Partitions-Mass-Storage-Definitions-Naming-HOWTO

Jean-Daniel Dodin

<jdd@dodin.org>

Revision History

Revision V0.1	2009-05-09	Revised by: jdd

Table of Contents

1.

Partitions-Mass-Storage-Definitions-Naming-HOWTO, copyright (c) 2009 Jean-Daniel Dodin

2. Copyright and Licence

The copyright of this document is to the author, Jean-Daniel Dodin, according to the following licence.

GNU Free Documentation License

---

3. Mass Storage Involved Here

Mass storage involved in the present HOWTO are rewritable random access ones. Most of them are magnetic rotating disks (floppies, Hard Drive) or flash memory (USB key or any kind of memory card).

For example, cdroms and dvds are **not**concerned by this HOWTO (see Wikipedia). Tapes are not either.

Mass storage are used by the kernel, so the basic doc can be found on the kernel Web site

Reference site should be the International Disk drive Equipment and Materials Association. *Should*because this Web site is not very friendly.

4. Definitions

4.1. Warning

Many definitions about drives are only virtual. That is they are used, but the hardware is often quite different from the expected description. Usually this have no odd result, any mass storage have to be seen as a *black box*.

4.2. Bytes

Computers counts with binaries, 1 & 0, 1111100001110... To be able to read this better, humans uses nibbles (4 bits) often shown as Hexadecimal numbers from 0 to f (0123456789abcdef). Nibbles are usually grouped by two and this gives a byte. The most used memory unit is byte and it's multiples, KiB (Kilo Bytes), MiB (Mega Bytes), GiB (Gigabytes). The "i" denotes the binary use (0ne Ki is 1024, not 1000), the uppercase "B" denotes Bytes, not bits.

4.3. Sectors

Sometime, the word *block*is used in place of *sectors*.

Mass storage devices (at least the ones we are dealing with here) store bytes in "Sectors" of 512 Bytes. This is uneven, because any sector count have to be divided by two to have the KiB number, so most partitonning software accepts letters k (KiB), m (MiB), g... as options. Wise ones do not make any case difference.

Sector size is the available byte count. The true sector is bigger, as it have to include housekeeping data. You don't have to worry about that.

Notice that as of 03-22-2006, the IDEMA annouced a new sector size of 4kiB (4096 Bytes): http://www.idema.org/_smartsite/modules/local/data_file/show_file.php?cmd=download&data_file_id=1446- doc file, can be openned with OpenOffice.org.

4.4. Heads

Rotating mass storage devices uses *heads*. True heads are the physical electromechanical device that writes and read the magnetic track. Drives being made of rotating plates, the plates have two sides, so disks can have two head by plate. Having two plates (frequent) you have four heads.

Heads are writing through very complex system, see detailed info here: http://www.spmtips.com/library/data_storage.

4.5. Tracks

Plates are rotating. When a head is still, the plate rotation and the width of the head are defining a track.

Heads are moving from the external part of the plate to the inner part, step by steps. Each step defines a new track.

4.6. Cylinders

Heads are moving together, all at the same time. They may rotate - on they own center, not the plate center, of course. They also may have a linear move. You can see an example of linear moving head in any cdreader, looking at the move of the laser head. Most disks are as shown by this wikipedia image http://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/Hard_disk_platters_and_head.jpg/200px-Hard_disk_platters_and_head.jpg.

When you think of all the tracks defined by each head at the same time, you have a cylinder. So on a rotating drive, all the tracks of the same cylinder are read or written at the same time. The actual data is spread on all the plates. The way the data is actually written is up to the drive manufacturer, not the user.

4.7. Disks

Small disks are used directly as a hole bunch of sectors. Basic programs can access data directly on sectors. Many do (like dd or any partitionning programm).

But we live in a world of extremely high capacity mass storage. Terabytes is normal nowaday (2009), when a complete Linux system can live on a floppy (1440 bytes). So there is a need of making several parts from a mass storage device, though the partitions.

4.8. Partitions

Partitioning is a means to divide a single drive into many logical drives. A partition is a contiguous set of sectors. To lessen the heads travel, partitions can be "aligned" on the cylinder size, that is use an integer number of cylinder. This is not always done, but should as it have many other advantages for recovery.

4.9. Partition Table

As you can have many partitions, you need to have a partition table. This partition table is stored in the very beginning of the drive. It's very unlikely that you will have to change this table directly writing bytes with an hexadecimal editor, so we wont say more on the position of the table.

There are many Operating Systems all around that all share similar hardware and as many partition systems. We will look only at what one can find in a PC, even if it's not easy to define that nowaday. Say, for us, a PC is any computer able to run Linux (I know, it's not always true).

Each of these partition kinds are noted in the table by a special flag called "type" ("t" in fdisk). Most known are type 83 for Linux partitions and 82 for Linux swap (hex numbers).

Notice that most Operating Systems can share partition tables. At least, if a disk is hardware compatible with several systems, these systems should be able to see what the others have done, not to erase a drive by accident. I can't say for sure that its true in the real life.

4.10. File Systems

Partitions can be accessed directly as sectors, as any part of the disk, but are usually filled with a **file system**. File system and partitions are related only because a file system is in a partition, but that's all. You can have a disk without partition but with a file system or have partitions without file system (the swap partition beeing the most well known). For details on file systems, see Wikipedia.

In summary, file systems allow storing data in files with human readable names and to sort the files in a friendly way, for example as directories, subdirectories, text, images...

4.11. Files and Nodes

Nearly all what you can find on a mass storage partition, beside sectors, from an user point of view, is a file. But computers are curious geeks and you can treat files like disks if you want. Using the "loop" system, default in most Linux kernels, one can partition the inside of the file, create file systems on it and mount it. This is specially handy for experiments.

Some of these files are *devices*or *nodes*. Partitions are not files and are accessed via special nodes we will see later. These nodes are not created by touch but by *mknode*. Use with caution. Nodes need a type (*c*for "character" or "b" for *block*) and major and minor numbers. For what we need, major numbers are disk numbers and minor numbers are partition numbers. The list is visible in /proc/partitions

```
        cat /proc/partitions
major minor  #blocks  name

   8     0  488386584 sda
   8     1   52436128 sda1
   8     2          1 sda2
   8     5    2104483 sda5
   8     6   20972826 sda6
   8     7   52436128 sda7
   8     8  360434308 sda8
        #mknod b 8 9 /dev/sda9
```

Creates a /dev/sda9 node of no nuse, given this don't create partition, only the node. In a usual Linux distribution, nodes are dynamically created at boot time, so nobody should have to do so. However, sometime the automatic system fails.

5. Drive Naming in Linux

There is a special nomenclature that linux uses to refer to mass storage that must be understood.

5.1. Naming Convention

Linux used to deal with two kind of drives, depending of the electronic interface (controller), IDE and SCSI. Oldtimers remember the day where cdwriters where acccessed through "SCSI emulation". In fact IDE and SCSI use mostly the same low level commands and for 2007 up, with the new "SATA" interface, the naming was unified and, in new ditributions, all the drives have the same naming. For this part, CD or DVD readers/writers are seen like Hard Drives.

5.1.1. Old IDE Names

By convention, IDE drives where given device names /dev/hdato /dev/hdd. *H*ard *D*rive *A*(/dev/hda) is the first drive and *H*ard *D*rive *C*(/dev/hdc) is the third.

A typical PC has two IDE controllers, each of which can have two drives connected to it. For example, /dev/hdais the first drive (master) on the first IDE controller and /dev/hddis the second (slave) drive on the second controller (the fourth IDE drive in the computer).

So, typically, a computer with IDE controller can accomodate 4 drives: /dev/hda (primary master), /dev/hdb (primary slave), /dev/hdc (secondary master), /dev/hdd (secondary slave). Some (rare) Mother Boards have more than two controllers, some addition cards can also have controllers, these are numbered following the alphabet, but one have to figure out what real names are given for his particular hardware.

You can have drives where ever you want, it's not mandatory to fill the gaps. You may have interest to read about what drive/cdrom connect to what place, but it's out of this document scope.

5.1.2. New Hard Drives Names

Now all the rotating hard drives uses the same names as the old SCSI controllers, that is "s" in place of "h", so /dev/sda, and so on. The number of drives depends on the number of controllers on the Mother Board or the extended boards. Usually 4 are available. What will be the number of a drive is up to the controller card and the way it's read by the kernel, so difficult to say at first.

5.1.3. Flash Drives Names

Flash drives are usually not connected through IDE or SATA interfaces and so don't uses the same names. Several interfaces are used with each different names. The kernel documentations gives the names.

5.1.4. Low level Devices and Extra naming

You will find in some apps references to lowlevel SCSI devices and various naming conventions, for example (wodim is the command line cd burner):

```
          wodim --scanbus
scsibus1:
```

```
        1,0,0   100) *
        1,1,0   101) 'TSSTcorp' 'CD/DVDW TS-L632D' 'ac00' Removable CD-ROM
        1,2,0   102) *
        1,3,0   103) *
        1,4,0   104) *
        1,5,0   105) *
        1,6,0   106) *
        1,7,0   107) *
```

And you may have to use some sort of *SCSI:1,1,0*option to access the CDROM. try to avoid using this as much as possible, as it's very error prone and should be let to programmers only. I only mention it because you can't always avoid it.

If you do "cat /dev/ | more", you can see:

```
          lrwxrwxrwx  1 root root             3 mars  9 07:56 scd0 -> sr0
(...)
crw-r-----  1 root disk       21,   0 mars  9 07:56 sg0
crw-rw----+ 1 root disk       21,   1 mars  9 07:56 sg1
```

These scd, sr, sg devices are lowlevel interface (notice the "c" for "character"). Try not using them. *dmesg*and *more /var/log/boot.msg*should give you the usable sdxx device, like (short summary):

```
          <5>sd 0:0:0:0: [sda] 976773168 512-byte hardware sectors: (500GB/465GiB)
<5>sd 0:0:0:0: [sda] Write Protect is off
<7>sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
```

This mean the drive is */dev/sda*.

However these files (given by *dmesg*and *more /var/log/boot.msg*) used to be easy to read but are no more. Now the kernel starts in parallel several drivers, so the messages are mixed, you can have

```
          <6> sda:<6>USB Universal Host Contr'ller Interface driver v3.0
```

This don't mean that your sda drive is an usb one, but the usb module was started at the same time as the drive one and send it's messages simultaneously. You still have a */dev/sda*drive.

5.1.5. New Media Names

Here the dmesg content for inserting an USB key:

```
          scsi7 : SCSI emulation for USB Mass Storage devices
usb 5-3: New USB device found, idVendor=0951, idProduct=160e
usb 5-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 5-3: Product: DataTraveler 2.0
usb 5-3: Manufacturer: Kingston
usb 5-3: SerialNumber: 200706200000000059188185
usb-storage: device found at 9
usb-storage: waiting for device to settle before scanning
scsi 7:0:0:0: Direct-Access     Kingston DataTraveler 2.0 1.00 PQ: 0 ANSI: 2
sd 7:0:0:0: [sdb] 3930112 512-byte hardware sectors: (2.01GB/1.87GiB)
sd 7:0:0:0: [sdb] Write Protect is off
sd 7:0:0:0: [sdb] Mode Sense: 23 00 00 00
sd 7:0:0:0: [sdb] Assuming drive cache: write through
sd 7:0:0:0: [sdb] 3930112 512-byte hardware sectors: (2.01GB/1.87GiB)
sd 7:0:0:0: [sdb] Write Protect is off
sd 7:0:0:0: [sdb] Mode Sense: 23 00 00 00
sd 7:0:0:0: [sdb] Assuming drive cache: write through
 sdb: sdb1
sd 7:0:0:0: [sdb] Attached SCSI removable disk
```

```
sd 7:0:0:0: Attached scsi generic sg2 type 0
usb-storage: device scan complete
```

You see there all what we where speaking about right now! SCSI emulation, scsi, sd and sg names, but also the sdb that is most important for us.

Here are the messages for a high speed SDHC card:

```
          tifm_core: MMC/SD card detected in socket 0:1
mmc1: new SDHC card at address d555
mmcblk0: mmc1:d555 SD04G 3.79GiB
 mmcblk0: p1

/dev/mmcblk0p1 on /media/H2SD type vfat (rw,nosuid,nodev,noatime,flush,uid=1000,utf8,shortname=lower)
```

When the two cards are probably the same flash memory chip, the USB key uses the USB interface and SCSI emulation, the SDHC card uses the PCMCIA slot of the laptop, with a special device naming (/dev/mmcblk0). The use, as far as partitionning is involved is the same.

5.1.6. Disk ID

In a world where disks are many and removable, it's impossible to track what device is used by what disk. So there are now many way of using a disk name. This makes it extremely difficult to work with basic tools. These are "Disk labels" and "Disk UUID", also "Partition Labels". See fstab man page for details.

6. Partition Naming in Linux

6.1. Numbers

Partition naming is thanksfully simpler than drive one. Partitions are simply given a number from 0 up (decimal). Sometime a "p" is appended on front of the number:

```
        cat /proc/partitions
major minor  #blocks  name

   8     0  488386584 sda
   8     1   52436128 sda1
   8     2          1 sda2
   8     5    2104483 sda5
   8     6   20972826 sda6
   8     7   52436128 sda7
   8     8  360434308 sda8
 179     0    3979776 mmcblk0
 179     1    3975680 mmcblk0p1
```

As you see, partition devices are listed in /proc/partition. This file... is not a real file but is created on the fly. Don't worry, for what we need it's a file.

Notice the "p1" partition name for the SDHC card.

Max number of partitions is 15 for SCSI and all the drives using the new SATA driver, 63 for IDE drives (0 is the full drive, 0 to 15 is four bits 0 to 64, 6 bits)

6.2. Meaning of the Numbers

Not all the numbers have the same meaning. This mess come from the PC history. One can divide floppies with partitions, but then 4 ones seems sufficient. But then come Hard drives :-). So the partitons numbers 1, 2, 3 and 4 are *primary*partitions. One drive can only have 4 primaries.

To go further, we have to use one of these primary as a big one and sub-partition this one, so to have *logical*partitions. The big *extended*partition can be any of the 4.

So, remember, the primary partitions are inside the drive and the logical partitions are inside one of the primary, called the *extended*partition.

Once the logical partitions are created, it's no more recommended to write directly to the extended one. Writing to an extended partition would erase the logical ones like writing directly to a hard drive erase the partitons. Beware, **it's possible**!!

If, after creating 4 primary partitions, all the disk space is not used, the remaining space is lost (unusable), so most of the time, create the desired primaries, then at last the extended one with all the remaining room.

It's not necessary to create 4 primaries. You could use only one extended (Linux only), but there are some advantages of using primaries.

Primaries being 4, the first logical partition is always 5. So any partition with number of five and up is a logical one.

7. Device Major and Minor Numbers

The only important thing with a device file are its major and minor device numbers, which are shown instead of the file size:

```
      $ ls -l /dev/hda
brw-rw---- 1 root disk 8, 0 mars  9 07:56 /dev/sda
```

Shows permissions (brw-rw----), owner (root), group (disk), major device number (8), minor device number (0), date (mars 9 - french, no year), hour (07:56) and device name (guess :-).

When accessing a device file, the major number selects which device driver is being called to perform the input/output operation. This call is being done with the minor number as a parameter and it is entirely up to the driver how the minor number is being interpreted. The driver documentation usually describes how the driver uses minor numbers.

8. Partition Types

8.1. Linux Partition Types

A partition is labeled to host a certain kind of file system (not to be confused with a volume label. Such a file system could be the linux standard ext3 file system or linux swap space, or even foreign file systems like (Microsoft) NTFS or (Sun) UFS. There is a numerical code associated with each partition type. For example, the code for ext2 is 0x83and linux swap is 0x82(0x mean hexadecimal).

8.2. Foreign Partition Types

The partition type codes have been arbitrarily chosen (you can't figure out what they should be) and they are particular to a given operating system. Therefore, it is theoretically possible that if you use two operating systems with the same hard drive, the same code might be used to designate two different partition types. OS/2 marks its partitions with a 0x07 type and so does Windows NT's NTFS. MS-DOS allocates several type codes for its various flavors of FAT file systems: 0x01, 0x04 and 0x06 are known. DR-DOS used 0x81 to indicate protected FAT partitions, creating a type clash with Linux/Minix at that time, but neither Linux/Minix nor DR-DOS are widely used any more.

8.3. Swap Partitions

Every process running on your computer is allocated a number of blocks of RAM. These blocks are called pages. The set of in-memory pages which will be referenced by the processor in the very near future is called a "working set." Linux tries to predict these memory accesses (assuming that recently used pages will be used again in the near future) and keeps these pages in RAM if possible.

If you have too many processes running on a machine, the kernel will try to free up RAM by writing pages to disk. This is what swap space is for. It effectively increases the amount of memory you have available. However, disk I/O is about a hundred times slower than reading from and writing to RAM. Consider this emergency memory and not extra memory.

If memory becomes so scarce that the kernel pages out from the working set of one process in order to page in for another, the machine is said to be thrashing. Some readers might have inadvertently experienced this: the hard drive is grinding away like crazy, but the computer is slow to the point of being unusable. Swap space is something you need to have, but it is no substitute for sufficient RAM.

8.4. Complete List

From the fdisk help:

```
        0  Vide           1e  Hidden W95 FAT1 80  Old Minix       bf  Solaris
1  FAT12           24  NEC DOS         81  Minix / old Lin c1  DRDOS/sec (FAT-
2  XENIX root      39  Plan 9          82  Linux swap / So c4  DRDOS/sec (FAT-
3  XENIX usr       3c  PartitionMagic  83  Linux           c6  DRDOS/sec (FAT-
4  FAT16 <32M      40  Venix 80286     84  OS/2 hidden C:  c7  Syrinx
5  Extended        41  PPC PReP Boot   85  Linux extended  da  Non-FS data
6  FAT16           42  SFS             86  NTFS volume set db  CP/M / CTOS / .
7  HPFS/NTFS       4d  QNX4.x          87  NTFS volume set de  Dell Utility
8  AIX             4e  QNX4.x 2nd part 88  Linux plein tex df  BootIt
```

```
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba          e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            9f  BSD/OS          eb  BeOS fs
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  GPT
 f  W95 Etendu (LBA 54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
10  OPUS            55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
12  Compaq diagnost 5c  Priam Edisk     a8  UFS Darwin      f4  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f2  DOS secondary
16  Hidden FAT16    63  GNU HURD or Sys ab  Amorce Darwin   fb  VMware VMFS
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Amorce Solaris  ff  BBT
```

9. How Many Partitions

The exact number of partitions allowed on a drive is fixed by the kernel. So you can find the exact number is the kernel documentation, the last version is maintained here http://kernel.org/pub/linux/docs/device-list/If you have the kernel source installed, you can find your version on your computer at /usr/src/linux/Documentation/devices.txt.

Look at "limit on partition". Find yours. Common SATA number is 31, SCSI is 15, some are less.