http://www.ibm.com/developerworks/linux/library/l-bootload.html

What is a boot loader?

Most simply, a *boot loader* loads the operating system. When your machine loads its operating system, the BIOS reads the first 512 bytes of your bootable media (which is known as the *master boot record*, or *MBR*). You can store the boot record of only one operating system in a single MBR, so a problem becomes apparent when you require multiple operating systems. Hence the need for more flexible boot loaders.

The master boot record itself holds two things -- either some of or all of the boot loader program and the partition table (which holds information regarding how the rest of the media is split up into partitions). When the BIOS loads, it looks for data stored in the first sector of the hard drive, the MBR; using the data stored in the MBR, the BIOS activates the boot loader.

Due to the very small amount of data the BIOS can access, most boot loaders load in two stages. In the first stage of the boot, the BIOS loads a part of the boot loader known as the *initial program loader*, or *IPL*. The IPL interrogates the partition table and subsequently is able to load data wherever it may exist on the various media. This action is used initially to locate the second stage boot loader, which holds the remainder of the loader.

The second stage boot loader is the real meat of the boot loader; many consider it the only real part of the boot loader. This contains the more disk-intensive parts of the loader, such as user interfaces and kernel loaders. These user interfaces can range from a simple command line to the all-singing, all-dancing GUIs.

Boot loaders are usually configured in one of two ways: either as a primary boot loader or as a secondary boot loader. *Primary boot loaders* are where the first stage of the boot loader is installed on the MBR (per the previous description). *Secondary boot loaders* are where the first stage of the boot loader is installed onto a bootable partition. A separate boot loader must then be installed into the MBR and configured to pass control to the secondary boot loade

LILO

LInux LOader, or *LILO*, comes as standard on all distributions of Linux. As one of the older/oldest Linux boot loaders, its continued strong Linux community support has enabled it to evolve over time and stay viable as a usable modern-day boot loader. Some new functionality includes an enhanced user interface and exploitation of new BIOS functions that eliminate the old 1024-cylinder limit.

Although LILO continues to be developed, the basic principles of how LILO works still remain the same.

Making LILO your boot loader

What you will need to do to use LILO as your boot loader depends on whether you are installing the OS fresh or have already installed Linux and are planning on moving to LILO. If you're starting fresh, you can jump straight to the Configuring LILO section. If you already have a Linux distribution installed, you usually get an option to install and configure LILO (and can boot your machine into your new Linux install).

For existing Linux users migrating to LILO, first you have to acquire the latest version of LILO (see Resources). Before doing anything else, I advise you to make sure you have a Linux boot disk handy -- it makes life a lot easier if you accidentally mess something up and would like to be able to get back into your original Linux configuration! Once you have LILO on your system, making it take over your MBR is very easy. As the root user, type:

```
# /sbin/lilo -v -v
```

This will use the current LILO defaults and splat anything that is currently in the MBR. However, read about Configuring LILO to make sure you are able to boot up as expected. Also note that if you want to run Windows and Linux on a single machine, you should install your Windows OS first and then the Linux OS, so that the boot loader you choose in the Linux install won't be written over by the Windows boot loader. Unlike the Linux boot loaders, the majority of Windows boot loaders will not allow you to load Linux. If you've already installed Linux first, don't fret; just create yourself a Linux boot disk so that after you have installed Windows, you can get back into your Linux install and overwrite the MBR.

Configuring LILO

LILO configuration is all done through a configuration file located in /etc/lilo.conf. Listing 1 shows an example configuration, relating to my home machine, for dual booting a Linux and Windows machine. You can visualize how this configuration relates to an actual machine by looking at my basic workstation setup:

- Primary HDD (physical disk 1) with Windows XP installed (initially all there was on the machine). In Linux terms, this HDD is /dev/hda (hd0,0 in GRUB terms).
- Secondary HDD (physical disk 2) with Red Hat Linux installed; the root partition is on the third partition of this hard drive, /dev/hdb3 (hd1,2 in GRUB terms).

Listing 1. Example lilo.conf file

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=100
compact
default=Linux
image=/boot/vmlinuz-2.4.18-14
        label=Linux
        root=/dev/hdb3
        read-only
        password=linux
other=/dev/hda
        label=WindowsXP
```

The options used in Listing 1 are:

- The `boot=` line tells LILO where to install the boot loader. In the previous example, this will install it to the MBR of first hard disk. You could alternatively install LILO in /dev/hdb3 (the Linux partition in the example), which would then require you to install another boot loader into /dev/hda that points it to the LILO boot loader; then you just let LILO act as a secondary boot loader. In general, /dev/hda is the most common place for your boot loader to reside. You can also make a LILO floppy boot disk by pointing this parameter to the floppy drive, most

commonly /dev/fd0.

- `map=` points to the map file used by LILO internally during bootup. When you install LILO using the `/sbin/lilo` command, it automatically generates this file, which holds the descriptor table (among other things). My advice is to leave this as it is!
- `install=` is one of the files used internally by LILO during the boot process. This holds both the primary and secondary parts of the boot loader. A segment of this boot.b file is written to the MBR (the primary part of the boot loader), which then points to the map and subsequently points to the secondary boot loader. Again, leave this as it is!
- `prompt=` tells LILO to use the user interface (giving you in this example two selections -- Linux and WindowsXP). In addition using the prompt/user interface, you get the option to specify specific parameters for the Linux kernel or others if appropriate. If you do not specify this option in the configuration file, LILO will boot into the default OS with no user interaction and no waiting. (It's worth noting, though, that if you hold the SHIFT key down during boot, you can get the prompt up anyway, which is quite useful if you don't want the average Joe to be exposed to the boot loader).
- `timeout=` is the number of tenths of a second that the boot prompt will wait before automatically loading the default OS, in this case Linux. If `prompt` is not specified in the lilo.conf, this parameter is ignored.
- The `compact` option magically makes the boot process quicker by merging adjacent disk read requests into a single request. It can be a mixed blessing, though, as I've seen a number of posts on forums regarding issues with this option. This option especially useful if you wish to boot from a floppy.
- The `default=` option tells LILO which image to boot from by default, such as after the timeout period. This relates to a label of one of the images in the lilo.conf file. If you don't specify this option in the configuration file, it will boot the first image specified in the file.
- For each version of Linux you want to make available for users to boot into, you should specify `image=` and the following three options. The image option specifies the kernel version you wish to boot to.
- `label=` identifies the different OS you want to boot from at the user interface at runtime. In addition, this label is used for specifying the default OS to boot from. (Note: Avoid spaces in the label name; otherwise, you will get an unexpected error when loading the file.)
- The `root=` option tells LILO where the OS file system actually lives. In our example, it is /dev/hdb3, which is the third partition of the second disk.
- `read-only` tells LILO to perform the initial boot to the file system read only. Once the OS is fully booted, it is mounted read-write.
- The `password=` option allows you to set a password for the specific OS you are booting into. In the example this password is held in the lilo.conf file as readable text, so is easily accessible for all to read. Alternatively if you set password="" you can set the password when the bootloader is installed. These can be set on each of the operating systems you wish to boot from if required (in our example we only set a password on the Linux boot).
- `other=` acts like a combination of the image and root options, but for operating systems other than Linux. In our example, it tells LILO where to find the Windows OS, which resides on the first disk in the first partition. This will usually be the case if you have installed Windows first, then Linux.
- `label=` is the same as all other label options.

You can use many other parameters in the lilo.conf file, but the parameters in Listing 1 should get you

into a fairly usable state on your machine. For further information on these and other lilo.conf parameters, refer to the manual pages (man lilo.conf). Since lilo.conf is not read at boot time, the MBR needs to be "refreshed" when this is changed. If you do not do this upon rebooting, none of your changes to lilo.conf will be reflected at startup. Like getting LILO into the MBR in the first place, you need to run:

```
$ /sbin/lilo -v -v
```

The `-v -v` flags give you very verbose output. There are a fair number of parameters you can specify when running LILO like we did. See the manual pages for further information (man lilo).

The initial boot process

When LILO initially loads, it brings up in order each of the letters -- *L-I-L-O*. If all the letters come up, the first stage boot was successful. Anything less indicates a problem:

- L: The first stage boot loader has been loaded. If LILO stops here, there were problems loading the second stage boot loader. This is usually accompanied by an error code. The common problems at this stage are media problems or incorrect disk parameters specified in your lilo.conf file.
- LI: The second stage boot loader has been loaded. LILO halting at this point indicates the second stage boot loader could not be executed. Again, this can be due to problems similar to just L: loading or if the boot.b file has been corrupted, moved, or deleted.
- LIL: The second stage boot loader has now been executed. At this point, media problem could again be responsible or the map file (as specified in the lilo.conf file) could have had problems finding the descriptor tables.
- LIL?: Loaded to the same point as above. This usually means the second stage boot loader loaded at an incorrect address, caused most likely by boot.b being in a different place than specified in the lilo.conf file.
- LIL-: Loaded to the same point as above. Problem loading the descriptor table, most likely due to a corrupt descriptor table.
- LILO: LILO has successfully loaded with no errors.

Additional configuration at boot time

Once LILO has successfully loaded, you will see a LILO prompt. Still using the example lilo.conf file as before, at this point you have two choices, which may not be immediately obvious to LILO newbies. First, you may let LILO time out (after 10 seconds), which will boot /dev/hdb3, the Linux partition. Second, you can press the TAB key, which will list a selection of operating systems to boot from. In our example lilo.conf, we would get "Linux" and "WindowsXP" as our options. Typing either of these will load up that OS. Specifically loading the Linux option will then prompt you to enter a password, which in this case is `linux`. Incorrectly entering the password will take you back to the LILO prompt.

A final word of advice when trying out LILO for the first time: I found it a lot safer to work out my LILO configuration using a floppy boot disk rather than my hard disk. To do this, you must replace the `boot=/dev/hda` with `boot=/dev/fd0` in the lilo.conf file. That way, if I messed up any of the configuration in my lilo.conf file, I could take out the boot disk and boot into Linux as before. Once I was happy everything booted fine using the floppy disk, I then changed my lilo.conf back to use `boot=/dev/hda` and ran `/sbin/lilo` a final time to upload my changes.

GRUB vs. LILO

As stated at the start of this article, all boot loaders work in a similar way to fulfill a common purpose. But LILO and GRUB do have a number of differences:

- LILO has no interactive command interface, whereas GRUB does.
- LILO does not support booting from a network, whereas GRUB does.
- LILO stores information regarding the location of the operating systems it can to load physically on the MBR. If you change your LILO config file, you have to rewrite the LILO stage one boot loader to the MBR. Compared with GRUB, this is a much more risky option since a misconfigured MBR could leave the system unbootable. With GRUB, if the configuration file is configured incorrectly, it will simply default to the GRUB command-line interface.