


```
$ tar -zxf linux-2.6.19.tar.gz
$ ls
linux-2.6.19/
linux-2.6.19-dirty/
```

Now make all of the different changes you wish to do in the *-dirty* directory and leave the clean, original kernel directory alone. After finishing making changes, you should create a patch to send it to other people:

```
$ diff -Naur -X linux-2.6.19/Documentation/dontdiff linux-2.6.19/ \
linux-2.6.19-dirty/ > my_patch
```

This will create a file called *my_patch* that contains the difference between your work and a clean 2.6.19 kernel tree. This patch then can be sent to other people via email.

New Kernel Versions

If a new kernel version is released, and you wish to port your changes to the new version, you need to try to apply your generated patch onto a clean kernel version. This can be done in the following steps:

1. Generate your original patch, as in the previous example.
2. Using the official patch from *kernel.org*, move the old kernel version forward one release:

```
$ cd linux-2.6.19
$ patch -p1 < ../patch-2.6.20
$ cd ..
$ mv linux-2.6.19 linux-2.6.20
```

3. Move your working directory forward one release by removing your patch, then apply the new update:

```
$ cd linux-2.6.19-dirty
$ patch -p1 -R < ../my_patch
$ patch -p1 < ../patch-2.6.20
$ cd ..
$ mv linux-2.4.19-dirty linux-2.6.20-dirty
```

4. Try to apply your patch on top of the new update:

```
$ cd linux-2.6.20-dirty
$ patch -p1 < ../my_patch
```

If your patch does not apply cleanly, resolve all of the conflicts that are created (the *patch* command will tell you about these conflicts, leaving behind *.rej* and *.orig* files for you to compare and fix up manually using your favorite editor). This merge process can be the most difficult part if you have made changes to portions of the source tree that have been changed by other people.

If you use this development process, I highly recommend getting the excellent *patchutils* set of programs (found at <http://cyberelk.net/tim/patchutils>). These programs enable you to manipulate text patches easily in all sorts of useful ways, and have saved kernel developers many hours of tedious work.


```
PATCHLEVEL = 6
SUBLEVEL = 19
-EXTRAVERSION =
+EXTRAVERSION = -dirty
NAME=Crazed Snow-Weasel

# *DOCUMENTATION*
```

You can continue on, working with this single patch, or create a new one to go on top of this patch. As an example, if three different patches had been created, patch1, patch2, and patch3, they will be applied one on top of one another.

To see the list of patches that are currently applied:

```
$ quilt series -v
+ patches/patch1
+ patches/patch2
= patches/patch3
```

This output shows that all three patches are applied, and that the current one is patch3.

If a new kernel version is released, and you wish to port your changes to the new version, quilt can handle this easily with the following steps:

1. Pop off all of the patches that are currently on the tree:

```
$ quilt pop -a
Removing patch patches/patch3
Restoring drivers/usb/Makefile
Removing patch patches/patch2
Restoring drivers/Makefile
Removing patch patches/patch1
Restoring Makefile
No patches applied
```

2. Using the official patch from *kernel.org*, move the old kernel version forward one release:

```
$ patch -p1 < ../patch-2.6.20
$ cd ..
$ mv linux-2.6.19 linux-2.6.20
```

3. Now have quilt push all of the patches back on top of the new tree:

```
$ quilt push
Applying patch patches/patch1
patching file Makefile
Hunk #1 FAILED at 1.
1 out of 1 hunk FAILED -- rejects in file Makefile
Patch patches/patch1 does not apply (enforce with -f)
```

4. As the first patch doesn't apply cleanly, force the patch to be applied and then fix it up:

```
$ quilt push -f
Applying patch patches/patch1
patching file Makefile
Hunk #1 FAILED at 1.
1 out of 1 hunk FAILED -- saving rejects to file Makefile.rej
```


ketchup

ketchup is a very handy tool used to update or switch between different versions of the Linux kernel source tree. It has the ability to:

- Find the latest version of the kernel, download it, and uncompress it.
- Update a currently installed version of the kernel source tree to any other version, by patching the tree to the proper version.
- Handle the different development and stable branches of the kernel tree, including the *-mm* and *-stable* trees.
- Download any patches or tarballs needed to do the update, if they are not present on the machine already.
- Check the GPG signatures of the tarball and patches to verify that it has downloaded a correct file.

ketchup can be found at <http://www.selenic.com/ketchup/> and has lots of additional documentation in the wiki at <http://www.selenic.com/ketchup/wiki/>.

Here is a set of steps that show how simple it is to use *ketchup* to download a specific kernel version, and then have it switch the directory to another kernel version with only a minimal number of commands.

To have *ketchup* download the 2.6.16.24 version of the kernel source tree into a directory, and rename the directory to be the same as the kernel version, enter:

```
$ mkdir foo
$ cd foo
$ ketchup -r 2.6.16.24
None -> 2.6.16.24
Unpacking linux-2.6.17.tar.bz2
Applying patch-2.6.17.bz2 -R
Applying patch-2.6.16.24.bz2
Current directory renamed to /home/gregkh/linux/linux-2.6.16.24
```

Now, to upgrade this kernel to contain the latest stable kernel version, just enter:

```
$ ketchup -r 2.6
2.6.16.24 -> 2.6.17.11
Applying patch-2.6.16.24.bz2 -R
Applying patch-2.6.17.bz2
Downloading patch-2.6.17.11.bz2
--22:21:14-- http://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.17.11.
bz2
=> `/home/greg/.ketchup/patch-2.6.17.11.bz2.partial'
Resolving www.kernel.org... 204.152.191.37, 204.152.191.5
Connecting to www.kernel.org[204.152.191.37]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 36,809 (36K) [application/x-bzip2]
100%[=====] 36,809 93.32K/s
22:21:14 (92.87 KB/s) - `/home/greg/.ketchup/patch-2.6.17.11.bz2.partial'
saved [36809/36809]
Downloading patch-2.6.17.11.bz2.sign
```

