2. Cấu trúc logic của ổ đĩa cứng với kiến trúc phân vùng MBR

Master boot record (MBR): *là kiểu đặc biệt của 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:

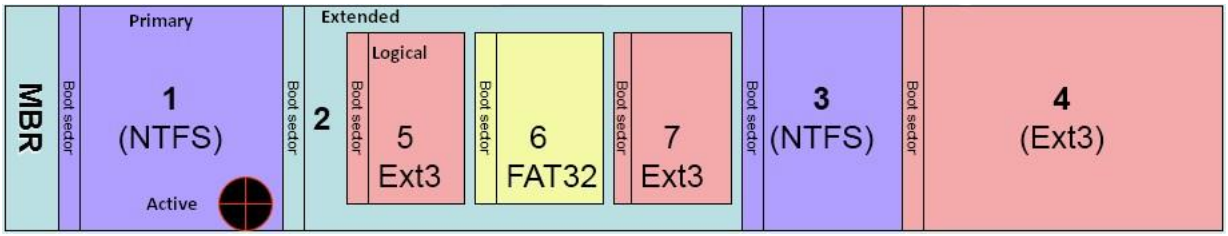
- Đoạn mã Bootloader code nằm ở đầu MBR, có kích thước 446 byte, dùng để chứa chương trình bootrap khởi động Hệ điều hành
- Bảng phân hoạch đĩa (partition table) có kích thước 64 byte, *chứa các thông tin về partition* như số thứ tự, tên ổ đĩa logic, kích thước .v.v. Mỗi MBR có thể quản lý tối đa 4 primary 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

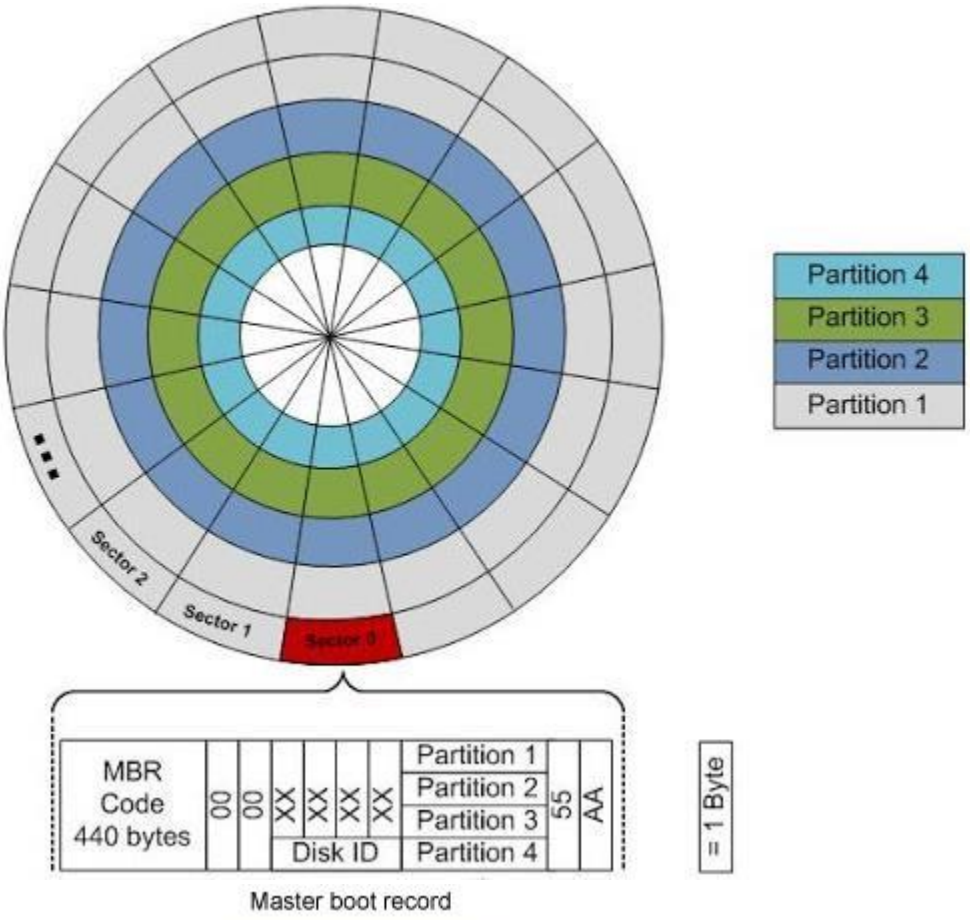
Boot sector: là sector đầu tiên của một partition trong ổ cứng

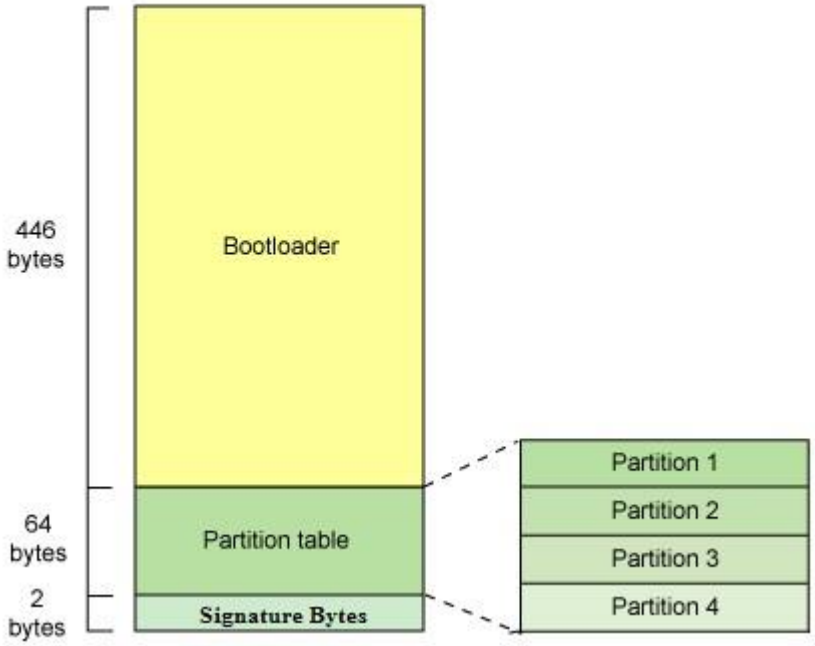
Partition:

- *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.*



Filesystem: FAT, NTFS, EXT, XFS…


Partition 4
Partition 3
Partition 2
Partition 1
Sector 2
Sector 1
Sector 0
MBR Code 440 bytes
00
00
XX
XX
XX
XX
Disk ID
Partition 1
Partition 2
Partition 3
Partition 4
55
AA
= 1 Byte
Master boot record


446 bytes
Bootloader
64 bytes
Partition table
2 bytes
Signature Bytes
Partition 1
Partition 2
Partition 3
Partition 4

Đọc/ghi theo Sector/Cluster

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

Ví dụ: file 800 bytes → cần 2 sectors để lưu

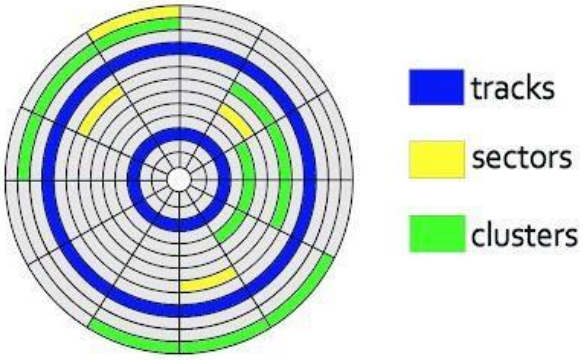
Các Hệ điều hành thường điều khiển việc read/write lên ổ đĩa cứng theo cluster (block) = n sector

Ví dụ: Windows XP/7/8

1 cluster = 8 sectors = 4KB

1 file có size 10 byte → size on disk là 1 cluster = 4KB

→ lãng phí space nhưng tăng performance



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

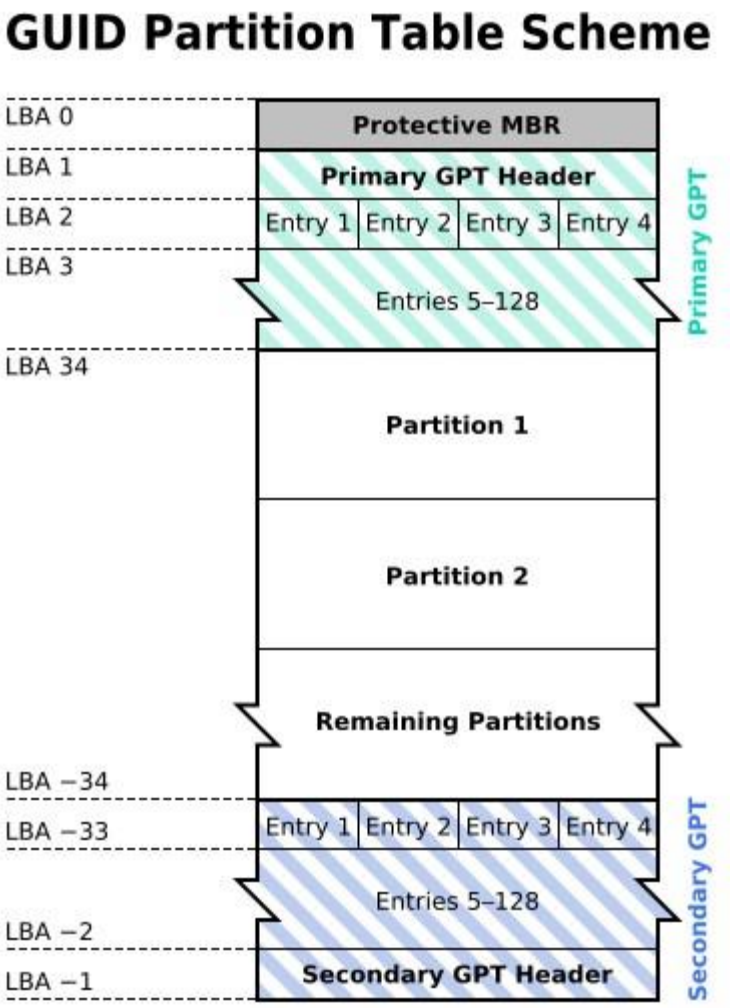
Những hệ điều hành cũ với BIOS legacy và kiểu bảng phân vùng Master Boot Record bị giới hạn bởi kích thước phân vùng nhỏ hơn 2 TB bởi chúng chỉ có thể ghi địa chỉ cho dung lượng đến **2^{32} x 512 byte** (kích thước sector)

UEFI *(Unified Extensible Firmware Interface - Giao diện firmware mở rộng hợp nhất*) là một chuẩn BIOS mới của Intel, có trên các bo mạch chủ đời mới được lưu trữ 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 2 TB

GPT (*GUID Partition Table*) là một phần nằm trong đặc tả EFI do Intel đưa ra, mô hình EFI giúp cho hệ điều hành có thể giao tiếp với firmware hệ thống. GPT có thể xem là một miêu tả về cách sắp xếp các phân vùng trên một ổ đĩa.

	MBR Disk	**GPT disk**
Support in legacy OS like DOS, Windows 98 etc	Yes	No
Support for more than 2TB (Tera Byte)	No	Yes
Support as Data disk in x86 version of OS	Yes	Yes
Support as Data disk in x64 version of OS	Yes	Yes
Support as Boot disk in x86 version of OS	Yes	No
Support as Boot disk in x64 version of OS	Yes	Yes
Support for more than 4 primary partition	No	Yes (Upto 128 Partition)
Booting support through BIOS mode	Yes	No
Booting support through uEFI mode	No	Yes

Hiện nay phần lớn đều sử dụng hệ thống sắp xếp phân vùng theo dạng *Master Boot Record*. MBR chỉ hỗ trợ tối đa 4 phân vùng primary. GPT ra đời với khả năng tùy biến cao hơn như cho phép tạo nhiều hơn các phân vùng primary và tăng kích thước tối đa cho một phân vùng



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 vùng sẽ được gán một GUID (Globally Unique Identifier) 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 ổ đĩa định dạng GPT

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

4. Ký hiệu ổ đĩa, phân vùng trên Linux

Những ổ đĩa IDE sẽ có tên là **hdX**

X có giá trị từ [a-z] đại diện cho một ổ đĩa vật lý: hda, hdb…

Khi chia partition, partition sẽ có dạng: **hdXY**

X là kí tự ổ đĩa.

Y là số thứ tự.

*Các partition primary được đánh số từ **1 » 3**, các partition logical bắt đầu từ **5, 6, 7,…***

CDROM cũng được hiểu như một ổ đĩa IDE.

Ổ đĩa SATA, SCSI, USB sẽ có tên là **sdX**

III. File System

1. Cấu trúc File system

Mỗi một block storage device như partition HDD, CDROM hay USB đều có một cách tổ chức dữ liệu để việc lưu trữ/truy xuất hiệu quả. Cách tổ chức dữ liệu như thế được gọi là File system.

Các File system thường gặp là FAT 32, NTFS, EXT3/4, UFS…

Cách tổ chức dữ liệu đơn giản nhất là FAT, dùng một bảng cấp phát file (File allocation table)

Boot Sector	FAT 1	FAT 2 (Duplicate)	Root Folder	Other Folders and All Files

FAT1 là một cái bảng chứa metainfo (thông tin mô tả) về file và thư mục (tên, đường dẫn, ngày tháng tạo lập, thuộc tính) và địa chỉ của vùng sector lưu nội dung file. Vì FAT1 rất quan trọng nên nó được back up lại thành FAT2.

Linux File system

EXT2 *Hệ thống file system trên các bản phân phối Linux cũ (Linux native)*

EXT3 = **EXT2** + **Journaling** *(Hệ thống file nhật ký → Một hệ thống file có tính năng log. Giúp toàn vẹn dữ liệu và tránh các trường hợp mất dữ liệu đột ngột do mất điện, cháy máy, … ext3 có độ tin cậy cao vì có khả năng ghi nhớ quá trình thao tác trên dữ liệu, khôi phục nhanh hệ thống file khi có sự cố)*

Nâng cấp từ ext2 lên ext3

```
# tune2fs -j /dev/sda1 ext3
```

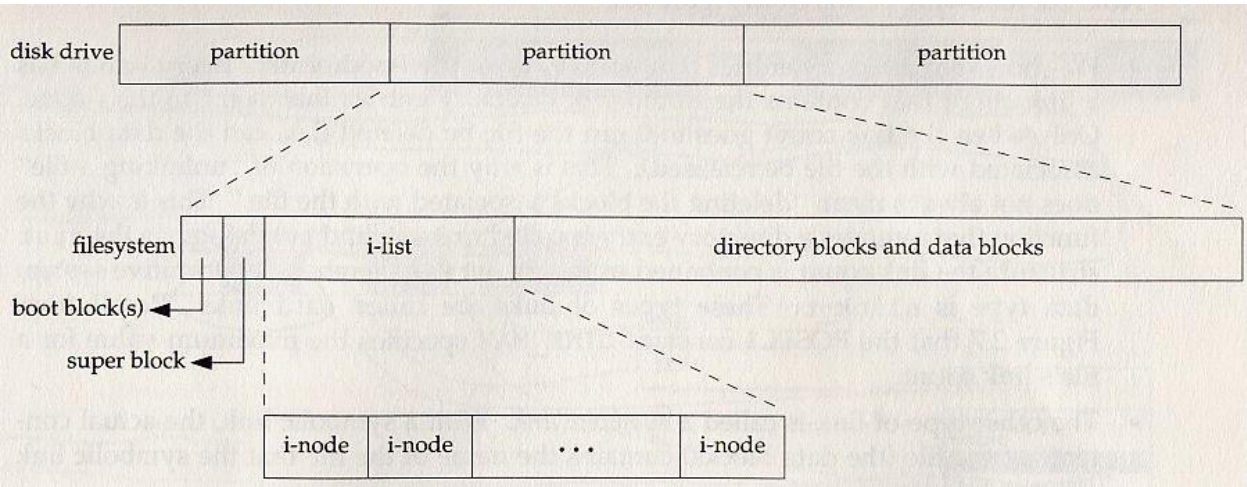
EXT4 *Kế thừa từ ext3, ra đời từ nhân Linux phiên bản 2.6.28 (25/12/2008). Kích thước file tối đa lên đến 16TB, kích thước file system tối đa đến 1 EB (Exabytes)*

Khả năng mở rộng thư mục: ext3 hỗ trợ 32000 thư mục con, ext4 không giới hạn việc tạo các thư mục con

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

Hệ thống file của Linux có các thành phần sau:

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



Super Block: là cấu trúc được tạo tại vị trí bắt đầu filesystem. Lưu trữ các thông tin :

- *Kích thước và cấu trúc filesystem.*
- *Thời gian cập nhật filesystem cuối cùng.*
- *Thông tin trạng thái.*

Inode: lưu những thông tin về tập tin và thư mục được tạo trong filesystem. 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 filesystem.*

```
# tune2fs -l /dev/sda1 | grep Inode
Inode count:              26104
Inodes per group:         2008
Inode blocks per group:   251
Inode size:               128
```

Data block: là vùng lưu dữ liệu thực sự của tập tin và thư mục. Dữ liệu lưu trữ vào đĩa trong các data block. Mỗi block thường chứa 1024 byte. Ngay khi tập tin chỉ có 1 ký tự thì cũng phải cấp phát 1 block để lưu nó. Data Block của tập tin thông thường lưu inode của tập tin và nội dung của tập tin. Data Block của thư mục lưu danh sách những entry bao gồm inode number, tên của tập tin và những thư mục con.

Kiểm tra Filesystem Block Size

```
# tune2fs -l /dev/sda1 | grep -i “block size”
Block size: 4096
```

Giới hạn File system

Filesystem	***CentOS 5***	***CentOS 6***
Maximum filesize (ext3)	2TB	2TB
Maximum file system size (ext3)	16TB	16TB
Maximum filesize (ext4)	16TB	16TB
Maximum file system size (ext4)	1EB	1EB

KiloByte (KB), MegaByte (MB), GigaByte (GB), TeraByte (TB), PetaByte (PB), ExaByte (EB), ZettaByte (ZB), YottaByte (YB)
Linux hỗ trợ nhiều loại Filesystem: ext2, ext3, ext4, vfat, ntfs...với công nghệ được tích hợp vào bên trong kernel. Các Filesystem mới được sử dụng trong linux: ext4, Reiserfs, XFS và JFS.
Windows chỉ hỗ trợ các File system FAT và NTFS

2. Phân hoạch và format cấp cao

Để có thể đọc/ghi file lên USB stick, FDD, HDD thì các device này cần được format theo một file system nào đó mà hệ điều hành có thể hiểu được. USB stick, FDD chỉ có một partition, HDD có thể có nhiều partition nên với HDD có thể cần thêm bước phân hoạch partition trước khi format bằng lệnh fdisk.

```
# fdisk /dev/sdb
# mkfs.vfat /dev/sdb1
# mkfs.ext3 /dev/sdb2
```

Các lệnh format File system

```
# ls -li /sbin/mk*
# find / -xdev -samefile mkdosfs
/sbin/mkfs.msdos
/sbin/mkdosfs
/sbin/mkfs.vfat
```

3. Mount

Cú pháp: **mount <device> <mount_point>**

Khi cắm một device vào, nếu Linux nhận được, nó sẽ tạo ra một file mới trong thư mục **/dev**
Các file *device* như: partition ổ cứng, CD-ROM, usb… cần được mount vào một đường dẫn thư mục trước khi dùng. Thư mục được mount gọi là *mount point.*

Mount USB

Kiểm tra xem tên của USB

```
# fdisk -l
```

Tạo thư mục để mount USB

mkdir /mnt/usb

Mount USB định dạng FAT32

mount /dev/sdb1 /mnt/usb

df -hT

Umount USB

umount /mnt/usb

Mount CD/DVD

Với một số Linux distro thường có sẵn một link /dev/cdrom trỏ đến CDROM device. Có thể truy xuất đến CDROM qua link này thay vì phải xác định CDROM là file gì trong thư mục /dev

ls -l /dev/cdrom

/dev/cdrom → /dev/hdc

mount /dev/cdrom /media

Umount CD/DVD

umount /media

Mount NTFS

Hầu hết các bản Linux hiện nay đều có sẵn driver hỗ trợ NTFS nhưng thường không cài mặc định. Cài đặt package **fuse-ntfs-3g** hỗ trợ phân vùng định dạng NTFS

Mount Loop device

Dùng lệnh mount với tham số **-o loop**

Tạo file ISO image từ một thư mục

mkisofs -r -o httpd.iso /etc/httpd

Mount file ISO image

mkdir /mnt/httpd

mount -o loop httpd.iso /mnt/httpd

df -hT

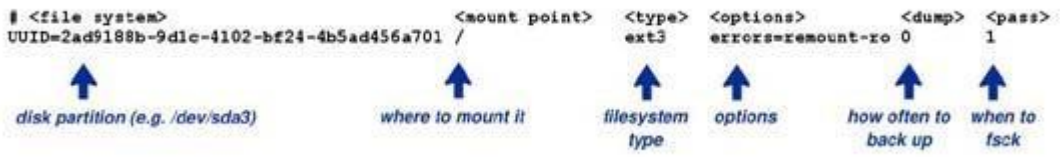
✍ Ngoài việc mount thiết bị, còn có thể mount thư mục với tùy chọn *--bind*

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

File **/etc/fstab** (*file system table*) ghi lại những partition được mount khi khởi động hệ thống. Trong file **/etc/fstab** có thể có những device mount không thành công do ghi không đúng qui cách, không tìm thấy mount point hay đường dẫn device bị thay đổi ... Những partition đã mount được thành công được ghi trong **/etc/mtab** (*mounted table*).

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**

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

```
# blkid
```

- Cột 2: đường dẫn thư mục (mount point) mà device sẽ gắn kết vào

- Cột 3: kiểu file system của device: *ext, swap, vfat, auto* (*hệ thống sẽ tự động nhận diện loại filesystem)*

- Cột 4: các tùy chọn khi mount. Nếu có nhiều tùy chọn thì chúng được phân cách bởi dấu phẩy.

auto: tự động mount thiết bị khi máy tính 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.

nouser: chỉ có người 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

```
# blkid /dev/md0
/dev/md0: UUID="4930918e-e7e3-41b1-ab93-9300ff3c7a85" TYPE="ext4"
# blkid /dev/md1
/dev/md1: UUID="c1a29ffa-903c-4ae9-a892-ba4b4afbbfb9" TYPE="ext4"
# cat /etc/fstab
UUID=c1a29ffa-903c-4ae9-a892-ba4b4afbbfb9 /                ext4    defaults        1 1
UUID=4930918e-e7e3-41b1-ab93-9300ff3c7a85 /boot            ext4    defaults        1 2
UUID=2993f738-d3fd-4fb0-8125-c40a2a888738 /data            ext4    defaults        1 2
UUID=71d901ba-6436-4a78-bf18-b945daf79131 swap             swap    defaults        0 0
```

```
# df -hT
Filesystem    Type    Size  Used Avail Use% Mounted on
/dev/md1      ext4     40G   15G   23G  39% /
tmpfs         tmpfs    48G     0   48G   0% /dev/shm
/dev/md0      ext4    485M   11M  449M   3% /boot
/dev/md3      ext4    870G  168G  658G  21% /data
# cat /proc/mdstat
```

Mount phân vùng /tmp với chế độ noexec

*Khá nhiều đoạn script dựa vào các script download trong thư mục /tmp và thực thi chúng. Bằng cách mount phân vùng **/tmp** với chế độ **noexec**, các script đặt trong thư mục /tmp sẽ không thể thực thi.*

IV. RAID & LVM

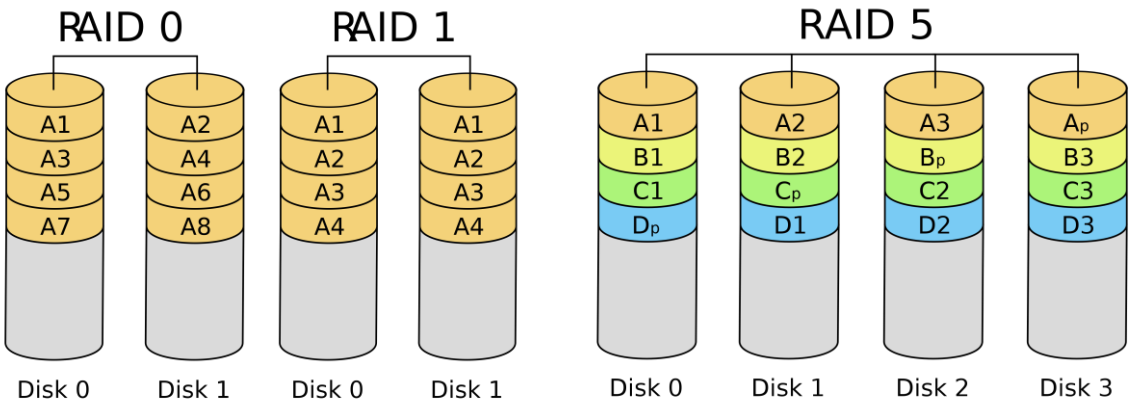
Cách quản lý HDD thông thường là chia nó thành các partition nhỏ hơn → *kích thước partition bị giới hạn bởi kích thước HDD chứa nó*. Có hai kĩ thuật ghép là RAID và LVM cho phép mở rộng kích thước partition lớn hơn kích thước HDD chứa nó.

Driver cho RAID, LVM thường được biên dịch vào kernel.

1. Redundant Array of Inexpensive Disks - RAID

- RAID là viết tắt của Redundant Arrays of Independent Disks (Các dãy đĩa dư thừa độc lập).
- RAID thường chỉ được ứng dụng cho các máy chủ.
- Nguyên lý của RAID là đổi dung lượng lấy sự an toàn dữ liệu hoặc tăng tốc độ đọc ghi.
- Để thiết lập được một hệ thống RAID ta cần phải có ít nhất 2 đĩa cứng trở lên.
- RAID có thể được thực thi bằng phần cứng hoặc bằng phần mềm (RAID controller). RAID controller phần cứng thường là một PCI card hoặc được gắn on-board trên mainboard.
- Cấu hình RAID phần cứng được thực hiện qua BIOS là trong suốt đối với hệ điều hành. Nếu không có RAID controller phần cứng thì có thể dùng phần mềm thông qua hệ điều hành. Với Linux có 2 phần mềm RAID là: **raidtools** (tập các công cụ do nhiều người cùng phát triển) và **mdadm** (của Neil Brown).

Các loại RAID chính



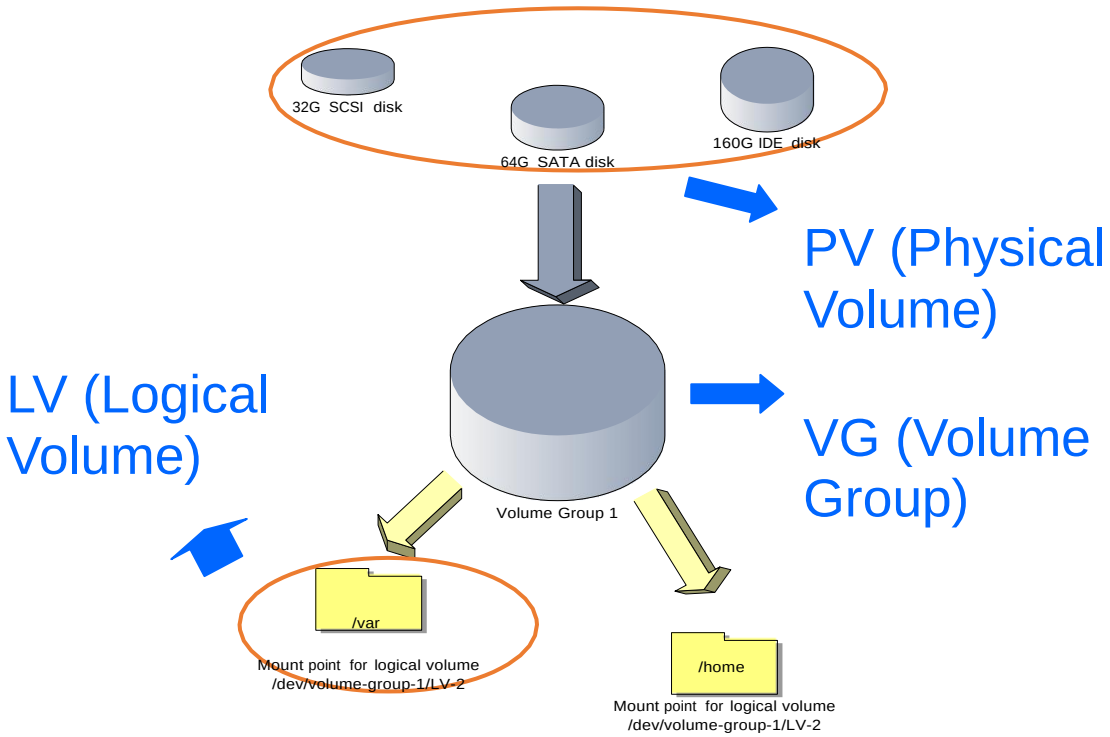
RAID 0 (striping): 2 hay nhiều disk cùng dung lượng được RAID lại thành một array. Khi ghi dữ liệu được chia làm nhiều phần, mỗi phần được ghi lên một disk. Điều này cải thiện hiệu suất ghi vì việc ghi dữ liệu được thực hiện đồng thời trên cả 2 disk. Khi một disk bị hỏng thì một phần của dữ liệu sẽ mất do đó cả dữ liệu sẽ bị hỏng. RAID 0 không có khả năng khôi phục dữ liệu.
RAID 1 (mirroring): 2 disk cùng dung lượng được RAID lại thành một array. Khi cần ghi dữ liệu, sẽ có 2 bản sao của dữ liệu, mỗi bản được ghi vào một disk. Thời gian ghi dữ liệu trên array bằng thời gian ghi dữ liệu trên một disk, do đó hiệu suất ghi không đổi. Bởi mỗi disk giữ một bản sao của dữ liệu nên sẽ lãng phí một nửa dung lượng lưu trữ nhưng khi hỏng một disk thì vẫn còn disk kia lưu dữ liệu.
RAID 5 (striping with parity): cần ít nhất 3 disk cùng dung lượng để RAID lại thành một array. Lấy ví dụ 4 disk xem hình vẽ. Khi file A cần ghi vào array, nó được chia làm 3 phần A1, A2, A3; phần Ap, còn được gọi là *checksum*, được sinh ra từ A1, A2, A3 theo một thuật toán nào đó. Mỗi phần trong 4 phần này được ghi vào một disk. Nếu một disk hỏng, chả hạn disk 0 (mất A1) thì từ A2, A3, Ap có thể tính lại được A1. Tức là khi hỏng một disk bất kì trong 4 disk thì dữ liệu có thể khôi phục được nhờ vào checksum và các phần còn lại không hỏng.

2. Logical Volume Management v2

Ổ cứng dạng MBR bị giới hạn bởi 4 phân vùng primary

Giải pháp cho vấn đề phân vùng

Linux giải quyết được vấn đề này bằng cách sử dụng *Logical Volume Manager* (LVM), cho phép tổ chức ổ cứng thành các khối Logical Volumes (LV) thay vì sử dụng phân vùng như trước kia.



- Linh hoạt trong việc phân chia partition.
- Dễ dàng mở rộng kích thước của volume.
- Để mở rộng dung lượng lưu trữ dữ liệu, đơn giản chỉ cần thêm đĩa mới vào.

Linux LVM là phương pháp cho phép ấn định không gian đĩa cứng thành những *Logical Volume* giúp cho việc thay đổi kích thước trở nên dễ dàng hơn so với kỹ thuật partition. Với kỹ thuật LVM ta có thể thay đổi kích thước mà không cần thay đổi bảng partition.

LVM là một trình quản lý storage device theo 3 lớp từ thấp đến cao

- Physical Volume (PV): là những *đĩa cứng vật lý* HDD /dev/sda, /dev/sdb hoặc *phân vùng đĩa* /dev/sdc1, /dev/sdd1
- Volume Group (VG): nhiều PV khi được ghép lại với nhau tạo thành một Volume Group. Dung lượng của VG bằng tổng dung lượng của các PV thành phần. Các PV thành phần có thể thuộc cùng một HDD hoặc thuộc nhiều HDD khác nhau. Có thể thêm vào hoặc loại đi một PV từ VG.
- Logical Volume (LV): mỗi VG lại được chia thành một hoặc nhiều Logical Volume (Virtual partition). Có thể resize các LV một cách dễ dàng và snapshot chúng. Có 3 kiểu mapping volume: *linear* volumes, *striped* volumes, và *mirrored* volumes.
- Số PV và LV tối đa cho phép trên một VG là 256

✍ *phân vùng /boot không thể là một Logical Volume, vì trình nạp khởi động (boot loader) sẽ không nhận ra được. Do đó phải tạo một phân vùng /boot độc lập, không nằm trong LVM*

Bảng lệnh LVM

	CREATE	VIEW	REMOVE	RESIZE	CHANGE	*SCAN*
PV	pvcreate	pvs, pvdisplay	pvremove		pvchange	***PVSCAN***
VG	vgcreate	vgs, vgdislpay	vgremove	vgextend, vgreduce	vgchange	***VGSCAN***
LV	**LVCREATE**	**LVS, LVDISPLAY**	**LVREMOVE**	**LVEXTEND, LVREDUCE**	**LVCHANGE**	***LVSCAN***

- pvcreate: khởi tạo những physical volume để sử dụng trong môi trường LVM. Physical volume có thể là đĩa cứng, thiết bị lưu trữ khác, hoặc partition…
- pvs, pvdisplay: hiển thị thông tin của physical volume.

- vgcreate: khởi tạo một volume group từ những physical devices đã được khởi tạo bằng pvcreate.
- vgextend: thêm physical volume vào volume group.
- vgs, vgdisplay: xem không tin của volume group

- lvcreate: tạo logical volume từ volume group.
- lvdisplay: xem thông tin của logical volume.

Cài đặt system-config-lvm quản lý LVM trên giao diện đồ họa

```
# yum install system-config-lvm -y
# system-config-lvm
```