

Download the New Source

The Linux kernel developers realize that users do not wish to download the entire source code to the kernel for every update. That would be a waste of bandwidth and time. Because of this, they offer a patch that can upgrade an older kernel release to a newer one.*

On the main *kernel.org* web site, you will remember that it contained a list of the current kernel versions that are available for download, as shown in Figure 6-1.

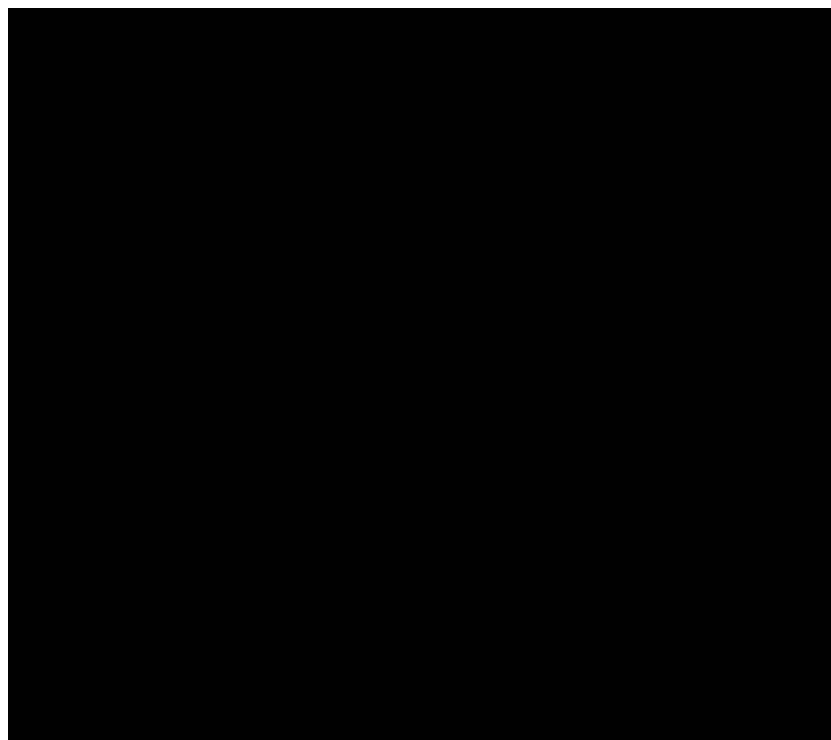


Figure 6-1. The main *kernel.org* web site

Previously, you used the link pointed to you by the F to download the entire source code for the kernel. However, if you click on the name of the kernel release, it will download a patch file instead, as shown in Figure 6-2.

* It is called *patch* because the program *patch* takes the file and applies it to the original tree, creating the new tree. The patch file contains a representation of the changes that are necessary to reconstruct the new files, based on the old ones. Patch files are readable, and contain a list of the lines that are to be removed and the lines that are to be added, with some context within the file showing where the changes should be made.

Finding the Patch

As we want to go from the 2.6.17.9 kernel release, to the 2.6.17.11 release, we will need to download two different patches. We will need a patch from the 2.6.17.9 release to the 2.6.17.10 release, and then from the 2.6.17.10 release to the 2.6.17.11 release.*

The stable and base kernel patches are located in the same directory structure as the main source trees. All incremental patches can be found one level lower, in the *incr* subdirectory. So, to find the patch that goes from 2.6.17.9 to 2.6.17.10, we look in the */pub/linux/kernel/v2.6/incr* directory to find the files we need:†

```
$ cd ~/linux
$ lftp ftp.kernel.org/pub/linux/kernel/v2.6/incr
cd ok, cwd=/pub/linux/kernel/v2.6/incr
lftp ftp.kernel.org:/pub/linux/kernel/v2.6/incr> ls *2.6.17.9*.bz2
-rw-rw-r-- 1 536      536      2872 Aug 22 19:23 patch-2.6.17.9-10.
bz2
lftp ftp.kernel.org:/pub/linux/kernel/v2.6/incr> get patch-2.6.17.9-10.bz2
2872 bytes transferred
lftp ftp.kernel.org:/pub/linux/kernel/v2.6/incr> get patch-2.6.17.10-11.bz2
7901 bytes transferred
lftp ftp.kernel.org:/pub/linux/kernel/v2.6/incr> exit
$ ls -F
good_config linux-2.6.17.9/ patch-2.6.17.10-11.bz2 patch-2.6.17.9-10.bz2
```

Applying the Patch

As the patches we have downloaded are compressed, the first thing to do is uncompress them with the *bzip2* command:

```
$ bzip2 -dv patch-2.6.17.9-10.bz2
patch-2.6.17.9-10.bz2: done
$ bzip2 -dv patch-2.6.17.10-11.bz2
patch-2.6.17.10-11.bz2: done
$ ls -F
good_config linux-2.6.17.9/ patch-2.6.17.10-11 patch-2.6.17.9-10
```

Now we need to apply the patch files to the kernel directory. Go into the directory:

```
$ cd linux-2.6.17.9
```

Now run the *patch* program to apply the first patch moving the source tree from the 2.6.17.9 to the 2.6.17.10 release:

```
$ patch -p1 < ../patch-2.6.17.9-10
```

* If you need to upgrade more than two versions, it is recommended as a way to save steps, to go backward and then upgrade forward. In this case, we could go backward from 2.6.17.9 to 2.6.17 and then forward from 2.6.17 to 2.6.17.11.

† In this example, we use the very good *lftp* FTP program to download the patch files. Any FTP program or a web browser can be used to download the same files. The important thing here is to show where the files are located.

Again verify that the output of the patch program did not show any errors and look at the *Makefile*:

```
$ head -n 5 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 17
EXTRAVERSION = .11
NAME=Crazed Snow-Weasel
```

Now that the source code is successfully updated to the version you wish to use, it is a good idea to go back and change the directory name to refer to the kernel version number to avoid confusion at a later time:

```
$ cd ..
$ mv linux-2.6.17.9 linux-2.6.17.11
$ ls -F
good_config linux-2.6.17.11/ patch-2.6.17.10-11 patch-2.6.17.9-10
```

Reconfigure the Kernel

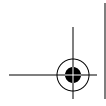
Previously, we used the *make menuconfig* or *gconfig* or *xconfig* method to change different configuration options. But once you have a working configuration, the only thing that is necessary is to update it with any new options that have been added to the kernel since the last release. To do this, the *make oldconfig* and *make silentoldconfig* options should be used.

make oldconfig takes the current kernel configuration in the *.config* file, and updates it based on the new kernel release. To do this, it prints out all configuration questions, and provides an answer for them if the option is already handled in the configuration file. If there is a new option, the program stops and asks the user what the new configuration value should be set to. After answering the prompt, the program continues on until the whole kernel configuration is finished.

make silentoldconfig works exactly the same way as *oldconfig*, but it does not print anything to the screen, unless it needs to ask a question about a new configuration option.

Usually, when upgrading between different versions of the stable releases, no new configuration options are added, as this is supposed to be a stable kernel series. If this happens, there are no new questions that need to be answered for the kernel configuration, so the program continues successfully without any need for user intervention. An example of this is moving from the 2.6.17.9 to 2.6.17.11 release:

```
$ cd linux-2.6.17.11
$ make silentoldconfig
scripts/kconfig/conf -s arch/i386/Kconfig
#
# using defaults found in .config
#
```

Can This Be Automated?

The whole process of downloading the proper patch file, uncompressing it, and then applying it seems to be ripe for automating. Kernel developers being the type that like to automate repetitive tasks, the program *ketchup* has been created to handle all of this automatically. See Appendix A for more details on how this program works and how to use it.

