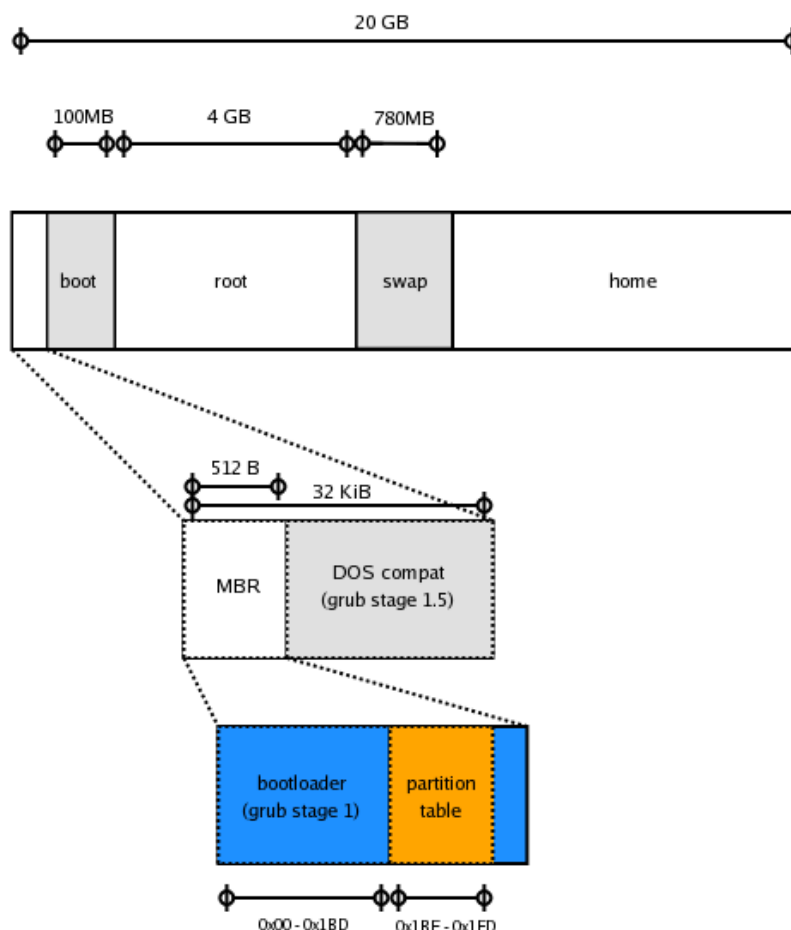


Abstract

This description details the structure at the start of a hard disk in a PC, and how GRUB fits into that structure.

The PC hard disk layout

The [diagram](#) below is the disk layout of my linux laptop which is very standard. There are 4 primary partitions whose size and function are labeled, but this document is going to concentrate on the expanded area in the diagram at the start of the disk. I will describe its structure and how GRUB fits in. Note the structure is the same for windows systems, only the content (grub) is different.



The Master Boot Record

The first sector (512 bytes) is the MBR and consists of, 446 bytes bootloader, 64 bytes partition table and a 2 byte signature (0xAA55). It gets its name because it is the first boot record that is loaded, which can then load other boot records from various locations on the disk. To give a hexdump of this first sector of the disk you can do `dd bs=512 count=1 if=/dev/hda | od -Ax -tx1z -v` (changing /dev/hda as appropriate for your setup):

```
000000 eb 48 90 d0 bc 00 7c fb 50 07 50 1f fc be 1b 7c >.H....|.P.P....|<
000010 bf 1b 06 50 57 b9 e5 01 f3 a4 cb be be 07 b1 04 >...PW.....<
000020 38 2c 7c 09 75 15 83 c6 10 e2 f5 cd 18 8b 14 8b >8,|.u.....<
000030 ee 83 c6 10 49 74 16 38 2c 74 f6 be 10 07 03 02 >....It.8,t.....<
000040 80 00 00 80 b8 85 64 00 00 08 fa 80 ca 80 ea 53 >.....d.....S<
000050 7c 00 00 31 c0 8e d8 8e d0 bc 00 20 fb a0 40 7c >|..1..... ..@|<
000060 3c ff 74 02 88 c2 52 be 79 7d e8 34 01 f6 c2 80 ><.t...R.y}.4....<
000070 74 54 b4 41 bb aa 55 cd 13 5a 52 72 49 81 fb 55 >tT.A..U..ZrI..U<
000080 aa 75 43 a0 41 7c 84 c0 75 05 83 e1 01 74 37 66 >.uC.A|..u....t7f<
000090 8b 4c 10 be 05 7c c6 44 ff 01 66 8b 1e 44 7c c7 >.L...|.D..f..D|.<
0000a0 0a 10 00 c7 11 02 01 00 66 80 5c 08 c7 11 06 00 >...n...f...n...<
```

```

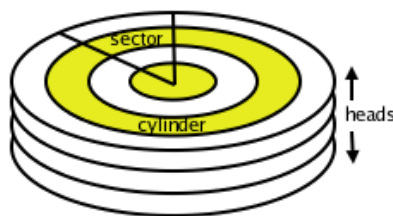
0000a0 04 10 00 c7 44 02 01 00 00 05 3c 00 c7 44 00 00 <....D...f...D...>
0000b0 70 66 31 c0 89 44 04 66 89 44 0c b4 42 cd 13 72 >pf1..D.f.D..B..r<
0000c0 05 bb 00 70 eb 7d b4 08 cd 13 73 0a f6 c2 80 0f >...p.}....s.....<
0000d0 84 f0 00 e9 8d 00 be 05 7c c6 44 ff 00 66 31 c0 >.....|.D..f1.<
0000e0 88 f0 40 66 89 44 04 31 d2 88 ca c1 e2 02 88 e8 >..@f.D.1.....<
0000f0 88 f4 40 89 44 08 31 c0 88 d0 c0 e8 02 66 89 04 >..@.D.1.....f..<
000100 66 a1 44 7c 66 31 d2 66 f7 34 88 54 0a 66 31 d2 >f.D|f1.f.4.T.f1.<
000110 66 f7 74 04 88 54 0b 89 44 0c 3b 44 08 7d 3c 8a >f.t..T..D.;D.}<.<
000120 54 0d c0 e2 06 8a 4c 0a fe c1 08 d1 8a 6c 0c 5a >T....L.....l.Z<
000130 8a 74 0b bb 00 70 8e c3 31 db b8 01 02 cd 13 72 >.t...p..1.....r<
000140 2a 8c c3 8e 06 48 7c 60 1e b9 00 01 8e db 31 f6 >*....H|'.....1.<
000150 31 ff fc f3 a5 1f 61 ff 26 42 7c be 7f 7d e8 40 >1.....a.&B|..}.@<
000160 00 eb 0e be 84 7d e8 38 00 eb 06 be 8e 7d e8 30 >.....}.8.....}.0<
000170 00 be 93 7d e8 2a 00 eb fe 47 52 55 42 20 00 47 >...}.*...GRUB .G<
000180 65 6f 6d 00 48 61 72 64 20 44 69 73 6b 00 52 65 >eom.Hard Disk.Re<
000190 61 64 00 20 45 72 72 6f 72 00 bb 01 00 b4 0e cd >ad. Error.....<
0001a0 10 ac 3c 00 75 f4 c3 00 00 00 00 00 00 00 00 >..<.u.....<
0001b0 00 00 00 00 00 00 00 00 5f 00 5f 00 00 00 00 01 >....._..<
0001c0 01 00 0b fe 7f 97 3f 00 00 00 59 03 64 00 00 00 >.....?...Y.d...<
0001d0 41 98 83 fe 7f a4 98 03 64 00 cd 2f 03 00 00 00 >A.....d../....<
0001e0 41 a5 83 fe ff ff 65 33 67 00 fc 08 fa 00 80 fe >A.....e3g.....<
0001f0 ff ff 0f fe ff ff 61 3c 61 01 1f ed f2 00 55 aa >.....a<a.....U.<

```

Note there is no mystery about or disassembly required of the blue section above as GRUB is open source and one can download, inspect and modify its stage 1 source code.

The DOS compatibility region

This region is optional as far as linux is concerned at least, but is added by default by most partition managers. To understand why this region was required we need to describe how disks used to be addressed. Generally now disks are addressed in LBA mode which allows for greater capacity disks while abstracting software away from the specifics of the disk itself. Previously though disks were addressed in CHS mode, which represented the physical construction of the disk as can be seen in the [diagram](#) below:



DOS had the requirement that its image did not span across cylinders, and so this region was added by partition managers so that the first partition was aligned on a cylinder boundary. Therefore this region's size is determined by the number of sectors (512 bytes) per cylinder. The maximum (and usual given today's disk sizes and LBA) sectors per track is 63, which leaves 62 sectors free after the MBR (31,744 bytes).

GRUB uses this region to store its stage 1.5, which is file system specific code used to find the operating system image on the "boot" file system. Currently GRUB does not need all this space as can be seen for the copies of all the stage 1.5 files in the boot partition:

```

$ ls -ls /boot/grub/*stage1_5
-rw-r--r-- 1 root root 9428 Mar  8 14:27 /boot/grub/reiserfs_stage1_5
-rw-r--r-- 1 root root 9308 Mar  8 14:27 /boot/grub/xfs_stage1_5
-rw-r--r-- 1 root root 8448 Mar  8 14:27 /boot/grub/jfs_stage1_5
.....

```

```
-rw-r--r-- 1 root root 1950 Mar  8 14:21 /boot/grub/elts_stage1_5
-rw-r--r-- 1 root root 7684 Mar  8 14:27 /boot/grub/fat_stage1_5
-rw-r--r-- 1 root root 7272 Mar  8 14:27 /boot/grub/ufs2_stage1_5
-rw-r--r-- 1 root root 7188 Mar  8 14:27 /boot/grub/minix_stage1_5
-rw-r--r-- 1 root root 7028 Mar  8 14:27 /boot/grub/iso9660_stage1_5
-rw-r--r-- 1 root root 6996 Mar  8 14:27 /boot/grub/ffs_stage1_5
-rw-r--r-- 1 root root 6612 Mar  8 14:27 /boot/grub/vstafs_stage1_5
```

Therefore one could theoretically use the last 43 sectors of this region (22,016 bytes) for anything. There is no point in creating a file system in here as the overhead would be too much, but you could dd stuff in and out like:

```
dd bs=512 seek=20 count=43 if=myfile of=/dev/hda
```

```
dd bs=512 skip=20 count=43 if=/dev/hda of=myfile
```

Note be very sure you know what your doing before running these commands.

Note on my laptop the last sector of this region contains the first sector of a MSWIN4.1 boot record (which I understand is 3 sectors in total), as it came with winxp installed (in a FAT32 partition). GRUB makes this sector redundant even in a dual boot situation, so don't worry about overwriting it. You can inspect this sector using:

```
dd bs=512 skip=62 count=1 if=/dev/hda | od -Ax -tx1z -v
```

Note also the [official grub docs](#) on these disk areas which appeared about 7 years after I wrote the notes above.

GRUB 1 boot process

GRUB or the GRand Unified Bootloader is the usual bootloader used on linux systems, and resides on the system as described above. The boot process with GRUB is as follows:

- BIOS code reads MBR from a disk (looks at last 2 bytes to verify if MBR).
- Starts executing bootloader code (GRUB stage 1).
- Bootloader jumps to location (sector num) of next stage. This sector num is stored at a particular location in the bootloader "code" at GRUB install time and usually points to a stage 1.5 located in the "DOS compat space" immediately after the MBR.
- Stage 1.5 knows about the boot file system so it opens the file system on the specified (at install time) partition. It looks for the stage 2 executable here and executes it. Note since stage 1.5 knows about the boot file system it gives much greater flexibility in upgrading stage 2 and the kernel etc. as their new locations don't need to be written to the earlier GRUB stages.
- Stage 2 contains most of the GRUB logic. It loads the menu.lst file and executes the statements, usually providing a menu to the user etc.

Subsequent steps are distro specific but for completeness I'll describe them for my fedora linux distribution:

- When GRUB starts booting one of the entries, it reads the initial ramdisk and starts the kernel running telling it about the ramdisk.
- In the initial ramdisk, the nash shell is run to parse the /linuxrc file. It essentially finds the location of the file system it itself is on and passes that to the kernel as its root (/) file system. This allows for greater flexibility of the devices the kernel resides on.
- The kernel reads its root (/) file system and executes /bin/init by default. This in turn parses /etc/inittab which usually sets up the login consoles and starts executing the scripts in /etc/init.d
- These scripts start various subsystems, culminating in starting X. X in turn starts the display manager which gives a login prompt.

GRUB 2 boot process

Here is a good overview of the [GRUB 2 boot process](#) which was written 5 years after I wrote the following notes.

The structure of [GRUB images](#) has changed quite a bit with version 2, (which is still in development at the time

of writing). Instead of stage 1, stage 1.5 and stage 2 images it has a collection of modules that can be combined in different ways. To mirror the functionality the original GRUB's stages as described above, you would have for example:

stage 1 = boot.img

stage 1.5 = diskboot.img+kernel.img+pc.mod+ext2.mod (the core image)

stage 2 = normal.mod+_chain.mod

The boot process for GRUB 2 then would be:

- BIOS code reads MBR from a disk (looks at last 2 bytes to verify if MBR).
- Starts executing bootloader code (boot.img).
- This loads the first sector of the core image (diskboot.img), whose location was stored at a particular location in boot.img at GRUB install time and usually points to the core image located in the "DOS compat space" immediately after the MBR. diskboot.img in turn loads and executes the rest of the core image.
- These core image modules know about the boot file system and can open the file system on the specified (at install time) partition. It looks for and loads modules there, including normal.mod
- normal.mod loads the grub.cfg file and executes the statements, usually providing a menu to the user etc.

This is a more flexible mechanism. For example one can prepend pxeboot.img to the core image instead of diskboot.img. This will then load the whole core image from the network and then start executing kernel.img.

Booting GRUB 2 from GRUB legacy

I hit this issue in Dec 2011 when I installed my new laptop with both Fedora 16 and RHEL 6.2. Each was given half the disk, with RHEL 6.2 installed last. Hence the MBR pointed to the boot partition in RHEL's portion of the disk; i.e. GRUB legacy. To get this in turn to boot GRUB2 as an option, in effect chainloading GRUB2 from GRUB1, I added this entry to the old GRUB menu.lst

```
title GRUB2 entries
    root (hd0,0)
    kernel /grub2/core.img
    boot
```