Learn Linux, 101: Boot managers

Introducing GRUB, GRUB 2, and LILO

Ian Shields
Senior Programmer
IBM

04 December 2012
(First published 13 April 2010)

Learn how to choose and configure a boot manager for your Linux® system. You can use the material in this article to study for the LPI 101 exam for Linux system administrator certification, or just to learn for fun.
04 Dec 2012 - *This article updated to include material for the LPI Exam 101: Objective Changes as of July 2, 2012.*

Major updates include new information about GRUB 2 in GRUB 2 plus GRUB 2 rescue images and flash drives in Rescue media for GRUB 2. LPI has dropped LILO from the objectives so the LILO section appears later in the article. Other sections contain minor updates that reflect GRUB 2 as part of the objectives and the revised content order.

View more content in this series

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for Linux Professional Institute Certification level 1 (LPIC-1) exams.

See our developerWorks roadmap for LPIC-1 for a description of and link to each article in this series. The roadmap is in progress and reflects the latest (April 2009 with minor updates in July 2012) objectives for the LPIC-1 exams: as we complete articles, we add them to the roadmap. In the meantime, though, you can find earlier versions of similar material, supporting previous LPIC-1 objectives prior to April 2009, in our LPI certification exam prep tutorials.

Overview

In this article, learn to choose, install, and configure a boot manager for a Linux system. Learn to:

- Configure multiple boot locations and backup boot options
- Install and configure a boot loader such as GRUB, GRUB 2, or LILO
- Add boot parameters or run boot loader commands at boot time

- Recover from common booting problems
- Boot a GRUB 2 system using GRUB Legacy

This article discusses the PC boot process and the three main boot loaders used in Linux—GRUB, GRUB 2, and LILO—and helps you prepare for Objective 102.2 in Topic 102 of the Linux Professional Institute's Junior Level Administration (LPIC-1) exam 101. The objective has a weight of 2. Note that LILO is no longer required for LPIC-1. We include it here so you will know something about it.

Prerequisites

To get the most from the articles in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. You should also be familiar with hard drive layout as discussed in the article "Learn Linux 101: Hard disk layout." Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here.

Note: The images in this article are screen captures taken from early in the boot process. If you are reading this article using a screen reader, you may find it advantageous to have the corresponding configuration files available for reference; download them from the Download section later in this article.

Boot process overview

Before we get into specific boot loaders, let's review how a PC starts or *boots*. Code called *BIOS* (for *B*asic *I*nput *O*utput *S*ervice) is stored in non-volatile memory such as a ROM, EEPROM, or flash memory. When the PC is turned on or rebooted, this code is executed. Usually it performs a power-on self test (POST) to check the machine. Finally, it loads the first sector from the master boot record (MBR) on the boot drive.

As discussed in the article "Learn Linux 101: Hard disk layout," the MBR also contains the partition table, so the amount of executable code in the MBR is less than 512 bytes, which is not very much code. Note that every disk, even a floppy, contains executable code in its MBR, even if the code is only enough to put out a message such as "Non-bootable disk in drive A:". This code that is loaded by BIOS from this first sector is called the *first stage boot loader* or the *stage 1 boot loader*.

The standard hard drive MBR used by MS DOS, PC DOS, and Windows® operating systems checks the partition table to find a primary partition on the boot drive that is marked as *active*, loads the first sector from that partition, and passes control to the beginning of the loaded code. This new piece of code is also known as the *partition boot record*. The partition boot record is

actually another stage 1 boot loader, but this one has just enough intelligence to load a set of blocks from the partition. The code in this new set of blocks is called the *stage 2 boot loader*. As used by MS-DOS and PC-DOS, the stage 2 loader proceeds directly to load the rest of operating system. This is how your operating system pulls itself up by its bootstraps until it is up and running.

This works fine for a system with a single operating system. What happens if you want multiple operating systems, say OS/2, Windows XP, and three different Linux distributions? You *could* use some program (such as the DOS FDISK program) to change the active partition and reboot. This is cumbersome. Furthermore, a disk can have only four primary partitions, and the standard MBR can have only one active primary partition; it cannot boot from a logical partition. But our hypothetical example cited five operating systems, each of which needs a partition. Oops!

The solution lies in using some special code that allows a user to choose which operating system to boot. Examples include:

Loadlin
: A DOS executable program that is invoked from a running DOS system to boot a Linux partition. This was popular when setting up a multi-boot system was a complex and risky process.

OS/2 Boot Manager
: A program that is installed in a small dedicated partition. The partition is marked active, and the standard MBR boot process starts the OS/2 Boot Manager, which presents a menu allowing you to choose which operating system to boot.

A smart boot loader
: A program that can reside on an operating system partition and is invoked either by the partition boot record of an active partition or by the master boot record. Examples include:
  - BootMagic, part of Norton PartitionMagic
  - LILO, the LInux LOader
  - GRUB, the GRand Unified Boot loader (now referred to as GRUB Legacy)
  - GRUB 2, a newer boot loader that is now used in many common distributions

Evidently, if you can pass control of the system to some program that has more than 512 bytes of code to accomplish its task, then it isn't too hard to allow booting from logical partitions, or booting from partitions that are not on the boot drive. All of these solutions allow these possibilities, either because they can load a boot record from an arbitrary partition, or because they have some understanding of what file or files to load to start the boot process.

Chain loading

When a boot manager gets control, one thing that it could possibly load is another boot manager. This is called *chain loading*, and it most frequently occurs when the boot manager that is located in the master boot record (MBR) loads the boot loader that is in a partition boot record. This is almost always done when a Linux boot loader is asked to boot a Windows or DOS partition, but can also be done when the Linux boot loader for one Linux system (say Fedora) is configured to load the boot loader for another Linux system (say Ubuntu). For example, you might use GRUB in one partition to launch the GRUB boot loader in another partition's boot record in order to start the Linux system in that partition. This is not common but illustrates the possibilities.

Linux boot loaders

From here on, we will focus on GRUB, GRUB 2, and LILO as these are the boot loaders included with most Linux distributions. LILO has been around for a while. GRUB is newer. The original GRUB has now become *GRUB Legacy*, and GRUB 2 is being developed under the auspices of the Free Software Foundation (see Resources for details). We will discuss GRUB, then GRUB 2 so you have some idea of the major differences and how GRUB and GRUB 2 can coexist. For the rest of this article, we assume GRUB means GRUB Legacy, unless the context specifically implies GRUB 2. A new version of LILO called *ELILO* (which is designed for booting systems that use Intel's *Extensible Firmware Interface*, or *EFI*, rather than BIOS) is also available. See Resources for additional information about GRUB 2 and ELILO.

The installation process for your distribution will probably give you a choice of which boot loader to set up. Any of GRUB, GRUB 2 or LILO will work with most modern disks, although some distributions, notably Fedora, no longer ship LILO. Remember that disk technology has advanced rapidly, so you should always make sure that your chosen boot loader, as well as your chosen Linux distribution (or other operating system), as well as your system BIOS, will work with your shiny new disk. Failure to do so may result in loss of data. Likewise, if you're adding a new distribution to an existing system, you may need to make sure you have the latest LILO, GRUB, or GRUB 2 in your MBR. You will also need a fairly recent version of your boot loader if you plan to boot from an LVM or RAID disk.

The stage 2 loaders used in LILO and GRUB allow you to choose from several operating systems or versions to load. However, LILO and GRUB differ significantly in that a change to the system requires you to use a command to recreate the LILO boot setup whenever you upgrade a kernel or make certain other changes to your system, while GRUB can accomplish this through a configuration text file that you can edit. GRUB 2 also requires a rebuild from a configuration file that is normally stored in /etc.

GRUB

GRUB, or the GRand Unified Boot loader, is one of the most common Linux boot loaders. GRUB can be installed into the MBR of your bootable hard drive, or into the partition boot record of a partition. It can also be installed on removable devices such as floppy disks, CDs, or USB keys. It is a good idea to practice on a floppy disk or USB key if you are not already familiar with GRUB. We show you how in our examples.

GRUB, or GNU GRUB, is now developed under the auspices of the Free Software Foundation. A new version, GRUB 2 is under development and is now shipping in Linux distributions, so the original GRUB 0.9x versions are now known as Grub Legacy.

During Linux installation, you will often specify your choice of boot manager. If you choose LILO, then you may not have GRUB installed. If you do not have GRUB installed, then you will need to install the package for it. From this point on, we will assume that you already have the GRUB package installed. See our series roadmap for the articles on package management if you need help with this.

GRUB (Legacy) has a configuration file that is usually stored in /boot/grub/grub.conf. If your file system supports symbolic links, as most Linux file systems do, you will probably have /boot/grub/menu.lst as a symbolic link to /boot/grub/grub.conf.

The `grub` command (/sbin/grub, or, on some systems, /usr/sbin/grub) is a small, but reasonably powerful shell that supports several commands for installing GRUB, booting systems, locating and displaying configuration files, and similar tasks. This shell shares much code with the second stage GRUB boot loader, so it is useful to learn about GRUB without having to boot to a second stage GRUB environment. The GRUB stage 2 runs either in *menu mode*, to allow you to choose an operating system from a menu, or in *command mode*, where you specify individual commands to load a system. There are also several other commands, such as `grub-install`, that use the grub shell and help automate tasks such as installing GRUB.

Listing 1 shows part of a GRUB configuration file. As you look through it, remember one important thing: GRUB, at least GRUB Legacy, counting drives, partitions, and things that need to be counted, starts at 0 rather than 1. Also note that the first kernel line is very long. We show it with a backslash (\) indicating where we broke it for publication.

Listing 1. /boot/grub/menu.lst GRUB configuration example

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,5)
#          kernel /boot/vmlinuz-version ro root=/dev/sda6
#          initrd /boot/initrd-version.img
#boot=/dev/sda6
default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
#hiddenmenu
password --md5 $1$RW1VW/$4XGAklxB7/GJk0uO47Srx1
title Upgrade to Fedora 11 (Leonidas)
 kernel /boot/upgrade/vmlinuz preupgrade \
      repo=hd::/var/cache/yum/preupgrade stage2=\
      hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/install.img \
      ks=hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/ks.cfg
 initrd /boot/upgrade/initrd.img
title Fedora (2.6.26.8-57.fc8)
 root (hd0,5)
 kernel /boot/vmlinuz-2.6.26.8-57.fc8 ro root=LABEL=FEDORA8 rhgb quiet
 initrd /boot/initrd-2.6.26.8-57.fc8.img
title Fedora (2.6.26.6-49.fc8)
 root (hd0,5)
 kernel /boot/vmlinuz-2.6.26.6-49.fc8 ro root=LABEL=FEDORA8 rhgb quiet
 initrd /boot/initrd-2.6.26.6-49.fc8.img
title GRUB Menu
 rootnoverify (hd0,1)
 chainloader +1
title Windows
 rootnoverify (hd0,0)
 chainloader +1
```

The first set of options above control how GRUB operates. For GRUB, these are called *menu commands*, and they must appear before other commands. The remaining sections give per-

image options for the operating systems that we want to allow GRUB to boot. Note that "title" is considered a menu command. Each instance of title is followed by one or more general or menu entry commands. Our LILO example was a typical basic example for a dual boot system with Windows and two Linux systems. This example comes from the same system that we used before. In fact, it is the boot menu that is chain loaded from the "Fedora 8" entry in the LILO menu. You will recognize many of the same kinds of elements occurring in both LILO and GRUB configuration files. You might like to think about what would need to change if you added the extra operating systems here to the earlier LILO example.

The menu commands that apply to all other sections in our example are:

#
: Any line starting with a # is a comment and is ignored by GRUB. This particular configuration file was originally generated by anaconda, the Red Hat installer. You will probably find comments added to your GRUB configuration file if you install GRUB when you install Linux. The comments will often serve as an aid to the system upgrade program so that your GRUB configuration can be kept current with upgraded kernels. Pay attention to any markers that are left for this purpose if you edit the configuration yourself.

default
: Specifies which system to load if the user does not make a choice within a timeout. In our example, default=2 means to load the **third** entry. Remember that GRUB counts from 0 rather than 1. If not specified, then the default is to boot the first entry, entry number 0.

timeout
: Specifies a timeout in seconds before booting the default entry. Note that LILO uses tenths of a second for timeouts, while GRUB uses whole seconds.

splashimage
: Specifies the background, or *splash*, image to be displayed with the boot menu. GRUB Legacy refers to the first hard drive as (hd0) and the first partition on that drive as (hd0,0), so the specification of splashimage=(hd0,6)/boot/grub/splash.xpm.gz means to use the file /boot/grub/splash.xpm.gz located on partition 7 of the first hard drive. Remember to count from 0. Note also that the image is an XPM file compressed with gzip. Support for splashimage is a patch that may or may not be included in your distribution.

password
: Specifies a password that must be entered before a user can unlock the menu and either edit a configuration line or enter GRUB commands. The password may be in clear text. GRUB also permits passwords to be stored as an MD5 digest, as in our example. This is somewhat more secure, and most administrators will set a password. Without a password, a user has complete access to the GRUB command line.

Our example shows two Fedora 8 kernels and an upgrade to Fedora 11 option, plus Windows XP and a chain loaded "GRUB Menu" option. The commands used in these sections are:

title
: Is a descriptive title that is shown as the menu item when Grub boots. You use the arrow keys to move up and down through the title list and then press the **Enter** key to select a particular entry.

root
: Specifies the partition that will be booted. As with splashimage, remember that counting starts at 0, so the first Red Hat system that is specified as root (hd0,6) is actually on partition 7 of the first hard drive (/dev/hda7 in this case), while the first Ubuntu system, which is specified as root (hd1,10), is on the second hard drive (/dev/hdb11). GRUB will attempt to mount this partition to check it and provide values to the booted operating system in some cases.

kernel
: Specifies the kernel image to be loaded and any required kernel parameters. This is similar to a combination of the LILO `image` and `append` commands that you will see in the section on LILO. We have two different Fedora 8 kernels, plus the upgrade kernel in this example.

initrd
: Is the name of the *initial RAM disk*, which contains modules needed by the kernel before your file systems are mounted.

savedefault
: Is not used in this example. If the menu command `default=saved` is specified, and the `savedefault` command is specified for an operating system, then booting that operating system will cause it to become the default until another operating system with `savedefault` specified is booted. In this example, the specification of default=2 will override any saved default.

boot
: Is an optional parameter that instructs GRUB to boot the selected operating system. This is the default action when all commands for a selection have been processed.

lock
: Is not used in this example. This will not boot the specified entry until a password is entered. If you use this, then you should also specify a password in the initial options; otherwise, a user can edit out your lock option and boot the system, or add "single" to one of the other entries. It is possible to specify a different password for individual entries if you wish.

rootnoverify
: Is similar to root, except that GRUB does not attempt to mount the file system or verify its parameters. This is usually used for file systems such as NTFS that are not supported by GRUB. You might also use this if you want GRUB to load the master boot record on a hard drive, for example, to access a different configuration file, or to reload your previous boot loader.

chainloader
: Specifies that another file will be loaded as a stage 1 file. The value "+1" is equivalent to 0+1, which means to load one sector starting at sector 0; that is, load the first sector from the device specified by root or rootnoverify.

You now have some idea of what you might find in a typical /boot/grub/grub.conf (or /boot/grub/menu.lst) file. There are many other GRUB commands to provide extensive control over the boot process as well as help with installing GRUB and other tasks. You can learn more about these in the GRUB manual, which should be available on your system through the command `info grub`.

Now that we have a GRUB configuration file, we need to create a boot floppy to test it. The simplest way to do this is to use the `grub-install` command as shown in Listing 2. If you are installing GRUB onto a floppy or onto a partition, then you should unmount the device first. This

does not apply if you are installing GRUB in the MBR of a hard drive, as you only mount partitions (/dev/sda1, /dev/sda2, etc.) and not the whole hard drive (/dev/sda).

Listing 2. Installing GRUB to a floppy disk

```
[root@pinguino ~]# grub-install /dev/fd0
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0) /dev/fd0
(hd0) /dev/sda
```

Note: You may also use the GRUB device name (fd0) instead of /dev/fd0, but if you do, you must enclose it in quotes to avoid shell interpretation of the parentheses. For example: `grub-install '(fd0)'`

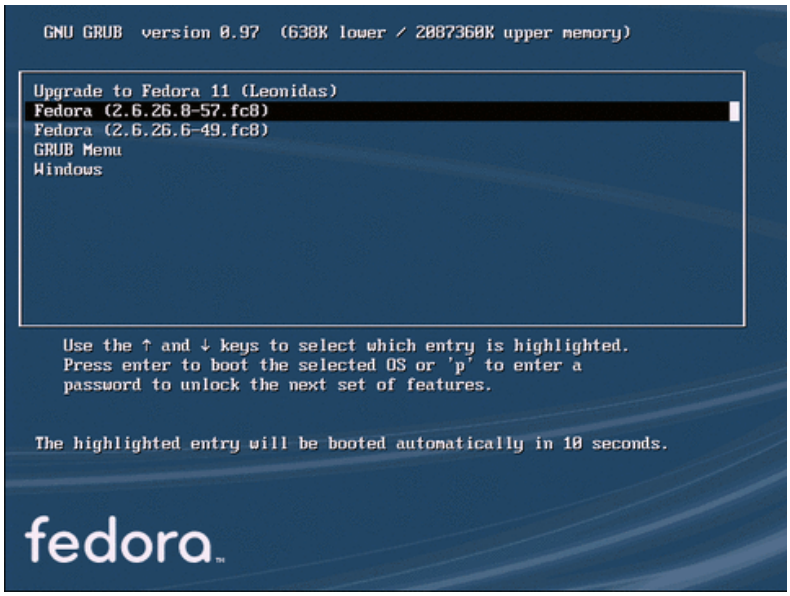
The device map tells you how GRUB will match its internal view of your disks (fd0, hd0, hd1) to the Linux view (/dev/fd0, /dev/sda, /dev/sdb). On a system with one or two IDE hard drives and perhaps a floppy drive, this will probably be correct. If a device map already exists, GRUB will reuse it without probing. If you just added a new drive and want to force a new device map to be generated, add the `--recheck` option to the `grub-install` command. Sometimes you may see a message like that shown in Listing 3. In this case too, you will need to add the `--recheck` option to the `grub` command to force GRUB to probe the available devices and rebuild a device map.

Listing 3. Installing GRUB to a floppy disk

```
[root@pinguino ~]# grub-install /dev/fd0
/dev/fd0 does not have any corresponding BIOS drive.
```

If you started with an empty floppy and now mount it, you will discover that it still appears to be empty. What has happened is that GRUB wrote a customized stage 1 loader to the first sector of the disk. This does not show up in the file system. This stage 1 loader will load stage 2 and the configuration file from your hard drive. Try booting the diskette and you will see very little IO activity before your menu is displayed. You use the up and down arrows to select different boot targets.

Figure 1. Booting using GRUB



Once you have tested your boot floppy, you are ready to install GRUB in the MBR of your hard drive. For the first IDE hard drive, you would use:

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

or

```
grub-install '(hd0)'
```

To install it into the partition boot record for partition 11, use:

```
grub-install /dev/sda11
```

or

```
grub-install '(hd0,10)'
```

Remember that GRUB numbers from 0.

You can add boot parameters to the command. In fact, you can edit any of the commands in the GRUB configuration or even load another configuration file at boot time. If you used a password, you will need to enter the `p` command and then type your password. After that, you select an entry and then enter the `e` command to edit it. Select the line you want to edit and enter the `e` again. Edit the line by typing in new characters or erasing existing characters, and then press **Enter** to return to the menu. Enter the `b` command to boot your system with the modified values. Figure 2 shows the command editing phase of this process where you are adding the 'single' parameter to the kernel statement to boot into single-user mode.

Figure 2. Booting into single user mode with GRUB

GRUB 2

GRUB 2 was rewritten from scratch to make it significantly more modular and portable. It targets different architectures and boot methods and has many new features. See Resources for links to more information. If you work with GRUB and begin to use GRUB 2, you will find it is completely different and you will probably get many surprises.

The first thing you might notice about GRUB 2 is that it does not install as a partition boot loader. Although I told the Ubuntu installer to install GRUB on the partition that I installed Ubuntu 9.10 on (/dev/sda7), that partition is not bootable by chain loading. The rest of this section is a brief introduction to GRUB 2 with enough information to help you begin using it either alone or in conjunction with GRUB Legacy.

The version of GRUB 2 for PCs is distributed in the grub-pc package. You will find several programs, normally in .usr/bin or /usr/sbin, that are part of the grub-pc package as shown in Listing 4. At the time of writing, the man pages for these programs are minimal. Try running the programs with the `--help` option to get more information, or search the Internet for additional help.

Listing 4. GRUB 2 executables in /usr/bin and /usr/sbin

```
ian@pinguino:~$ dpkg -L grub-pc|grep "bin/"
/usr/bin/grub-mkimage
/usr/bin/grub-mkrescue
/usr/sbin/grub-setup
/usr/sbin/update-grub2
/usr/sbin/update-grub
/usr/sbin/grub-install
/usr/sbin/upgrade-from-grub-legacy
/usr/sbin/grub-set-default
/usr/sbin/grub-reboot
```

The heart of GRUB 2 is a multiboot kernel (/boot/grub/core.img) along with a configuration file (/boot/grub/grub.cfg). These will be generated for you if you run `grub-install` and set the target as

your MBR (for example: `grub-install /dev/sda`). Run `grub-install --help` as shown in Listing 5 to get an idea of the programs that get called to do all the work. Some things are similar to Grub Legacy, but there are a number of new items, such as `--modules`, `--grub-setup`, `--grub-mkimage`, and so on.

Listing 5. GRUB 2 help for grub-install

```
ian@pinguino:~$ grub-install --help
Usage: grub-install [OPTION] install_device
Install GRUB on your drive.

  -h, --help              print this message and exit
  -v, --version           print the version information and exit
  --modules=MODULES       pre-load specified modules MODULES
  --root-directory=DIR    install GRUB images under the directory DIR
                          instead of the root directory
  --grub-setup=FILE       use FILE as grub-setup
  --grub-mkimage=FILE     use FILE as grub-mkimage
  --grub-mkdevicemap=FILE use FILE as grub-mkdevicemap
  --grub-probe=FILE       use FILE as grub-probe
  --no-floppy             do not probe any floppy drive
  --recheck               probe a device map even if it already exists
  --force                 install even if problems are detected
  --disk-module=MODULE    disk module to use

INSTALL_DEVICE can be a GRUB device name or a system device filename.

grub-install copies GRUB images into the DIR/boot directory specified by
--root-directory, and uses grub-setup to install grub into the boot
sector.

Report bugs to <bug-grub@gnu.org>.
```

If you run `grub-install /dev/sda`, the process will build a core image file for you, build a configuration file, and install GRUB 2 in your MBR. If you're not ready to commit to GRUB2 for your whole setup, you can build these parts yourself and then boot the GRUB 2 core image from GRUB Legacy or LILO.

Building the GRUB 2 core image

The easiest way to build a new core image file is to run grub-install with the option `--grub-setup=/bin/true`, which causes /bin/true to be run instead of the real GRUB 2 setup program, as shown in Listing 6. We first remove the existing core.img to show that the file is indeed generated.

Listing 6. Building the GRUB 2 core image with grub-install

```
ian@pinguino:~$ sudo rm /boot/grub/core.img
ian@pinguino:~$ sudo grub-install --grub-setup=/bin/true /dev/sda
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(hd0) /dev/sda
ian@pinguino:~$ ls -l /boot/grub/core.img
-rw-r--r-- 1 root root 25105 2010-04-08 12:10 /boot/grub/core.img
```

Building the GRUB 2 configuration file

The GRUB 2 configuration file is normally /boot/grub/grub.cfg. Unlike GRUB Legacy, you should not normally edit this file yourself. You should build it using either `grub-mkconfig` or the `update-grub` command which is a front-end to `grub-mkconfig`. These commands look for general settings, such as background, timeouts, and so on, in /etc/default/grub, and then run executables from /etc/grub.d/ to build various parts of the configuration file, such as the header, a section for the current Linux distribution, sections for other operating systems, and your own custom additions. If you need to customize the GRUB 2 menu, you should add your changes to a file in /etc/grub.d/ such as 40_custom, or add your own file. remember that it needs to be executable. You will then run `update-grub` to generate a new /boot/grub/grub.cfg file as shown in Listing 7.

Listing 7. Building a GRUB 2 configuration file with update-grub

```
ian@pinguino:~$ sudo update-grub
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-2.6.31-20-generic
Found initrd image: /boot/initrd.img-2.6.31-20-generic
Found linux image: /boot/vmlinuz-2.6.31-14-generic
Found initrd image: /boot/initrd.img-2.6.31-14-generic
Found memtest86+ image: /boot/memtest86+.bin
Found GRUB boot loader on /dev/sda1
Found Fedora release 8 (Werewolf) on /dev/sda6
done
```

Listing 8 shows a few entries from the resulting configuration file. We have indicated long lines that we broke for publication using a trailing backslash (\). Notice that the menuentry stanzas look more like shell scripts than the plain commands without logic of GRUB Legacy. Another important change from GRUB Legacy is that partition numbering now starts at 1, although disk numbering still starts at 0. So /dev/sda7 is (hd0,7) in GRUB 2 where it would be (hd0,6) in GRUB Legacy.

Listing 8. Partial GRUB 2 configuration file

```
### BEGIN /etc/grub.d/05_debian_theme ###
set menu_color_normal=white/black
set menu_color_highlight=black/white
### END /etc/grub.d/05_debian_theme ###

### BEGIN /etc/grub.d/10_linux ###
menuentry "Ubuntu, Linux 2.6.31-20-generic" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
        set quiet=1
        insmod ext2
        set root=(hd0,7)
        search --no-floppy --fs-uuid --set 8954fa66-e11f-42dc-91f0-b4aa480fa103
        linux   /boot/vmlinuz-2.6.31-20-generic \
          root=UUID=8954fa66-e11f-42dc-91f0-b4aa480fa103 ro   quiet splash
        initrd  /boot/initrd.img-2.6.31-20-generic
}
menuentry "Ubuntu, Linux 2.6.31-20-generic (recovery mode)" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
        insmod ext2
        set root=(hd0,7)
        search --no-floppy --fs-uuid --set 8954fa66-e11f-42dc-91f0-b4aa480fa103
        linux   /boot/vmlinuz-2.6.31-20-generic
         root=UUID=8954fa66-e11f-42dc-91f0-b4aa480fa103 ro single
        initrd  /boot/initrd.img-2.6.31-20-generic
```

```
}
```

Booting GRUB 2 from Grub Legacy

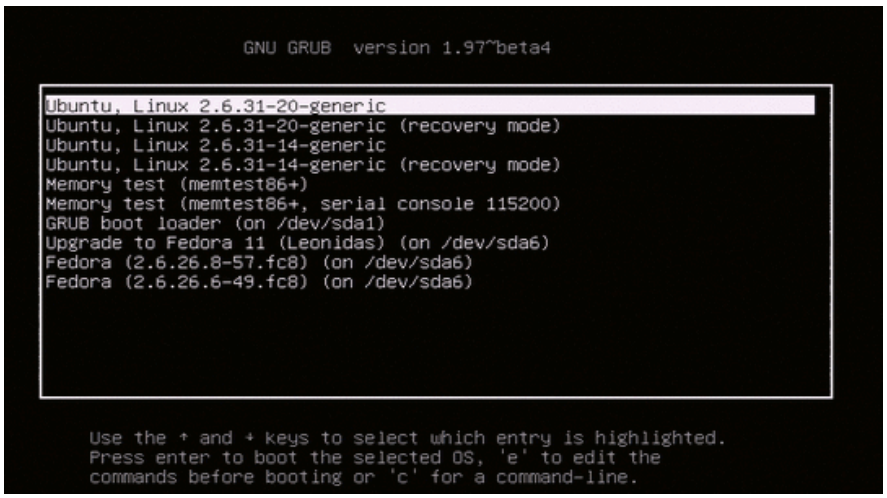
We mentioned that you cannot chain load GRUB 2 from a partition boot record. Once you build your core image and configuration file, add an entry to your GRUB Legacy grub.conf or menu.lst file to boot the GRUB 2 core image which then displays the GRUB 2 menu that you built. Listing 9 shows the entry that we also use in Listing 15. Since this configuration file is for GRUB Legacy, the root entry specifies (hd0,6) for /dev/sda7.

Listing 9. Grub Legacy configuration entry to boot GRUB 2 core image

```
# Ubuntu 9.10 uses GRUB 2, so use kernel to drive GRUB 2 menu.
title           Ubuntu 9.10 (sda7)
root            (hd0,6)
kernel          /boot/grub/core.img
```

When you select this entry from your GRUB Legacy menu, you will see a menu similar to the one in Figure 3, where the first entry is highlighted and it and the following entry are the two illustrated in Listing 8.

Figure 3. GRUB 2 menu example



You might be tempted to try installing GRUB 2 in a partition boot record. If you try you will probably see output similar to that in Listing 10.

Listing 10. Attempting to install GRUB 2 into a partition boot record

```
ian@pinguino:~$ sudo grub-install /dev/sda7
grub-setup: warn: Attempting to install GRUB to a partition instead of the MBR.
This is a BAD idea.
grub-setup: warn: Embedding is not possible.  GRUB can only be installed in
this setup by using blocklists.  However, blocklists are UNRELIABLE and its
use is discouraged.
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(hd0) /dev/sda
```

This explains why my attempt to install GRUB 2 in the /dev/sda7 partition while I was installing Ubuntu 9.10 failed. It is worth noting that this release of Ubuntu will install GRUB 2 for a new installation, but will preserve the existing boot manager if you are upgrading an existing Ubuntu installation.

LILO

LILO, or the LInux LOader, is one of the three most common Linux boot loaders. You can install LILO into the MBR of your bootable hard drive, or into the partition boot record of a partition. You can also install it on removable devices such as floppy disks, CDs, or USB keys. It is a good idea to practice on a floppy disk or USB key if you are not already familiar with LILO, as in our examples.

During Linux installation, you will usually specify a boot manager. If you choose GRUB, then you may not have LILO installed. If you do not have it installed already, then you will need to install the package for it. You may also want to install the lilo-doc package for additional documentation and more examples. From this point on, we will assume that you already have the LILO package installed. See our series roadmap for the articles on package management if you need help.

You configure LILO using a configuration file, which is usually /etc/lilo.conf. You can use the `liloconfig` command (normally found in /usr/sbin) to generate a starting configuration file, and then edit it as needed. The configuration file in Listing 11 was generated in this way. The file is reasonably well annotated, and the man pages for lilo and lilo.conf will give you more help. This is a typical LILO configuration file that might be used on a dual-boot system with Windows and one or more Linux systems.

Listing 11. /etc/lilo.conf example

```
# Originally generated by liloconfig - modified by Ian Shields

# This allows booting from any partition on disks with more than 1024
# cylinders.
lba32

# Specifies the boot device (floppy)
boot=/dev/fd0

# Specifies the device that should be mounted as root.
# If the special name CURRENT is used, the root device is set to the
# device on which the root file system is currently mounted. If the root
# has been changed with  -r , the respective device is used. If the
# variable ROOT is omitted, the root device setting contained in the
# kernel image is used. It can be changed with the rdev program.
root=/dev/sda7

# Bitmap configuration for /boot/coffee.bmp
bitmap=/boot/coffee.bmp
bmp-colors=12,,11,15,,8
bmp-table=385p,100p,1,10
bmp-timer=38,2,13,1

# Enables map compaction:
# Tries to merge read requests for adjacent sectors into a single
# read request. This drastically reduces load time and keeps the map
# smaller. Using COMPACT is especially recommended when booting from a
# floppy disk.
```

```
compact

# Install the specified file as the new boot sector.
# LILO supports built in boot sectors, you only need
# to specify the type, choose one from 'text', 'menu' or 'bitmap'.
# new: install=bmp      old: install=/boot/boot-bmp.b
# new: install=text     old: install=/boot/boot-text.b
# new: install=menu     old: install=/boot/boot-menu.b or boot.b
# default: 'menu' is default, unless you have a bitmap= line
# Note: install=bmp must be used to see the bitmap menu.
# install=menu
install=bmp

# Specifies the number of _tenths_ of a second LILO should
# wait before booting the first image.  LILO
# doesn't wait if DELAY is omitted or if DELAY is set to zero.
# delay=20

# Prompt to use certain image. If prompt is specified without timeout,
# boot will not take place unless you hit RETURN. Timeout is in tenths of
# a second.
prompt
timeout=200

# Enable large memory mode.
large-memory

# Specifies the location of the map file. If MAP is
# omitted, a file /boot/map is used.
map=/boot/map

# Specifies the VGA text mode that should be selected when
# booting. The following values are recognized (case is ignored):
#   NORMAL  select normal 80x25 text mode.
#   EXTENDED  select 80x50 text mode. The word EXTENDED can be
#     abbreviated to EXT.
#   ASK  stop and ask for user input (at boot time).
#   <number>  use the corresponding text mode. A list of available modes
#     can be obtained by booting with  vga=ask  and pressing [Enter].
vga=normal

# Defines non-standard parameters for the specified disk.
#disk=/dev/sda
# bios=0x80

# If you are using removable USB drivers (with mass-storage)
# you will need to tell LILO to not use these devices even
# if defined in /etc/fstab and referenced in /proc/partitions.
# Adjust these lines to your devices:
#
# disk=/dev/sda inaccessible
# disk=/dev/sdb inaccessible

# These images were automagically added. You may need to edit something.

image=/boot/vmlinuz-2.6.31-14-generic
 label="Lin 2.6.31-14"
 initrd=/boot/initrd.img-2.6.31-14-generic
 read-only

image=/boot/vmlinuz-2.6.31-20-generic
 label="Lin 2.6.31-20"
 initrd=/boot/initrd.img-2.6.31-20-generic
 read-only

image=/boot/memtest86+.bin
 label="Memory Test+"
```

```
 read-only

# If you have another OS on this machine (say DOS),
# you can boot if by uncommenting the following lines
# (Of course, change /dev/sda1 to wherever your DOS partition is.)
other=/dev/sda6
        label="Fedora 8"

 other=/dev/sda1
   label="Windows XP"
```

We will test our configuration by creating a boot floppy, so we specified `boot=/dev/fd0`, near the top of the file. We also increased the prompt timeout to 20 seconds (200 increments of one tenth of a second), and added a second "other" entry to chain load to the boot loader on the Fedora 8 partition (/dev/sda6). LILO only allows one `root` command in a configuration file, so it can boot multiple images off the root partition, but needs to chain load to another boot loader to boot an image from a different installation, such as Fedora 8 (/dev/sda6) or Windows XP (/dev/sda1) in our example.

You use the `lilo` command, located in /sbin/lilo, to write a stage 1 boot record and create a map file (/boot/map) using configuration information, such as the above example, normally located in /etc/lilo.conf. There are some auxiliary uses that we will mention later.

Now, if we format a floppy disk with a Linux file system such as ext2, we can run the `lilo` command (/sbin/lilo) to create a bootable floppy disk. Our output is shown in Listing 12. Note that the `lilo` command has five levels of verbosity. Specify an extra -v for each level. This example was created using `lilo -v -v`.

Listing 12. Creating a bootable floppy disk with lilo

```
LILO version 22.8, Copyright (C) 1992-1998 Werner Almesberger
Development beyond version 21 Copyright (C) 1999-2006 John Coffman
Released 19-Feb-2007, and compiled at 10:52:38 on Aug 25 2009
Running Linux kernel 2.6.31-14-generic on i686
Ubuntu

raid_setup returns offset = 00000000  ndisk = 0
 BIOS   VolumeID   Device
Reading boot sector from /dev/fd0
pf_hard_disk_scan: ndevs=1
  0800  54085408  /dev/sda
device codes (user assigned pf) = 0
device codes (user assigned) = 0
device codes (BIOS assigned) = 1
device codes (canonical) = 1
mode = 0x03,  columns = 80,  rows = 25,  page = 0
Using BITMAP secondary loader
Calling map_insert_data
Secondary loader: 19 sectors (0x3800 dataend).
Warning: The boot sector and map file are on different disks.
bios_boot = 0x00  bios_map = 0x80  map==boot = 0  map S/N: 54085408
Mapping bitmap file /boot/coffee.bmp
Calling map_insert_file
Compaction removed 592 BIOS calls.
Bitmap: 603 sectors.
BIOS data check was okay on the last boot

Boot image: /boot/vmlinuz-2.6.31-14-generic
Setup length is 26 sectors.
```

```
Compaction removed 7452 BIOS calls.
Mapped 7601 sectors.
Mapping RAM disk /boot/initrd.img-2.6.31-14-generic
Compaction removed 14696 BIOS calls.
RAM disk: 14930 sectors.
Added Lin_2.6.31-14 *

Boot image: /boot/vmlinuz-2.6.31-20-generic
Setup length is 26 sectors.
Compaction removed 7468 BIOS calls.
Mapped 7617 sectors.
Mapping RAM disk /boot/initrd.img-2.6.31-20-generic
Compaction removed 14704 BIOS calls.
RAM disk: 14938 sectors.
Added Lin_2.6.31-20

Boot image: /boot/memtest86+.bin
Setup length is 4 sectors.
Compaction removed 243 BIOS calls.
Mapped 254 sectors.
Added Memory_Test+

Boot other: /dev/sda6, loader CHAIN
Pseudo partition start: 43198848
Compaction removed 0 BIOS calls.
Mapped 6 (4+1+1) sectors.
Added Fedora_8

Boot other: /dev/sda1, on /dev/sda, loader CHAIN
Compaction removed 0 BIOS calls.
Mapped 6 (4+1+1) sectors.
Added Windows_XP

 BIOS   VolumeID   Device
  80    54085408    0800
Writing boot sector.
/boot/boot.0200 exists - no boot sector backup copy made.
Map file size: 336896 bytes.
RAID device mask 0x0000
One warning was issued.
```

We now have our bootable LILO diskette. If LILO encounters an error, you might see an error message, and the boot sector will not be written. The message will give you some idea of what you need to fix.

If we use the diskette we just created and reboot the machine, we will see a prompt similar to that in Figure 4. Note the countdown timer at the top of the screen, which is counting down from 20 seconds (200 tenths of a second). If no choice is made within 20 seconds, the highlighted choice (the first one, or Lin_2.6.31-14 in this case) is automatically booted.

Figure 4. LILO boot screen

You can use the arrow keys to move up and down the selection list. Figure 5 shows how to select the newer Lin_2.6.31-20 kernel, which is the second entry. Note that the timer value is no longer displayed. Press **Enter** to boot this kernel.

Figure 5. Selecting another boot target with LILO

When you have tested your boot diskette, change the boot=/dev/fd0 entry in your lilo.conf file to install LILO on the MBR or a partition boot record. For example, boot=/dev/sda will install LILO in the master boot record of your first hard drive.

You now have an introduction to LILO and its configuration file. You can override some configuration options from the `lilo` command line. You will find more information in the lilo man page using the command `man lilo`, or `man lilo.conf`. You will find even more extensive

information in the PostScript user guide that is installed with the lilo or lilo-doc package. This should be installed in your documentation directory, but the exact location may vary by system. One way to locate the file is to filter the package list through grep. Listing 13 shows this for the Debian-based Ubuntu 9.10 system that we have been using in this example.

Listing 13. Locating the LILO user guide with dpkg

```
ian@pinguino:~$ dpkg -L lilo-doc | grep "ps"
/usr/share/doc/lilo-doc/user.ps.gz
/usr/share/doc/lilo-doc/tech.ps.gz
```

The corresponding command on an RPM-based system such as Fedora would be: `rpm -ql lilo-doc | grep "ps"`

LILO auxiliary commands

LILO has several auxiliary commands:

lilo -q
: Displays information from the map file.

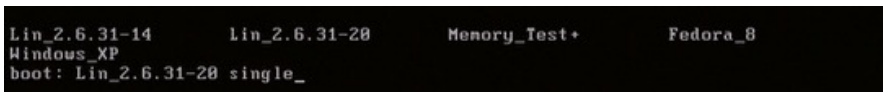
lilo -R
: Sets lilo to automatically boot the specified system on the next reboot only. This is very convenient for automatically rebooting remote systems.

lilo -I
: Displays information about the path of a kernel.

lilo -u
: Uninstalls lilo and restores the previous boot record.

When LILO boots a Linux system, you may want to provide additional parameters at boot time. For example, if your graphical startup was not working, you may want to boot into mode 3 or into single user mode to recover. If you select an entry and then press the **TAB** key, you should be presented with a terminal window that has been prefilled with your selected label name, as shown in Figure 6. Any text you type after the label name will be passed to the kernel. For example, in our example, you would select the latest Ubuntu system by simply moving down a line, then pressing **TAB**. If you don't select a name first, then you may have to type the full name that you specified in the label. You can see why labels that are more succinct than our examples might be good idea: typing Lin_2.6.31.20 isn't easy.

Figure 6. Adding kernel parameters with LILO



In Figure 6, we added "single" to the boot parameters to signify that we want to boot in single user mode to do some kind of system recovery. See the Recovery section for more information on using boot parameters.

Remember also that with LILO you **must** run the `lilo` command whenever you update the configuration file (/etc/lilo.conf). You should also run the `lilo` command if you add, move, or remove partitions or make any other changes that might invalidate the generated boot loader.

System updates

Most distributions provide tools for updating the system. These tools are usually aware of the boot loader in use and will often update your configuration file automatically. If you build your own custom kernel, or prefer to use a configuration file with a non-standard name or location, then you may need to update the configuration file yourself.

- If you use GRUB, you can edit the /boot/grub/grub.conf file to make your changes, and the GRUB stage 2 loader will read the file when you reboot. You do not normally need to reinstall GRUB just because you add a new kernel. However, if you move a partition, or add drives, you may need to reinstall GRUB. Remember the stage 1 loader is very small, so it simply has a list of block addresses for the stage 2 loader. Move the partition and the addressed change, so stage 1 can no longer locate stage 2. We'll cover some recovery strategies and also discuss GRUB's stage 1.5 loaders next.
- If you use GRUB 2, you rebuild the GRUB 2 configuration using the `update-grub` command as described in Building the GRUB 2 configuration file.
- If you use LILO, then you **must** run the `lilo` command whenever you update your configuration file or make changes such as adding a hard drive or deleting a partition.
- If you run more than one Linux system in different partitions, consider using a boot partition as described below.

Recovery

Let's now look at some things that can go wrong with your carefully prepared boot setup, particularly when you install and boot multiple operating systems. The first thing to remember is to resist your initial temptation to panic. Recovery is usually only a few steps away. The strategies here should help you through many types of crises.

Anyone with physical access to a machine has a lot of power. Likewise, anyone with access to a GRUB command line also has access to files on your system without the benefit of any ownership or other security provisions provided by a running system. Keep these points in mind when you select your boot loader. The choice between LILO and GRUB is largely a matter of personal preference. Choose the loader that best suits your particular needs and style of working.

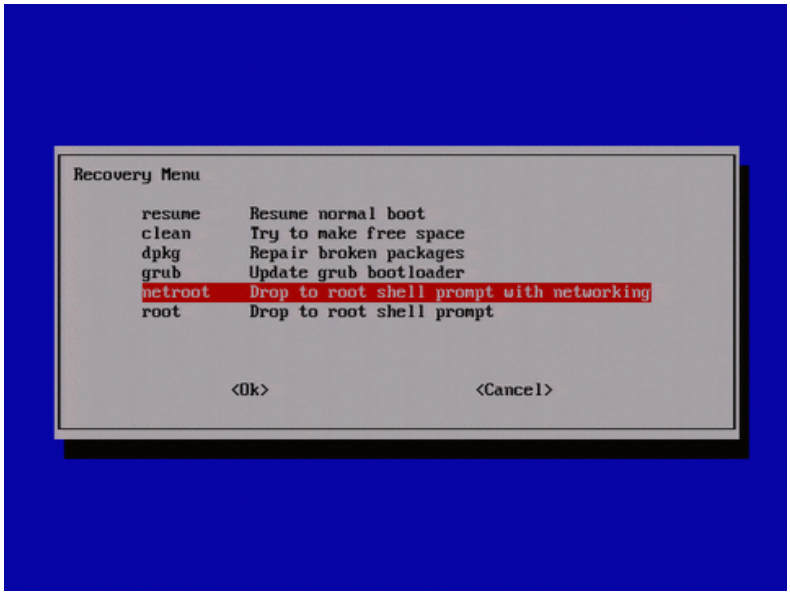
Another install destroys your MBR

Sometimes you will install another operating system and inadvertently overwrite your MBR. Some systems, such as DOS and Windows, always install their own MBR. It is usually very easy to recover from this situation. If you develop a habit of creating a boot floppy every time you run `lilo` or reinstall GRUB, you are home free. Simply boot into your Linux system from the floppy and rerun `lilo` or `grub-install`.

If you don't happen to have a boot floppy, but you still have almost any Linux distribution available, you can usually boot the Linux install media in a recovery mode. When you do so, the root file system on your hard drive will either be mounted at some strange recovery point, or not be mounted at all. You can use the `chroot` command to make this odd mount point become your root (/) directory. Then run `lilo` or `grub-install` to create a new boot floppy or to reinstall the MBR. I prefer to create a floppy and use it to boot, making sure that all is well before I go and rewrite

the MBR, but you may be more courageous than I. Booting a CD or DVD in rescue mode may give you a menu of choices or may simply drop you into single user mode where you have root authority. If you try booting the Ubuntu 9.10 system we used for the LILO examples into single-user mode, you will see a menu like that in Figure 7. The highlighted option drops to a root shell with networking.

Figure 7. Ubuntu 9.10 recovery menu



A menu such as this is a common choice with rescue media. For our rescue example, we do not use a CD, but we boot the Ubuntu 9.19 system into single user mode as described above in the section on LILO. We then walk through the typical steps needed to create a boot diskette for another system, in this case the Fedora 8 system on /dev/sda6. Whether you boot from a rescue CD or another OS on the same system, the steps will be similar, although some (such as mounting the image to be rescued) may already be done for you.

When you eventually reach a root prompt, you will want to do some or all of the things shown in Listing 14. For our example, we create a mount point (/mnt/sysimage), mount /dev/sda6, mount some special file systems in the new (to-be) root, and then chroot into the mounted system. The `cat /etc/issue` commands show the effect of the `chroot` command, which makes the target directory the new root of the file system, at least until we drop out of the chrooted environment with Ctrl-d or `exit`.

Listing 14 shows an example in the same environment we used for our earlier configuration examples. Think of this as a terminal window with you logged in as root. In other words, be very careful what you write to your hard drive.

Listing 14. Using a rescue system and chroot

```
root@pinguino:~# mkdir /mnt/sysimage
root@pinguino:~# mount /dev/sda6 /mnt/sysimage
root@pinguino:~# cd /mnt/sysimage
root@pinguino:~# mount -t proc none proc/
```

```
root@pinguino:~# mount -t sysfs none sys/
root@pinguino:~# mount -o bind /dev dev/
root@pinguino:~# cat /etc/issue
Ubuntu 9.10 \n \l

root@pinguino:~# chroot /mnt/sysimage
root@pinguino:~#
sh-3.00# chroot /mnt/sysimage
[root@oinguine /]# cat /etc/issue
Fedora release 8 (Werewolf)
Kernel \r on an \m

[root@oinguine /]# grub-install /dev/fd0
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)   /dev/fd0
(hd0)   /dev/sda
[root@oinguine /]#
```

Once you have your bootable floppy, press Ctrl-d to exit from the chroot environment and then reboot your system, remembering to remove any installation CD or recovery media that you may have used. If you don't happen to have an installation CD or DVD handy, there are many recovery and live LINUX CDs available online and some diskette or USB memory key ones too. See Resources for details.

Although beyond the scope of this article, you may want to know that it is possible to have your MBR boot a Windows 2000 or Windows XP system and install LILO or GRUB on a partition boot record. The ntldr boot program can also chain load other boot sectors, although setup can be a little tricky. You will need to copy the boot sector to a Windows partition (typically using the `dd` command) and modify the hidden boot.ini file to make this work.

You moved a partition

If you moved a partition and forgot about your boot setup, you have a temporary problem. Typically, LILO or GRUB refuse to load. LILO will probably print an 'L' indicating that stage 1 was loaded and then stop. GRUB will give you an error message. What has happened here is that the stage 1 loader, which had a list of sectors to load to get to the stage 2 loader, can perhaps load the sectors from the addresses it has, but the sectors no longer have the stage 2 signature. If you built a boot diskette using the methods outlined earlier, remember that all that `lilo` or `grub-install` put on the diskette was a single boot sector, so your boot diskette probably won't help. As in the previous example, you will probably need to boot some kind of rescue environment and rebuild your boot floppy with LILO or GRUB. Then reboot, check your system and reinstall your boot loader in the MBR.

You may have noticed that our configuration examples used some labels and (Universally Unique IDs) UUIDs for partitions. For example:

```
kernel /boot/vmlinuz-2.6.26.8-57.fc8 ro root=LABEL=FEDORA8 rhgb quiet
```

or

```
kernel /boot/upgrade/vmlinuz preupgrade repo=hd::/var/cache/yum/preupgrade stage2=\
hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/install.img \
```

```
ks=hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/ks.cfg
```

I often use labels UUIDs like this to help avoid problems when I move partitions. You still need to update the GRUB or LILO configuration file and rerun `lilo`, but you don't have to update /etc/fstab as well. This is particularly handy if you create a partition image on one system and restore it at a different location on another system. It's also handy if you boot from a drive, such as a USB drive, that may not always be attached at the same location.

Using a boot partition

Another approach to recovery, or perhaps avoiding it, is to use a separate partition for booting. This is particularly useful if you have a test system with several distributions on it that you may rebuild frequently. The boot partition need not be very large, perhaps 100MB or so. Put this partition somewhere where it is unlikely to be moved and where it is unlikely to have its partition number moved by the addition or removal of another partition. In a mixed Windows and Linux environment, /dev/sda2 (or /dev/hda2 depending on how your disks are labeled) is often a good choice for your boot partition. Listing 15 shows the entries in the small boot partition (/dev/sda2) that is actually loaded by the master boot record on the system we have been using as an example.

Listing 15. Using a small boot partition with GRUB

```
# 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/.

## default num
# Set the default entry to the entry number NUM. Numbering starts from 0, and
# the entry number 0 is the default if the command is not used.
#
# You can specify 'saved' instead of a number. In this case, the default entry
# is the entry saved with the command 'savedefault'.
default  0

## timeout sec
# Set a timeout, in SEC seconds, before automatically booting the default entry
# (normally the first entry defined).
timeout  10

## password ['--md5'] passwd
# If used in the first section of a menu file, disable all interactive editing
# control (menu entry editor and command-line)  and entries protected by the
# command 'lock'
# e.g. password topsecret
#      password --md5 $1$gLhU0/$aW78kHK1QfV3P2b2znUoe/
# password topsecret

title           Fedora 8 (sda6) Grub menu
root            (hd0,5)
chainloader     +1
boot

# Ubuntu 9.10 uses GRUB 2, so use kernel to drive GRUB 2 menu.
title           Ubuntu 9.10 (sda7)
root            (hd0,6)
kernel          /boot/grub/core.img

# This is a divider, added to separate the menu items below from the Debian
```

```
# ones.
title  Other operating systems:
root

# Windows XP
title  Windows NT/XP
root  (hd0,0)
savedefault
chainloader +1
```

Another reason for having a boot partition arises when your root partition uses a file system not supported by your boot loader. For example, it is very common to have a /boot partition formatted ext2 or ext3 when the root partition (/) uses LVM, which is not supported by GRUB Legacy.

If you have multiple distributions on your system, **do not** share the /boot partition between them. Remember to set up LILO or GRUB to boot from the partition that will later be mounted as /boot. Remember also that the update programs for a distribution will usually update the GRUB or LILO configuration for that system. In an environment with multiple systems, you may want to keep one with its own /boot partition as the main one and manually update that configuration file whenever an update of one of your systems requires it. Another approach is to have each system install a boot loader into its own partition boot record and have your main system simply chain load the partition boot records for the individual systems, giving you a two-stage menu process.

Building a self-contained boot floppy or CD-ROM

Finally, let's look more closely at the GRUB setup and how to make a standalone floppy or CD that will get you to a GRUB prompt, no matter what has happened to your hard drive.

If you are familiar with hard drive layout as discussed in the article "Learn Linux 101: Hard disk layout," you will remember all that stuff about cylinders on hard drives. Even though you might think of a cylinder as a fictional entity with modern drives, many aspects of your file system have not forgotten them. In particular, you will find partitions use an integral number of cylinders aligned on cylinder boundaries. Within a partition, many file systems also manage space in units of cylinders. On many UNIX® and Linux systems, the layout of the file system is stored in a *superblock*, which is the first allocation unit in the file system. For systems such as ext2 or ext3 file systems and reasonably large hard drives, the space will be broken into several sections with a copy of the superblock in the beginning of each section. This will help with recovery if you accidentally mess up partition boundaries with a program like fdisk.

One other benefit of the cylinder mentality is that there is some space at the beginning of a disk right after the MBR. GRUB takes advantage of this by embedding a stage 1.5 boot loader in this space or in similar otherwise unused space on a partition whenever possible. The stage 1.5 loader understands the file system on the partition that contains the stage 2, so it is somewhat more protected against problems associated with files being moved.

That is all well and good, but how does it relate to a bootable floppy? Well, a floppy doesn't have much space or much notion of cylinders, so if you want to boot both stage 1 and stage 2 of GRUB from a floppy, you need to install stage 1 and then copy stage 2 to the sectors immediately following the boot sector. Listing 16 shows an example of how to do this. Use an empty diskette

as this process will destroy data. You should copy the files that came with your GRUB distribution rather than the ones from your /boot/grub directory, as /boot/grub/stage2 has been modified to work with your hard drive partitions. You should find the original stage1 and stage2 files in a subdirectory of /usr/share/grub. In our example they are in /usr/share/grub/i386-redhat.

Listing 16. Creating a GRUB boot floppy

```
[root@pinguino ~]# ls /usr/share/grub/
i386-redhat
[root@pinguino ~]# cd /usr/share/grub/i386-redhat/
[root@pinguino i386-redhat]# ls -l st*
-rw-r--r-- 1 root root    512 2008-05-28 12:05 stage1
-rw-r--r-- 1 root root 109212 2008-05-28 12:05 stage2
-rw-r--r-- 1 root root 109212 2008-05-28 12:05 stage2_eltorito
[root@pinguino i386-redhat]# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.29007 s, 1.8 kB/s
[root@pinguino i386-redhat]# dd if=stage2 of=/dev/fd0 bs=512 seek=1
213+1 records in
213+1 records out
109212 bytes (109 kB) copied, 7.4094 s, 14.7 kB/s


[root@lyrebird root]# ls /usr/share/grub
i386-redhat
[root@lyrebird root]# cd /usr/share/grub/i386-redhat
[root@lyrebird i386-redhat]# ls -l st*
-rw-r--r--    1 root     root          512 Aug  3  2004 stage1
-rw-r--r--    1 root     root       104092 Aug  3  2004 stage2
[root@lyrebird i386-redhat]# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
[root@lyrebird i386-redhat]# dd if=stage2 of=/dev/fd0 bs=512 seek=1

203+1 records in
203+1 records out
```

If you started with a MS-DOS formatted floppy before you did this, and you now attempt to mount the floppy, the mount command will give you an error. Copying the stage2 right after the diskette's boot sector (seek=1) destroyed the file system on your diskette.

Creating a bootable GRUB CD is somewhat similar, although you prepare the CD image on your hard drive. You will need a temporary directory, say grubcd, with subdirectories boot and boot/grub. You then need to copy the stage2_eltorito file from your GRUB distribution files to the grub subdirectory that you just created. Then use `genisoimage` to create a bootable .iso image file that you can burn to CD with your favorite burning tool. Listing 17 shows how to create the CD image as grubcd.iso.

Listing 17. Creating a GRUB bootable CD image

```
[ian@pinguino ~]$ mkdir -p grubcd/boot/grub
[ian@pinguino ~]$ cp /usr/share/grub/i386-redhat/stage2_eltorito grubcd/boot/grub
[ian@pinguino ~]$ genisoimage -R -b boot/grub/stage2_eltorito -no-emul-boot \
> -boot-load-size 4 -boot-info-table -o grubcd.iso grubcd
I: -input-charset not specified, using utf-8 (detected in locale settings)
Size of boot image is 4 sectors -> No emulation
Total translation table size: 2048
Total rockridge attributes bytes: 760
Total directory bytes: 4576
Path table size(bytes): 34
Max brk space used 0
233 extents written (0 MB)
```

You could boot the CD or diskette in an arbitrary PC; it does not have to be one with a Linux system on it. If you now boot the diskette, you will notice a delay while it loads stage 2 from the diskette. Similarly, if you boot the CD, it will load the GRUB shell from the CD. When you boot either, you will get a GRUB boot prompt. Press the tab key to see a list of commands available to you. Try `help commandname` to get help on the command called *commandname*. Listing 18 illustrates the GRUB command line.

Listing 18. The GRUB command line

```
    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>
 Possible commands are: blocklist boot cat chainloader clear cmp color configfile
 debug device displayapm displaymem dump embed find fstest geometry halt help
 hide impsprobe initrd install ioprobe kernel lock makeactive map md5crypt
 module modulenounzip pager partnew parttype password pause quit read reboot
 root rootnoverify savedefault serial setkey setup terminal terminfo testload testvbe
 unhide uppermem vbeprobe

grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
    Similar to `root', but don't attempt to mount the partition. This
    is useful for when an OS is outside of the area of the disk that
    GRUB can read, but setting the correct root device is still
    desired. Note that the items mentioned in `root' which derived
    from attempting the mount will NOT work correctly.

grub> find /boot/grub/grub.conf
 (hd0,1)
 (hd0,5)

grub>
```

In this example, we have found that there are GRUB configuration files on two different partitions on our first hard drive. We could load the GRUB menu from one of these using the `configfile` command. For example: `configfile (hd0,1)/boot/grub/grub.conf`

This would load the menu for that configuration file and we might be able to boot the system from this point. You can explore these grub commands in the GRUB manual. Try typing `info grub` in a Linux terminal window to open the manual.

One last point before we leave GRUB. We mentioned that the stage 2 GRUB file destroyed the file system on the diskette. If you want a bootable GRUB recovery diskette that loads GRUB files, including a configuration file from the diskette you can accomplish this using the following steps:

1. Use the `mkdosfs` command to create a DOS FAT file system on the diskette and use the `-R` option to reserve enough sectors for the stage 2 file.
2. Mount the diskette
3. Create a /boot/grub directory on the diskette
4. Copy the GRUB stage1, stage2, and grub.conf files to the boot/grub directory on the diskette. Copy your splash image file too if you want one.
5. Edit your grub.conf file on the diskette so it refers to the splash file on the diskette.
6. Unmount the diskette
7. Use the `grub` command shell to setup GRUB on the diskette using the GRUB root and setup commands.

We illustrate this in Listing 19.

Listing 19. Installing GRUB on diskette with a file system

```
[root@pinguino ~]# mkdosfs -R 220 /dev/fd0
mkdosfs 2.11 (12 Mar 2005)
[root@pinguino ~]# mkdir /mnt/floppy
[root@pinguino ~]# mount /dev/fd0 /mnt/floppy
[root@pinguino ~]# mkdir -p /mnt/floppy/boot/grub
[root@pinguino ~]# cp /boot/grub/stage1 /mnt/floppy/boot/grub
[root@pinguino ~]# cp /boot/grub/stage2 /mnt/floppy/boot/grub
[root@pinguino ~]# cp /boot/grub/splash* /mnt/floppy/boot/grub
[root@pinguino ~]# cp /boot/grub/grub.conf /mnt/floppy/boot/grub
[root@pinguino ~]# umount /dev/fd0
[root@pinguino ~]# grub
Probing devices to guess BIOS drives. This may take a long time.

    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> root (fd0)
 Filesystem type is fat, using whole disk

grub> setup (fd0)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/fat_stage1_5" exists... no
 Running "install /boot/grub/stage1 (fd0) /boot/grub/stage2 p /boot/grub/grub.c
onf "... succeeded
Done.
```

This example also illustrates the fact that you can run the `grub` command in a Linux shell. This is a good way to learn about GRUB without having to reboot your machine all the time. You will need root authority to run `grub` and you will be able to do most things GRUB can do, except actually reboot the system.

Rescue media for GRUB 2

GRUB 2 comes with a `grub-mkrescue` command to help you create a rescue floppy or CD image. Recent versions of `grub-mkrescue` use the xorriso package rather than the mkisofs package to create the ISO image, so you might need to install if your first attempt fails with a message that tells you something like: "xorriso: command not found". Listing 20 shows how to create a GRUB 2 rescue image in file rescue.iso. You don't have to be root to create the rescue ISO. We use root here to show you how to create a bootable USB flash drive and you need root privileges for that.

Listing 20. Creating a GRUB2 rescue image

```
[root@echidna ~]# /usr/bin/grub2-mkrescue -o rescue.iso
Enabling BIOS support ...
xorriso 1.2.4 : RockRidge filesystem manipulator, libburnia project.

Drive current: -outdev 'stdio:rescue.iso'
Media current: stdio file, overwriteable
Media status : is blank
Media summary: 0 sessions, 0 data blocks, 0 data, 7177m free
Added to ISO image: directory '/'='/tmp/tmp.Dw4KSbpoIx'
xorriso : UPDATE : 196 files added in 1 seconds
xorriso : UPDATE : 196 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 29191 bytes from file '/tmp/tmp.LepCeiJPZM'
ISO image produced: 1094 sectors
Written to medium : 1094 sectors at LBA 0
Writing to 'stdio:rescue.iso' completed successfully.
```

Once you have created the ISO image you can burn it to a CD (or DVD). If you prefer you can also copy it to a USB flash drive and boot from that assuming your BIOS supports booting from such devices. Listing 21 shows how to use the `dd` command to copy the ISO image to the USB flash drive /dev/sdc.

Warning: Make absolutely sure that you copy the image to the correct device. Copying it to the wrong device could destroy a lot of your data.

Listing 21. Writing a GRUB2 rescue image to a USB flash drive

```
[root@echidna ~]# dd if=rescue.iso of=/dev/sdc
4376+0 records in
4376+0 records out
2240512 bytes (2.2 MB) copied, 0.625654 s, 3.6 MB/s
```

You should now have a bootable USB flash drive that will boot to a GRUB 2 prompt. You can explore these GRUB 2 commands in the GRUB manual. Try typing `info grub2` in a Linux terminal window to open the manual.

Summary

You have now learned about the main boot loaders for traditional Linux systems, including how to recover from mistakes and boot problems.

Downloads

Description	Name	Size
Configuration files used in this article	l-lpic1-v3-102-2-samples.zip	8KB

Resources

Learn

- Use the developerWorks roadmap for LPIC-1 to find the developerWorks articles to help you study for LPIC-1 certification based on the April 2009 objectives.
- At the LPIC Program site, find detailed objectives, task lists, and sample questions for the three levels of the Linux Professional Institute's Linux system administration certification. In particular, see their April 2009 objectives for LPI exam 101 and LPI exam 102. Always refer to the LPIC Program site for the latest objectives. Note the Exam 101: Objective Changes as of July 2, 2012.
- Review the entire LPI exam prep series on developerWorks to learn Linux fundamentals and prepare for system administrator certification based on earlier LPI exam objectives prior to April 2009.
- Learn about GRUB legacy and GRUB 2 at the GNU GRUB home page. GNU GRUB is a multiboot boot loader derived from GRUB, GRand Unified Bootloader.
- The Multiboot Specification for an interface allows any complying boot loader to load any complying operating system.
- In "Basic tasks for new Linux developers" (developerWorks, April 2011), learn how to open a terminal window or shell prompt and much more.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth other resources for Linux developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.

Get products and technologies

- Download the System rescue CD-Rom, one of many tools available online to help you recover a system after a crash.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the developerWorks Community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Ian Shields

Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. Learn more about Ian in Ian's profile on developerWorks Community.