Installing Multiple Distributions on the same computer

Sanket Totewar

With availability of the internet, Linux has become accessible to nearly all the parts of the world. And with the great freedom that this operating system offers, we can see a steep rise in the number of distributions released the world over. The five distributions shipped with this issue have always competed with each other for the number one position. This has lead to new features and services emerging with every release. So, to make the most of the services offered by various distributions, most users often tend to install multiple distros and switch between them. This is commonly referred to as *distro hopping* and is a common practice between newbies and experts alike.

Synopsis

This article is geared towards helping you seamlessly install and run multiple Linux distributions on the same computer. Also we shall tackle cases where MS Windows or other operating systems are to be present on the same PC. In the article, I am taking Windows as an example, because it is the most common case.

Although great care has been taken to explain various concepts in detail, this article requires some familiarity with computer jargon and is not for the faint-hearted. It also requires a brief understanding of partitions as well as boot loaders. Plus there is a very rare chance of losing data while playing with partitions. So, it is a good idea to back up all important data before advancing any further.

Figure 1. *YaST Partitioner in Action*

Boot loaders

Let us see what happens in most cases when we turn on our computer. It is the job of the BIOS to find the primary bootable device (usually a hard disk) and load the initial bootstrap program from the MBR. The MBR is an abbreviation for Master Boot Record and is the first 512 bytes of the hard disk. This area stores a boot loader program whose purpose is to load the operating system and transfer execution to it. The operating system then initializes itself by loading the necessary drivers and programs needed for its normal operation. In case of multiple operating systems, the main boot loader transfers control to other operating systems or their individual boot loaders depending on the computer configuration.

The boot loader is hence a very important utility for a Linux distro hopper. Various distributions use a variety of boot loaders like GRUB and LiLo for the above mentioned purpose. Of these, GRUB is known to be very popular due to its stability and multi-boot support. GRUB is an abbreviation for Grand Unified Boot loader and is used by GNU and most Linux distributions. Recently, GRUB 2 has replaced older versions of GRUB which are now called GRUB legacy. GRUB legacy is no longer being developed and GRUB 2 has found its way to being the default boot loader being shipped with popular distributions like Ubuntu.

GRUB 2 offers a lot of advantages over other boot loaders. GRUB 2 has a variety of built-in programs and commands that can be used to boot into various operating systems. It recognizes filesystems of almost all Linux, Win-

dows, Mac, BSD and Solaris systems. It is also possible to boot into a CD/DVD image file like ISO using GRUB 2. It is also very customizable and gives you a choice to display background images. But before we look into the installation of GRUB 2, let's take a look at the various types of partitions.

Partitioning

Your hard disk can be divided into a number of separate portions that have their own identity. They are called partitions and are useful for installing operating systems as well as storing data. There are two main types of partitions: primary and extended. A hard disk can be divided into a maximum of four primary partitions or three primary partitions and one extended.

Primary partitions are easily bootable and are used to install operating systems. Extended partitions, on the other hand can be divided into a number of sub-partitions called logical drives. Logical drivers are usually used to store data or install secondary operating systems. Partitioning can be achieved using GParted, YaST Partitioner (Figure 1) or a similar program bundled with distributions in the DVD. It is a very good idea to first resize and create the partitions and then install all operating systems. Also you should label all your partitions to avoid any confusion.

One specific advantage of Linux distributions is that it can be installed on primary partitions as well as logical drives. There are various ways of installing them based on your current partition table. Let us see a few common cases in which our mission can be accomplished.

Case 1

Windows or another operating system already installed on the computer. As we all know, Windows tends to occupy the first primary partition for its system files. And some versions of Windows (like Windows 7) also create a primary partition named *System Reserved* for its normal working. So we are usually left with one primary and one extended partition for installing Linux distributions.

Figure 2 shows the disk partitioning map for case 1. Here, the first two

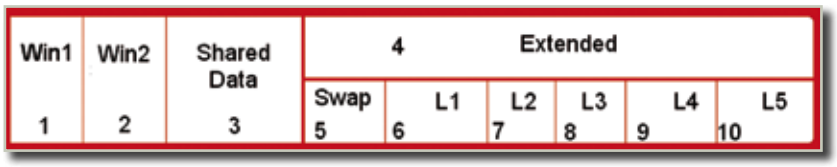

Figure 2. *Disk Partitioning – Case 1*

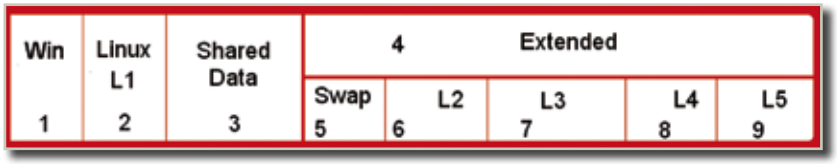

Figure 3. *Disk Partitioning – Case 2*

partitions (1) and (2) are reserved for MS Windows. The third primary partition (3) can be used for shared data. It is a good idea to keep the shared data as a primary partition so that it can be kept separate from the operating systems. To ensure that this data can be accessed in all operating systems, you can choose Fat 32 or NTFS as the file system.

The remaining space is converted into an extended partition (4). This partition consists of six logical drives (5 to 10). The first logical drive (5) is for the system swap. I have kept this as the first logical drive as one doesn't tend to introduce changes in the size of the swap. The other drives are for the various Linux distributions that you shall install. The number of logical drives may vary depending upon how many distributions you want to try.

Case 2

Installation on an empty hard disk. This case gives us the advantage of creating a partition table of our choice. Figure 3 shows a partition table where I have installed Windows (1) and one main Linux distribution (2). While installing Windows, I ensured that it is uses one partition only for its boot loader as well as system files. I am resolving to retain these operating systems for a long time and hence they are on primary partitions. The primary partition (3) having a NTFS filesystem contains shared data. The remaining space has been converted into an extended partition (4). Logical drive (5) is for Linux swap and is accessible for all distributions. Logical drives (6) to (9) contain other distributions that I may not retain in the long run.

Note: It is a good idea to install Windows before the Linux distribution as it will replace the MBR with its own entries.

Booting (Cases 1 & 2)

In the above cases, there is a special way of installing the Linux distributions. The boot loader of only one distribution can be installed in the MBR. This distribution will be responsible for updating the boot loader configuration files. Whenever we install or remove various kernels or distributions, this activity has to be triggered. This is because the main boot loader will never know about the changes made in other operating systems.

Windows uses its own boot loader which is saved on the system partition or the *System Reserved* partition. However, for multi-booting Linux distributions, you shall have to use the boot loader of one of the Linux distributions. If you are installing Ubuntu as one of the distributions, it is a good idea to install its boot loader in the MBR. This is because Ubuntu by default contains GRUB 2 which is our preferred boot loader. The boot loaders of the other distributions must be installed on their respective logical drives containing the root partition. Now your computer will show you the GRUB 2 menu to choose from all the existing OS.

GRUB 2 installed by Ubuntu will recognize the different kernels of the Linux distributions as well as other operating systems. However, it does not check for them at every boot. After installation, GRUB 2 generates a *grub.cfg* file that contains this information. This file is updated automatically every time you install a new ker-

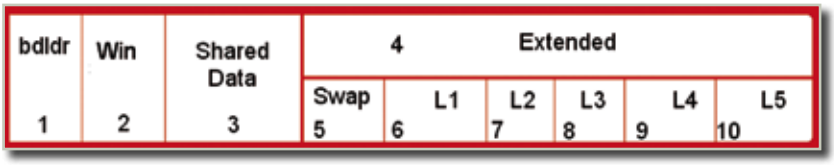

Figure 4. *Disk Partitioning – Case 3*

nel in Ubuntu. But how will GRUB 2 know about the changes made in other distributions? There is a simple command for this. All you have to do is log into your Ubuntu installation and enter the command line. In the Terminal, simply type `update-grub` which will automatically update the *grub.cfg* file. The `update-grub` command requires root privileges and you may have to add `sudo` as a prefix.

Case 3

Installation of the boot loader in a separate partition. This is a special case where I created a separate partition for the main boot loader GRUB 2. This boot loader is installed in the MBR and is executed first on every boot. And depending upon its configuration file, it will hand over execution to the other system boot loaders.

Figure 4 shows a partition table where I have made a special partition for the boot loader. Windows and shared data are installed on other primary partitions. The extended partition houses Linux swap and the Linux distributions. The biggest advantage of such a system is that the removal of any operating system does not hamper my ability of booting into other systems. I thus have complete control over booting my operating systems. Note that all the Linux distributions were installed with the default boot loaders in their respective partitions. Some booting options can also be changed after installation of the system. A good example is the YaST boot loader configuration menu as shown in Figure 5.

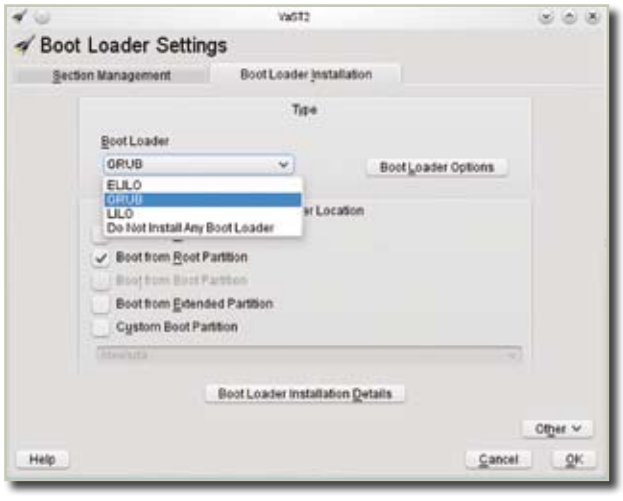

Figure 5. *YaST Boot Loader Configuration*

Installation of boot loader on partition

The latest GRUB 2 boot loader can be manually installed by downloading from the GNU website. It should be unpacked, installed and configured for the MBR. But it is easier to install it via the Ubuntu Live session instead as you already have it readily available. Changing the default boot loaders for the other distributions is not recommended as GRUB 2 might not be supported in them yet.

Boot into the Ubuntu Live session and open a terminal. Type `grub-install -version` to find the version of the existing GRUB installation. Mount the partition for installation of the boot loader. Here I have labeled my boot loader partition as *btldr* and it exists at */dev/sda1*. Here sda states that it is the first sata hard disk. If you are unsure about the position of your partition, check it using either GParted or the command `sudo fdisk -l` in the Terminal. Now use the command `grub-install --root-directory=/media/btldr /dev/sda` to install GRUB 2 in the MBR and its files on the defined partition. The terminal should give a message *Installation finished. No error reported* which means that the installation is successful. On entering the partition, you will see a directory */boot* that contains necessary GRUB 2 files. When you reboot the computer, you will be taken to a GRUB prompt. You can easily enter any operating system directly by loading a kernel and using the `boot` command. However it is recommended to generate a *grub.cfg* file that shows a menu at every system boot.

GRUB menu entries

The above commands just install GRUB 2 on the partition and MBR. This however does not create the configuration file *grub.cfg* that is essential to show a menu during boot. This file can be easily prepared using a text editor like gedit. Just enter the *btldr/boot/grub* directory and create an empty text file having *grub.cfg* as its name. Here we can create menu entries for chainloading to other operating systems.

Tip: You can also run the `grub-mkconfig` command and direct it to our boot loader partition. But this has to be done only from a native Ubuntu or Debian installation. First make sure that the boot loader partition is mounted. Then run the command `sudo grub-mkconfig -o /media/btldr/boot/grub/grub.cfg` which will update this file automatically.

GRUB naming convention is very simple to interpret. GRUB uses *hd* for a hard disks and the drive numbers are counted from zero based on their BIOS order. The partitions in the hard disk are numbered from one based on their position. Hence the third partition on the second hard disk is denoted as (hd1,3).

I have a Windows partition at (hd1,3) with NTFS as the file system. So, the menu entry for the same will be as follows:

```
menuentry " My Windows Installation " {
insmod ntfs
set root='(hd1,3)'
drivemap -s (hd0) ${root}
chainloader +1
}
```

The `chainloader` command tells GRUB 2 to hand over the control to the first few bytes of that specific partition. The `drivemap` command in GRUB 2 is useful for chainloading operating systems like Windows especially when it does not reside at the first drive.

I have an openSUSE installation at (hd0,7) with ext4 file system. GRUB recognizes all file systems above ext2 as an ext2 file system. So, the menu entry will be as follows:

```
menuentry " My openSUSE " {
insmod ext2
set root=(hd0,7)'
chainloader +1
 }
```

Entries can similarly be created for other distributions too. Copy the entries to the *grub.cfg* file and save it. The order of the copied entries depicts the order in which the entries will be shown in the boot menu.

To make a certain entry as the default highlighted entry in every boot, adding the following line to the beginning of *grub.cfg*: `set default="2"` Here 2 implies that the third entry is to be highlighted as counting is done from zero.

The following command when added will set a timeout in seconds for the highlighted entry. As soon as the timer reaches zero, the highlighted entry will be activated: `Set timeout=7` Here 7 implies that seven seconds will pass before the highlighted entry is selected. This is how we make a minimal *grub.cfg* file. More options can be added to this file to improve the user experience during booting. Also we can have a beautiful background image while the timer is running. These tutorials are however out of the scope of this article.

Booting (Case 3)

Here when the computer starts, you will be taken to a GRUB 2 boot menu depending on your configuration file. You can choose and entry using your keyboard keys. Or the default entry will get activated after the timeout. If the selected entry leads to another boot loader, that boot loader will get activated and you will see its menu now. This will go on till Windows or a Linux kernel is selected. The selected operating system will then be loaded.

Real Life Scenario

Figure 6 shows a partition table where I had only Linux distributions installed on this disk. The boot loader is installed in the MBR while its files are saved on this partition. The other two primary partitions are for Linux swap and my Ubuntu installation respectively. I have divided the extended partition into four logical drives. Of these, two are responsible for storage while the other two contain openSUSE and Fedora installations.

My Windows XP installation is present on the second disk i.e. at (hd1,1). For the above partition table my *grub.cfg* file at */btldr/boot/grub* has the following entries:

```
##// GRUB 2 configuration file ↵
   //##
##// manually created by SuNk8 ↵
   //##

set default="2"
set timeout=7

menuentry "Sanket's Ubuntu ↵
   Installation"{
insmod ext2
set root='(hd0,6)'
chainloader +1
}

menuentry "Sanket's openSUSE ↵
   Installation"{
insmod ext2
set root='(hd0,7)'
chainloader +1
}

menuentry "Sanket's Linux ↵
   Installation"{
insmod ext2
set root='(hd0,8)'
chainloader +1
}

menuentry "Windows XP on 160 ↵
   GB"{
insmod ntfs
set root='(hd1,1)'
drivemap -s (hd0) ${root}
chainloader +1
}
```

One day, I decided to remove Ubuntu and install Mandriva instead. I was able to accomplish this easily due to the separate boot partition. All I had to do was edit a few lines in the configuration file. There is no worry about facing boot problems just because I removed my main distribution. ■

Further reading

- *http://www.gnu.org/software/grub/*
- *http://members.iinet.net/~herman546/*
- *https://help.ubuntu.com/community/Grub2*

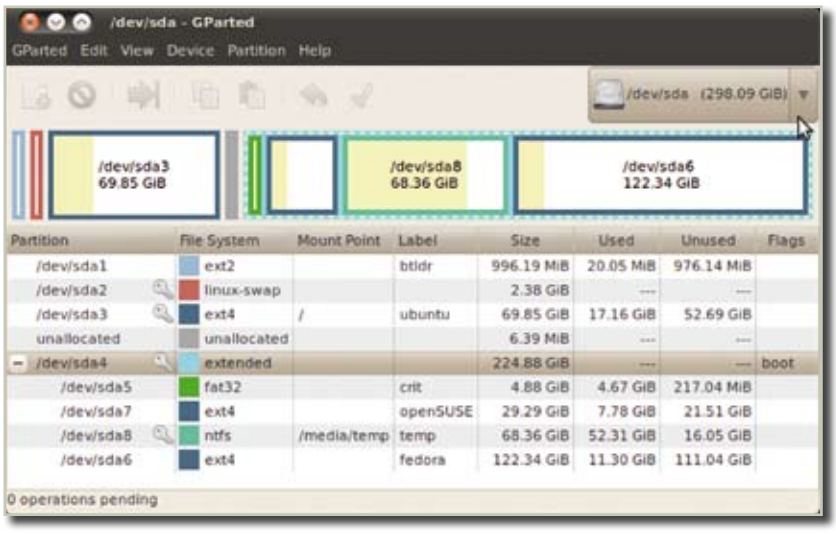

Figure 6. *Using GParted to resize and create partitions*