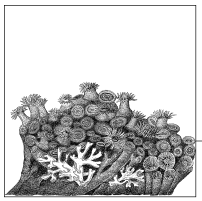

5
Installing and Booting from a Kernel

Previous chapters showed you how to download and build your kernel. Now that you have an executable file—along with any modules you built—it is time to install the kernel and attempt to boot it. In this chapter, unlike earlier ones, all of the commands need to be run as the root user. This can be done by prefixing each command with *sudo*, by using the *su* command to become *root*, or actually by logging in as *root*.

To see whether you have *sudo* installed and the proper access set up, do the following:

```
$ sudo ls ~/linux/linux-2.6.17.11/Makefile
Password:
Makefile
```

Enter either your own password at the password prompt, or the password of the system administrator (root). The choice depends on how the *sudo* command is set up. If this is successful, and you see the line containing:

```
Makefile
```

then you can skip to the next section.

If *sudo* is not installed or giving you the proper rights, try using the *su* command:

```
$ su
Password:
# exit
exit
$
```

At the password prompt, enter the password of the system administrator (*root*). When the *su* program successfully accepts the password, you are transferred to running everything with full root privileges. Be very careful while as *root*, and do only the minimum needed; then exit the program to continue back as your normal user account.

Using a Distribution's Installation Scripts

Almost all distributions come with a script called *installkernel* that can be used by the kernel build system to automatically install a built kernel into the proper location and modify the bootloader so that nothing extra needs to be done by the developer.*

Distributions that offer *installkernel* usually put it in a package called *mkinitrd*, so try to install that package if you cannot find the script on your machine.

If you have built any modules and want to use use this method to install a kernel, first enter:

```
# make modules_install
```

This will install all the modules that you have built and place them in the proper location in the filesystem for the new kernel to properly find. Modules are placed in the */lib/modules/kernel_version* directory, where *kernel_version* is the kernel version of the new kernel you have just built.

After the modules have been successfully installed, the main kernel image must be installed:

```
# make install
```

This will kick off the following process:

1. The kernel build system will verify that the kernel has been successfully built properly.
2. The build system will install the static kernel portion into the */boot* directory and name this executable file based on the kernel version of the built kernel.
3. Any needed initial ramdisk images will be automatically created, using the modules that have just been installed during the *modules_install* phase.
4. The bootloader program will be properly notified that a new kernel is present, and it will be added to the appropriate menu so the user can select it the next time the machine is booted.
5. After this is finished, the kernel is successfully installed, and you can safely reboot and try out your new kernel image. Note that this installation does not overwrite any older kernel images, so if there is a problem with your new kernel image, the old kernel can be selected at boot time.

* Notable exceptions to this rule are Gentoo and other "from scratch" types distributions, which expect users to know how to install kernels on their own. These types of distributions include documentation on how to install a new kernel, so consult it for the exact method required.

Installing by Hand

If your distribution does not have a *installkernel* command, or you wish to just do the work by hand to understand the steps involved, here they are:

The modules must be installed:

```
# make modules_install
```

The static kernel image must be copied into the */boot* directory. For an i386-based kernel, do the following:

```
# make kernelversion
2.6.17.11
```

Note that the kernel version will probably be different for your kernel. Use this value in place of the text `KERNEL_VERSION` in the following steps:

```
# cp arch/i386/boot/bzImage /boot/bzImage-KERNEL_VERSION
# cp System.map /boot/System.map-KERNEL_VERSION
```

Modify the bootloader so it knows about the new kernel. This involves editing a configuration file for the bootloader you use, and is covered later in "Modifying the Bootloader for the New Kernel" for the GRUB and LILO bootloaders.

If the boot process does not work properly, it's usually because an initial ramdisk image is needed. To create this properly, use the steps in the beginning of this chapter for installing a kernel automatically, because the distribution install scripts know how to properly create the ramdisk using the needed scripts and tools. Because each distribution does this differently, it is beyond the scope of this book to cover all of the different methods of building the ramdisk image.

Here is a handy script that can be used to install the kernel automatically instead of having to type the previous commands all the time:

```
#!/bin/sh
#
# installs a kernel
#
make modules_install

# find out what kernel version this is
for TAG in VERSION PATCHLEVEL SUBLEVEL EXTRAVERSION ; do
    eval `sed -ne "/^$TAG/s/ //gp" Makefile`
done
SRC_RELEASE=$VERSION.$PATCHLEVEL.$SUBLEVEL$EXTRAVERSION

# figure out the architecture
ARCH=`grep "CONFIG_ARCH " include/linux/autoconf.h | cut -f 2 -d "\""`

# copy the kernel image
cp arch/$ARCH/boot/bzImage /boot/bzImage-"$SRC_RELEASE"

# copy the System.map file
cp System.map /boot/System.map-"$SRC_RELEASE"

echo "Installed $SRC_RELEASE for $ARCH"
```

Modifying the Bootloader for the New Kernel

There are two common Linux kernel bootloaders: GRUB and LILO. GRUB is the one more commonly used in modern distributions, and does some things a little more easily than LILO, but LILO is still seen as well. We'll cover both in this section.

To determine which bootloader your system uses, look in the */boot/* directory. If there is a *grub* subdirectory:

```
$ ls -F /boot | grep grub
grub/
```

then you are using the GRUB program to boot with. If this directory is not present, look for the presence of the */etc/lilo.conf* file:

```
$ ls /etc/lilo.conf
/etc/lilo.conf
```

If this is present, you are using the LILO program to boot with.

The steps involved in adding a new kernel to each of these programs are different, so follow only the section that corresponds to the program you are using.

GRUB

To let GRUB know that a new kernel is present, all you need to do is modify the */boot/grub/menu.lst* file. For full details on the structure of this file, and all of the different options available, please see the GRUB info pages:

```
$ info grub
```

The easiest way to add a new kernel entry to the */boot/grub/menu.lst* file is to copy an existing entry. For example, consider the following *menu.lst* file from a Gentoo system:

```
timeout 300
default 0

splashimage=(hd0,0)/grub/splash.xpm.gz

title 2.6.16.11
    root (hd0,0)
    kernel /bzImage-2.6.16.11 root=/dev/sda2 vga=0x0305

title 2.6.16
    root (hd0,0)
    kernel /bzImage-2.6.16 root=/dev/sda2 vga=0x0305
```

The line starting with the word `title` defines a new kernel entry, so this file contains two entries. Simply copy one block of lines beginning with the `title` line, such as:

```
title 2.6.16.11
    root (hd0,0)
    kernel /bzImage-2.6.16.11 root=/dev/sda2 vga=0x0305
```

Then, add the block to the end of the file, and edit the version number to contain the version number of the new kernel you just installed. The title does not matter, so long as it is unique, but it is displayed in the boot menu, so you should make it something meaningful. In our example, we installed the 2.6.17.11 kernel, so the final copy of the file looks like:

```
timeout 300
default 0

splashimage=(hd0,0)/grub/splash.xpm.gz

title 2.6.16.11
    root (hd0,0)
    kernel /bzImage-2.6.16.11 root=/dev/sda2 vga=0x0305

title 2.6.16
    root (hd0,0)
    kernel /bzImage-2.6.16 root=/dev/sda2 vga=0x0305

title 2.6.17.11
    root (hd0,0)
    kernel /bzImage-2.6.17.11 root=/dev/sda2 vga=0x0305
```

After you save the file, reboot the system and ensure that the new kernel image's title comes up in the boot menu. Use the down arrow to highlight the new kernel version, and press Enter to boot the new kernel image.

LILO

To let LILO know that a new kernel is present, you must modify the */etc/lilo.conf* configuration file and then run the *lilo* command to apply the changes made to the configuration file. For full details on the structure of the LILO configuration file, please see the LILO manpage:

```
$ man lilo
```

The easiest way to add a new kernel entry to the */etc/lilo.conf* file is to copy an existing entry. For example, consider the following LILO configuration file from a Gentoo system:

```
boot=/dev/hda
prompt
timeout=50
default=2.6.12

image=/boot/bzImage-2.6.15
    label=2.6.15
    read-only
    root=/dev/hda2

image=/boot/bzImage-2.6.12
    label=2.6.12
    read-only
    root=/dev/hda2
```

The line starting with the word `image=` defines a new kernel entry, so this file contains two entries. Simply copy one block of lines beginning with `image=`, such as:

```
image=/boot/bzImage-2.6.15
    label=2.6.15
    read-only
    root=/dev/hda2
```

Then, add the block to the end of the file, and edit the version number to contain the version number of the new kernel you just installed. The label does not matter, so long as it is unique, but it is displayed in the boot menu, so you should make it something meaningful. In our example, we installed the `2.6.17.11` kernel, so the final copy of the file looks like:

```
boot=/dev/hda
prompt
timeout=50
default=2.6.12

image=/boot/bzImage-2.6.15
    label=2.6.15
    read-only
    root=/dev/hda2

image=/boot/bzImage-2.6.12
    label=2.6.12
    read-only
    root=/dev/hda2

image=/boot/bzImage-2.6.17
    label=2.6.17
    read-only
    root=/dev/hda2
```

After you save the file, run the */sbin/lilo* program to write the configuration changes out to the boot section of the disk:

```
# /sbin/lilo
```

Now the system can be safely rebooted. The new kernel choice can be seen in the list of kernels that are available at boot time. Use the down arrow to highlight the new kernel version, and press Enter to boot the new kernel image.