The GRand Unified Bootloader Explained!

An overwritten MBR (master boot record) or boot loading errors, are things that can leave most users paralysed. The purpose of this article is to familiarise you with GRUB, the default bootloader in most modern Linux systems.

GRUB (GRand Unified Bootloader) is an advanced bootloader that is capable of booting multiple operating systems on a single machine. It can load *nix as well as other proprietary operating systems. The folks from the MS Windows platform are unfortunately ignorant about the concept of bootloaders. Proprietary operating systems like Windows often hide the background features of a system, like bootloaders, from the user.

With the help of a bootloader you can theoretically load hundreds of operating systems. Most familiar Linux distros currently ship with GRUB (Figure 1), by default. In short, GRUB is what is displayed immediately after the BIOS. It enables a user to select which OS the machine should boot from a list, by using the arrow keys.

One of the biggest benefits of GRUB is that it is dynamically configurable. Lilo is another bootloader, which was once the default and has now been depreciated by most distros.

How GRUB works

When a computer boots, the BIOS passes the control to the first-boot device—it may be the hard disk, CD-ROM, floppy disk, or Flash drive. MBR is the first sector of the hard disk and is only 512 bytes in size (Figure 2). This sector consists of code required to boot a PC. MBR consists of 446 bytes of primary bootloader code and 64 bytes of the partition table. The partition table records the information regarding the primary and extended partitions. Boot loading is implemented in GRUB as Stage 1, Stage 2, Stage 1.5 (optional), etc. The

primary bootloader area (446 bytes) contains Stage 1, which in turn directs you to Stage 2 (i.e., the *menu.lst* configuration file, which has the list of operating systems on the machine).

Installing and configuring GRUB

Most Linux distros come with GRUB by default. If your distribution comes with other boot loaders like Lilo or Syslinux, you can get the latest release of GRUB, as follows:

```
$ wget ftp://alpha.gnu.org/gnu/grub/grub-1.96.tar.gz
$ tar -xzvvf grub-1.96.tar.gz -C .
$ cd grub-1.96
$ ./configure ; make
$ sudo make install
```

The next step is to configure GRUB by properly editing the *menu.lst* file. You can find the GRUB Stage 2 configuration file at */boot/grub/menu.lst*.

The next step is to add your installed operating system list to the GRUB menu. Each OS entry in GRUB will look like the following:

```
title            Ubuntu 8.04, kernel 2.6.24-17-generic
root             (hd0,4)
kernel           /boot/vmlinuz-2.6.24-17-generic root=/dev/
sda5  ro quiet splash vga=773
initrd           /boot/initrd.img-2.6.24-17-generic
quiet
```

Here, 'title' is the display name for the operating system that will appear on the GRUB bootloader screen. The following list describes what each term in the above snippet means:

- title – the display name of an operating system
- root – the partition where the kernel is located
- kernel – the path of the kernel location with specific boot parameters (space separated)
- initrd – the path of the initial ramdisk file

Going back to the snippet again, you will notice that the 'root' entry is given as (hd0,4). This is the standard GRUB naming convention, where:

- hd0 stands for the primary master hard disk
- hd1 stands for the primary slave hard disk
- hd2 stands for the secondary master hard disk
- hd3 stands for the secondary slave hard disk

Normally it would be hd0, where:

- (hd0,0) represents /dev/sda1, the first partition of the primary master hard disk
- (hd0,4) represents /dev/sda5, the first logical partition of the extended partition inside the primary master hard disk
- (hd1,0) represents /dev/sdb1, the first partition of the primary slave hard disk

Other proprietary operating systems like Windows can be loaded by a process called chainloading, as follows (without specifying the kernel or other such parameters—we will only specify the partition in which it is installed):

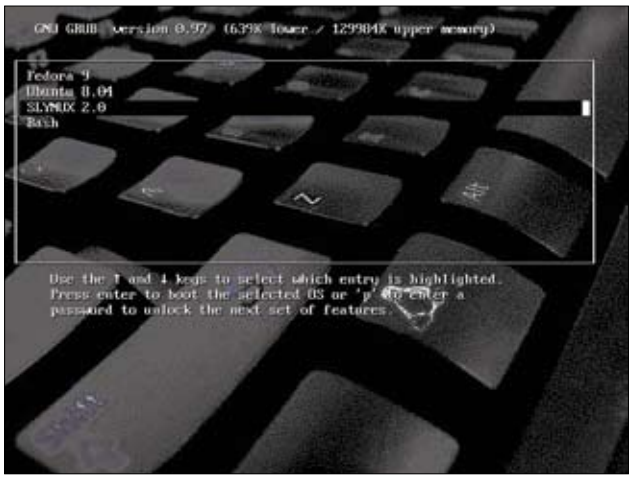

Figure 1: The GRUB bootloader menu: select the OS you want to boot

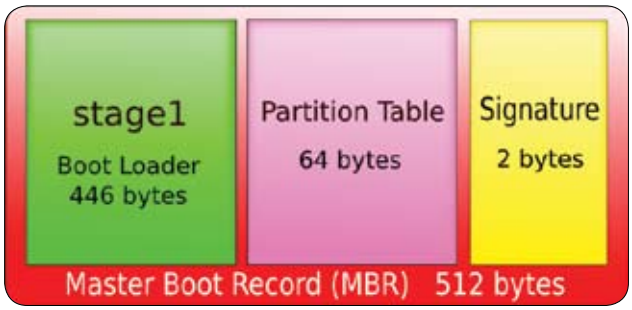

Figure 2: The MBR

```
title Windows Vista
rootnoverify (hd0,0)
chainloader +1
```

BACKING UP A PARTITION TABLE FOR INSTANT FIXES

It is a good practice to make a back-up of your partition table. It will be useful to restore the partition table in the event of a corruption.

First, generate a back-up file with the partition information as follows:

```
# sfdisk -d /dev/sda > ~/partition_table.backup
```

Now, in case of a disaster, you can always restore it as follows:

```
#  sfdisk /dev/sda < ~/partition_table.backup
```

Although, this is not related to GRUB in general, it's an important tip nonetheless, as this data is also stored in the MBR.

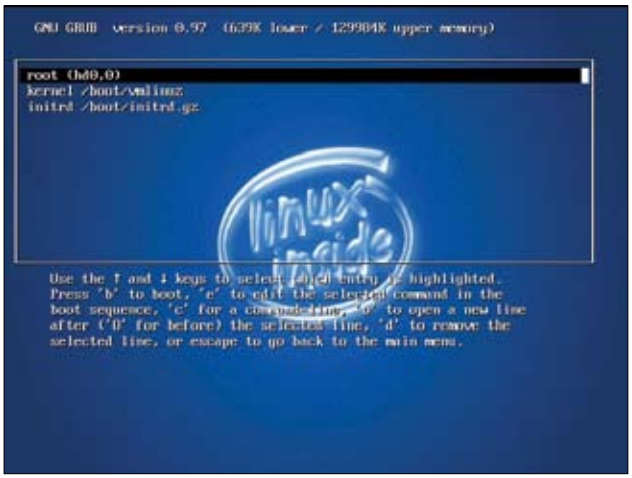

Figure 3: GRUB edit interface

The following list describes what each term in the above snippet means:

- title – the display name of an operating system
- rootnoverify – the partition in GRUB notation, where the OS is installed
- chainloader +1 – enables chainloading

Now that the GRUB nomenclature is more or less covered, let us install GRUB into MBR now—run the following command as the root user:

```
# grub-install /dev/sda
```

That's it! GRUB is now your default bootloader.

The GRUB command line

The dynamic nature of GRUB helps us to alter its configuration before loading the operating system. Also, it doesn't need to reinstall the bootloader into MBR each time we make modifications to the *menu.lst* file. Working on GRUB's command line interface is similar to a bash terminal interface.

In order to switch to the GRUB's command line interface from the bootloader, press *C*. Then you will get a prompt as shown below:

```
     GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

  [ Minimal BASH-like line editing is supported.  For the first
word, TAB
    lists possible command completions.  Anywhere else TAB
lists the possible
    completions of a device/filename.]

grub>
```

Type *Help* to take a look at the available commands.

In order to boot a kernel, issue the following commands one by one:

```
root (hd0,0)
kernel /boot/kernel
initrd /boot/initrd.gz
boot
```

For just editing the current OS entry, select the required entry and press *E (*Figure 3*)*. Then make the required modifications and press *B* to boot with the modified configuration.

Handling boot failures and MBR overwriting

A usual scenario all dual-boot (Linux and Windows) users face is when installing Windows after Linux; this causes MBR (the GRUB bootloader) to be overwritten by Windows. Following this, the computer straight away boots Windows, without displaying the entries for the other installed operating systems—this is why it is always advisable to install Linux after Windows. However, even if you've encountered a situation where you've lost GRUB, you can fix it easily.

Collect some GNU/Linux live CD like Knoppix and boot from it. If the live CD displays a GRUB menu, it is even easier. Press C to enter the GRUB command line:

```
grub> find /boot/grub/stage1
find /boot/grub/stage1
 (hd0,4)
 (hd0,8)
grub>
```

The output of the *find* command in the above snippet says that it has found two Linux installations on the system. Now, in order to install GRUB from either of these Linux installations, run the following set of commands:

```
grub> root (hd0,4)
root (hd0,4)
 Filesystem type is reiserfs, partition type 0x83

grub> setup (hd0)
grub>setup (hd0)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/reiserfs_stage1_5" exists... yes
 Running "embed /boot/grub/reiserfs_stage1_5 (hd0)"...  19
sectors are embedded.
succeeded
 Running "install /boot/grub/stage1 (hd0) (hd0)1+19 p
(hd0,4)/boot/grub/stage2 /boot/grub/grub.conf"... succeeded
Done.

grub>
```

That's it. Your Grub is now restored back into the MBR.

Alternately, you can boot into the live CD and get a root prompt:

```
# mkdir /mnt/fixroot
# mount /dev/sda5 /mnt/fixroot
# mount --bind /dev/ /mnt/fixroot/dev
# chroot /mnt/fixroot
# grub-install /dev/sda
```

What has to be done above is as follows: mount the root device (*/dev/sda5*) to */mnt/fixroot*. The devices currently available to the live system are then bound to */dev/* of the root partition (*/dev/sda5)* at */mnt/fixroot/dev*. Finally, we temporarily change-root to the filesystem at */dev/sda5* using the *chroot* command and execute *grub-install* to fix MBR. (Of course, don't forget to change /dev/sda5 to the correct Linux partition on your system.)

Forgot your root password?

If you have forgotten the root password of your Linux system, there's no need to panic! The fix is quite simple. Reboot you system. At the GRUB graphical menu, press *E* to edit and add the following parameters to the kernel argument:

```
kernel /boot/vmlinuz-2.6.24-17-generic root=/dev/sda5 rw
init=/bin/bash
```

Here, by appending the *init=/bin/bash* argument to the kernel line, we are telling Linux to immediately enter a bash prompt after booting the kernel. You can now reset the root password using the *passwd* command, as follows.

```
bash # passwd
```

Now, you may wonder that if it's so simple to reset the root password, then ordinary users can use this feature to their own advantage. The next section deals with how to password protect GRUB, so that unauthorised users can't reset root passwords.

Password protecting GRUB

Generate a MD5-encrypted password for GRUB as follows:

```
[slynux@gnubox ~]$ /sbin/grub-md5-crypt
Password:
Retype password:
$1$tIwKk$K2ZwLi3kmzjssimf7K.Sh/
[slynux@gnubox ~]$
```

Now, append the MD5 hash to your */boot/grub/menu.lst* file as follows, at the top of the file after the commented texts:

```
# menu.lst - See: grub(8), info grub, update-grub(8)
#            grub-install(8), grub-floppy(8),
#            grub-md5-crypt, /usr/share/doc/grub
#            and /usr/share/doc/grub-doc/.

password --md5 $1$tIwKk$K2ZwLi3kmzjssimf7K.Sh/
```

If GRUB is password protected, you won't be able to enter the edit mode by pressing *E*. Rather, you have to enter the password by pressing *P* first and *E* afterwards, to edit the menu.

What's next ?

GRUB has been undergoing mass development over the course of time—although what we just discussed is the default bootloader of most Linux distros, it has now been renamed to 'GRUB Legacy' by the GNU Project. GRUB 2, which is currently under development, is a new generation bootloader written completely from scratch, and its dynamic nature is improvised with certain GRUB moduling systems. It has various features ranging from i18n support, scripting support, cross-platform installation, etc. For more information, have a look at *www.gnu.org/software/grub/grub-2.en.html*. END

CUSTOMISING A SPLASH IMAGE

GRUB usually comes with a visually appealing graphical boot menu. The background picture can be customised to your tastes, though. The Splas' image is shown in the background of the GRUB bootloader screen after you switch on your PC.

In order to build a GRUB-compliant image to replace your current Splash screen, follow these steps:

1. Create an image of file format type xpm.gz, size 640 x 480 pixels with 14 colours only. An existing image can be converted to this format by using the convert command (which is a part of the ImageMagic package) as follows:

```
$ convert splash.png -resize 640x480! -colors 14 -depth 8
splash.xpm.gz
```

2. Copy splash.xpm.gz to the /boot/grub/ directory. Now, edit the /boot/grub/menu.lst as follows by adding the following line (or replacing the text if the line already exists) before the OS specifications are listed:
splashimage=(hd0,4)/boot/grub/splash.xpm.gz

Reboot your computer to check the new bootloader image.

By: Sarath Lakshman *is an 18 year old hacker and free software enthusiast from Kerala. He loves working on the awesome GNU/Linux environment and he contributes to the PiTiVi video editor project. He is also the developer of SLYNUX, a distro for newbies. He is currently studying at Model Engineering College, Cochin. He blogs at http://www.sarathlakshman.info*