CHAPTER 7

Building the Darwin Kernel

The Darwin kernel, on which Mac OS X is based, is available in a publicly accessible CVS archive. This is not a watered-down version: you can rebuild a kernel that matches your current Mac OS X kernel in every respect. The only noticeable difference will be when you type *uname -v*:

```
Darwin Kernel Version 6.0: Sat Jul 27 13:18:52 PDT 2002;
root:xnu/xnu-344.obj~1/RELEASE_PPC
```

Just because you can build your kernel, does that mean you should? For most users, the answer is *no*, for the following reasons:

- For many users, configuring a Unix kernel involves little more than choosing and configuring device drivers. On Darwin, most devices are not in the kernel; they have their own top-level directory in the CVS archive. So, you do not need to configure Darwin to set up additional hardware support.
- Apple hardware is predictable. Most of you will be building Darwin for a G3 or G4 machine, and the range of possible chipsets is limited.

However, if you want to try installing an unofficial kernel patch, or if you want to try your hand at optimizing the kernel, then this chapter's for you.

Darwin Development Tools

The Darwin kernel requires a collection of development tools that are not part of the Mac OS X Developer Tools package. To get these tools, visit the Darwin project at *http://developer.apple.com/darwin/* and follow the links for the Darwin Development Environment for Mac OS X. Those links lead to a package called *darwintools.pkg*, which you should install. This package installs a number of header files, libraries, and tools into */usr/local*. The tools

A Safety Net

If you have enough disk space to install two copies of Mac OS X, please do so before you start playing around with your working kernel. That way, you will have an operating system you can boot into if things go bad. (On most G3 and G4 Macintoshes, you can hold down the Option key when booting to select a boot disk.) Most importantly, your spare install of Mac OS X will contain backups of important files, such as the kernel and critical frameworks. If you're low on disk space, why not treat yourself to a FireWire drive? If you have a newer Macintosh with a built-in FireWire port, you can boot from a Mac OS X-compatible FireWire drive.

are described in Table 7-1. The source code for these utilities and libraries can be found in the *cctools*, *mkisofs*, *Libstreams*, and *bootstrap_cmds* CVS modules. If you are working with an interim or seed release of Darwin or Mac OS X that is out of sync with the current Darwin Development Environment, you may need to check these utilities out and install them yourself.

Table 7-1. Darwin development tools

Tool	Description	CVS module
check_dylib	Checks the integrity of a dynamic library.	*cctools*
checksyms	Checks a binary to ensure that it adheres to Mac OS X conventions.	*cctools*
cmpshlib	Compares two versions of a shared library for compatibility.	*cctools*
decomment	Strips C and C++ comments from an input file.	*bootstrap_cmds*
devdump	Interactively reads the contents of a device or filesystem image.	*mkisofs* (as *dump.c*)
hack_libgcc	Hacks a framework to export backward-compatible symbols.	*cctools*
indr	Prepends an underscore to selected symbol names in an object file.	*cctools*
isodump	Interactively reads the contents of an ISO 9660 image.	*mkisofs*
isoinfo	Reads information from an ISO 9660 image. Use *isoinfo -h* for a usage summary.	*mkisofs*
isovfy	Verifies an ISO image.	*mkisofs*
kern_tool	Supports cross-compilation of the kernel; a hacked version of the *nm* utility.	*cctools*
mkhybrid	Creates a hybrid ISO 9660/Joliet/HFS filesystem.	*mkisofs*
mkisofs	Creates a hard link to *mkhybrid*.	*mkisofs*
mkshlib	Creates a host and target shared library. The host shared library looks like a static library to the linker, but at runtime, the target shared library is loaded.	*cctools*
relpath	Finds and prints a relative pathname, given a starting directory and an ending directory.	*bootstrap_cmds*

Table 7-1. Darwin development tools (continued)

Tool	Description	CVS module
seg_addr_table	Works with segment address tables.	*cctools*
seg_hack	Changes segment names.	*cctools*
setdbg	Operates as an interactive kernel debugger.	*at_cmds*

Getting the Source Code

To get the Darwin source code, you'll need to register with the Apple Open Source web site and check the source code out of the CVS archive. (The kernel source code weighs in at about 35 MB; after you compile the kernel, it will occupy about 150 MB.) To register for CVS access, visit *http://developer.apple.com/darwin/tools/cvs/*. That page should lead to a getting-started page, where you can register as a user.

The first step in registering is to agree to the Apple Public Source License (*http://www.opensource.apple.com/apsl/*). When you agree to that license, you can create a username and password that lets you check files out of CVS and view the web-based CVS archive.

Using CVS

When you register with Apple, you choose a username and password. You'll need to use that username and password when you log into CVS. The first step is setting your `CVSROOT` environment variable. Under *tcsh*, issue this command:

```
setenv CVSROOT :pserver:username@anoncvs.opensource.apple.com:/cvs/Darwin
```

Under *bash* or *zsh*, use this command:

```
export CVSROOT=:pserver:username@anoncvs.opensource.apple.com:/cvs/Darwin
```

Replace *username* with your username. After you set this environment variable, you can log into CVS with *cvs login*:

```
% cvs login
(Logging in to username@anoncvs.opensource.apple.com)
CVS password: ********
```

Checking out sources

To check out the source code for a module, use the *checkout* command:

```
cvs -z3 checkout [-r VERSION] modulename
```

The *-z3* option tells CVS to use compression when transferring files.

Updating sources

To bring a module into sync with the latest changes to the repository, use the *update* command:

```
cvs -z3 update -P -d modulename
```

The *-d* option tells CVS to pick up any directories that were recently added, and *-P* tells CVS to prune any directories that were recently removed.

If you use *modulename* with the *update* command, you need to be in the same directory where you originally issued the *checkout* command. This will be the parent directory of the module's top-level source directory. If you don't specify a *modulename*, CVS will update only the files in and below your current working directory.

Here is an example session in which a module is checked out, its contents perused, and its source updated to the latest version:

```
% cvs checkout testmodule
cvs checkout: Updating testmodule
U testmodule/Makefile
U testmodule/bar.c
U testmodule/foo.c
% cd testmodule/
% ls -l
total 24
drwxr-xr-x  5 bjepson  staff  126 Apr 10 13:23 CVS
-rw-r--r--  1 bjepson  staff    3 Apr 10 13:22 Makefile
-rw-r--r--  1 bjepson  staff    2 Apr 10 13:22 bar.c
-rw-r--r--  1 bjepson  staff    2 Apr 10 13:22 foo.c

*** time passes ***

% cvs update -P -d
cvs update: Updating .
U bar.c
% ls -l bar.c
-rw-r--r--  1 bjepson  staff  2 Apr 10 13:23 bar.c
```

Getting the Right Version

The only version of Darwin that should work with your copy of Mac OS X is the same one that Apple used. Your mileage may vary if you try to use an older or newer version. So, before you try anything like that, get the correct version and use that as a dry run to verify that you can build and install a working kernel.

First, find your Darwin version with the *uname -v* command. The output you're looking for is the *xnu* (Darwin kernel) version, shown in *italic* type:

```
% uname -v
Darwin Kernel Version 6.0: Sat Jul 27 13:18:52 PDT 2002;
root:xnu/xnu-344.obj~1/RELEASE_PPC
```

You need to translate that number into an Apple CVS tag, by replacing the period (.) with a dash (-) and prefixing the version with `Apple-`. So, the Apple CVS tag for the *xnu* version previously shown would be `Apple-344`. This is the version you must supply with the *-r* flag. Now that you know the CVS tag, you can check it out:

```
cvs -z3 checkout -r APPLE_CVS_TAG modulename
```

Where *APPLE_CVS_TAG* is the CVS tag you computed, and *modulename* is *xnu*. For example:

```
% cvs -z3 checkout -r Apple-344
cvs server: Updating xnu
U xnu/APPLE_LICENSE
U xnu/Makefile
U xnu/PB.project
U xnu/README
.
.
.
```

The CVS tags are symbolic names associated with a snapshot of the source code in time. An easy way to browse the available tags is through the Darwin CVSWeb archive, available at *http://www.opensource.apple.com/tools/cvs/*. You will need to provide your registered username and password to access the archive. You can also use CVSWeb to peruse the archive and view the source code.

Building and Installing the Kernel

Now that you have downloaded the source from CVS, you can change to the *xnu* directory and load some environment variables. If you're using *tcsh*, you can use the following commands:

```
% cd xnu
% source SETUP/setup.csh
```

If you're using *bash* or *zsh*, you can use these commands:

```
$ cd xnu
$ . SETUP/setup.sh
```

To build the kernel, use this command (the output is not shown):

```
% make
```

When *make* is finished, you should see *mach_kernel* in the *xnu/BUILD/obj/RELEASE_PPC* directory. Before you install the new kernel, back up your old kernel as follows:

```
% sudo cp /mach_kernel /mach_kernel.backup
```

Next, copy the new kernel over the older version:

```
% sudo cp BUILD/obj/RELEASE_PPC/mach_kernel /
```

Cross your fingers, knock on wood, and reboot. If all goes well, you should see the build time, hostname, and your username (since you're the person who compiled the kernel) when you run *uname -v*:

```
Darwin Kernel Version 6.0: Thu Aug 22 15:52:19 EDT 2002;
bjepson:BUILD/obj/RELEASE_PPC
```

Once you've made it that far, you can start modifying the code or experimenting with unofficial patches!

Kernel Configuration

Darwin includes configuration directories for each major component of the operating system, as listed here:

xnu/bsd/conf
: Contains configuration files for the BSD portions of Darwin.

xnu/iokit/conf
: Contains configuration files for IOKit, Darwin's subsystem for device drivers.

xnu/libkern/conf
: Contains configuration files for *libkern*, a set of base classes for kernel C++ code.

xnu/libsa/conf
: Contains configuration files for implementations of standard C library functions that are used by the kernel.

xnu/osfmk/conf
: Contains configuration files for the Mach portions of Darwin.

xnu/pexpert/conf
: Contains configuration files for pexpert (platform expert). This is for low-level hardware support during the boot sequence.

libsa and pexpert are private to the *xnu* kernel and should not be used by kernel extensions.

To tweak machine-independent aspects of a Darwin kernel component, you can edit the *MASTER* file in each configuration directory. You can find machine-dependent configuration options in *MASTER.i386* (for *x*86 systems) and *MASTER.ppc* (for PowerPC systems).