Upgrading Your Linux Kernel

Matthew J. Fanto
fanto1mj@cmich.edu

December 3, 2002

1 Introduction

The kernel is the very core of an operating system. It controls everything from memory management and process scheduling to hardware and networking (for more information see [1]). The Linux kernel runs on many different platforms, ranging from i386, to ppc, to s390 mainframes. This tutorial will only focus on upgrading the i386 version, but most things found here will be relevant for other platforms.

2 Kernel Versions

Often the first question a person upgrading their kernel for the first time will ask is: what version of the kernel should I install? The kernel version is always of the form V.X.R where V is the kernel version, X denotes stable or development, and R is the release. The kernel version hardly changes (at the time of writing, the current version is 2), If X is an even number (2.2, 2.4, etc.) it denotes a stable kernel. If X is an odd number (2.3, 2.5, etc.) it denotes a development kernel. A development kernel should only be used by developers and those that enjoy data corruption. A development kernel is full of bugs, will take a lot of work to get it to compile properly, and will likely cause data corruption. The kernel release, R, just denotes minor changes from previous versions. It will include bug fixes and small modifications. This tutorial will only focus on installing a stable kernel (at the time of writing: 2.4.19).

3 Patches

Not every feature or driver a user wants can be found in the kernel. Some hardware is just not widely used enough to be included in the main kernel, or some feature is considered too experimental for a stable kernel. Patches are provided by hardware vendors and different kernel hackers. The -ac (Alan Cox patches) patch-set, for example, is usually more experimental in nature. We will not cover patching the kernel here.

4 Getting the Source

Obtain the latest stable source from http://www.kernel.org. You can find mirrors there, please use them and save bandwidth.

5 Unpacking the Source

Move the source tarball to the /usr/src directory and unpack.

```
# tar -xvzf linux-2.4.19.tar.gz
```

This should create a new directory, /usr/src/linux-2.4.19 where the new kernel source is located.

6 Configuring the Kernel

cd into the new directory

```
# cd /usr/src/linux-2.4.19
```

We now need to configure the kernel. This can be done many different ways. make menuconfig will show a text based color menu in which you can select different features/drivers to build. make xconfig will display a GUI configuration tool (requires X11), and make oldconfig will use your existing .config file (and prompt you for answers to new options).

While configuring your kernel, you may have a choice between building as a module and building into the kernel. If you build as a module, you must insert the module when you want to use that device/feature, but if you build into the kernel, it will always be present. If you do not use a piece of hardware often, it is best to build as a module, as the kernel will be smaller. If you are always going to be using something (sound card, ethernet card, etc.) it is probably best to build into the kernel.

For more information on each feature/driver, read the documentation provided (found in Documentation/).

7 Building the Kernel

After you have the kernel configured, you must run

```
# make dep
```

This will set up all the dependencies correctly.

Once make dep finishes, it's time to actually build the kernel

```
# make bzImage
```

Now, go take a nap or write me an email with comments/suggestions on this tutorial as this step will take awhile.

Once make bzImage has finished, you will need to build anything you selected as modules.

```
# make modules
```

If you are building a kernel with the same version as the one currently installed (use uname to check this), backup modules found in /lib/modules/V.X.R/. Now, it's time to install the modules.

```
# make modules_install
```

8 Installing the Kernel

This tutorial assumes you have the LILO boot loader installed. Copy your kernel, found in /usr/src/linux-V.X.R/arch/i386/boot/bzImage to /boot/bzImage-V.X.R. Now, copy System.map, found in /usr/src/linux-V.X.R/ to /boot/System.map-V.X.R. If you already have a kernel with the name bzImge-V.X.R in /boot, it's best to back it up, in case something goes wrong.

```
# cp /usr/src/linux-2.4.19/arch/i386/boot/bzImage /boot/bzImage-2.4.19
```

```
# cp /usr/src/linux-2.4.19/System.map /boot/System.map-2.4.19
```

Now, create a symbolic link from System.map-V.X.R to System.map.

```
# ln -fs /boot/System.map-2.4.19 /boot/System.map
```

Edit your lilo configuration file, found at /etc/lilo.conf. You will need to add the lines

```
image=/boot/bzImage-2.4.19
label=Linux-19
read-only
```

You may make label anything you wish. If you would like the new kernel to be the default kernel, change the default= line. For more lilo options, see the lilo documentation.

YOU MUST RUN THE LILO COMMAND OR THE KERNEL WILL NOT BE INSTALLED. This is a step many people forget.

```
# lilo
```

It should show you that it added the new kernel.

Reboot your computer and enjoy.