18

Browsing the Code

IN THIS CHAPTER

- Mozilla Firefox Trunk
- Mozilla Code Development Cycle
- Coding Practices
- Setting Up and Building Firefox
- Check-in Requirements
- Working with the Mozilla Source Code

For those of us who are programmers, there is nothing like having the source code at hand. Having buildable source lets us fix a nagging bug, add that feature we really want, and generally modify the product.

In this chapter we take the source code and go through the problems you might have building it. When we're done, you should be able to create a version of Firefox (or Thunderbird) that works.

Mozilla Firefox Trunk

You can access several versions of Firefox at any time. One source set is the downloadable source files, which is not the most current source—it might be as much as a month out of date). However, you can get the most current source from Concurrent Versions System (CVS).

Tip

This chapter talks about building Firefox. To build Thunderbird, the same process is used but you simply rename as necessary so that CVS retrieves the Thunderbird source instead of Firefox. Plus, you must modify your `.mozconfig` file.

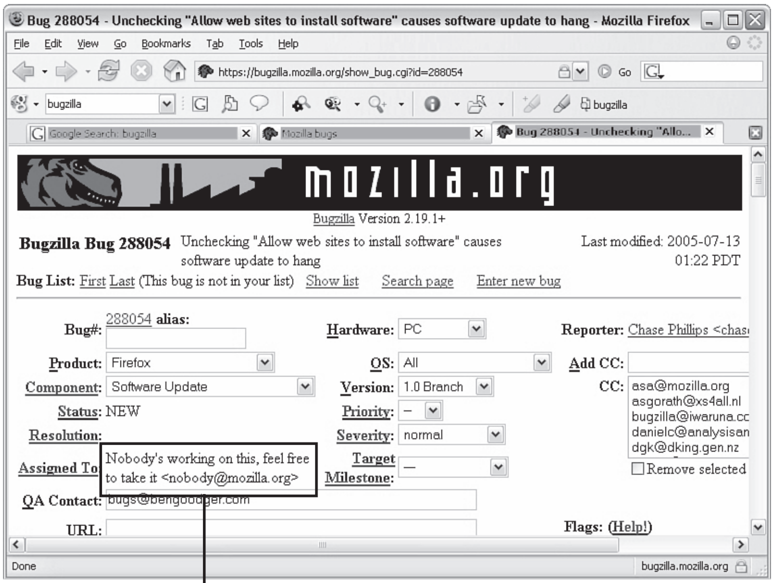
Mozilla Code Development Cycle

To be a successful Mozilla programmer, you have to be a team player. That includes following the established rules, even if you don't agree with them.

The Mozilla code development cycle is, generally, rather simple. Overall, there are only a few steps:

1. Find a bug you want to fix and that you think you can fix. As a suggestion, fix the bug unofficially before asking for the formal approval to work on the bug.

2. Make your fix to the relevant module. Test your fixes—do not just assume that they will work. If at all possible, test it under all platforms, including Windows, Mac, and Linux.
3. Create a patch file using the DIFF command (part of Cygwin).
4. Find the bug and ensure that it is currently unassigned (see Figure 18.1).


Bug 288054 - Unchecking "Allow web sites to install software" causes software update to hang - Mozilla Firefox

https://bugzilla.mozilla.org/show_bug.cgi?id=288054

Bugzilla Version 2.19.1+

Bugzilla Bug 288054 Unchecking "Allow web sites to install software" causes software update to hang

Last modified: 2005-07-13 01:22 PDT

Bug List: First Last (This bug is not in your list) Show list Search page Enter new bug

Bug#:	288054 **alias:**	**Hardware:**	PC	**Reporter:**	Chase Phillips <chas
Product:	Firefox	**OS:**	All	**Add CC:**	
Component:	Software Update	**Version:**	1.0 Branch	**CC:**	asa@mozilla.org asgorath@xs4all.nl bugzilla@iwaruna.cc danielc@analysisan dgk@dking.gen.nz
Status:	NEW	**Priority:**	--		
Resolution:		**Severity:**	normal		
Assigned To:	Nobody's working on this, feel free to take it <nobody@mozilla.org>	**Target Milestone:**	---		☐ Remove selected
QA Contact:	bugs@bengoodger.com				
URL:				**Flags:**	(Help!)

Done bugzilla.mozilla.org

Bugs with this do not have anyone working on them.

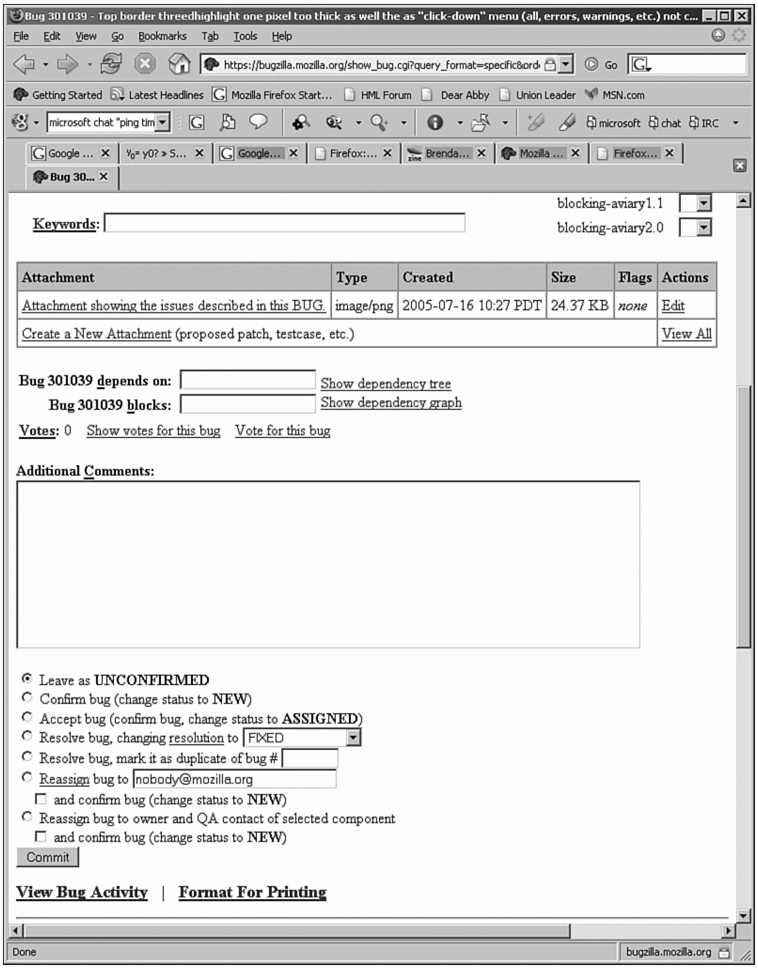
Figure 18.1

This bug is new and is currently unassigned.

5. Request the bug be assigned to you. This is done at the bottom of the Bugzilla bug form, as shown in Figure 18.2.
6. Fix the bug. Generate a patch file, and attach the patch file to the bug in Bugzilla. Request a review for the attachment's patch.

The review process is necessary before your bug will be added to the product. The reviewer (usually the module owner or his peer) checks to ensure that the fix is acceptable and tests the fix. Do not take this as saying that Mozilla doesn't trust you—this benefits you in that there is an extra set of eyes looking for unintended side effects.

Without careful review, there is a tendency for a bug fix to simply cause another bug in the program. Always be careful not to break something else when fixing your bug.


Bug 301039 - Top border threedhighlight one pixel too thick as well the as "click-down" menu (all, errors, warnings, etc.) not c...
File Edit View Go Bookmarks Tab Tools Help
https://bugzilla.mozilla.org/show_bug.cgi?query_format=specific&ord
Go
Getting Started
Latest Headlines
Mozilla Firefox Start...
HML Forum
Dear Abby
Union Leader
MSN.com
microsoft chat "ping tim
microsoft
chat
IRC
Google ...
Google...
Firefox:...
Brenda...
Mozilla ...
Firefox...
Bug 30...
Done
bugzilla.mozilla.org

blocking-aviary1.1
blocking-aviary2.0

Keywords:

Attachment	Type	Created	Size	Flags	Actions
Attachment showing the issues described in this BUG.	image/png	2005-07-16 10:27 PDT	24.37 KB	*none*	Edit
Create a New Attachment (proposed patch, testcase, etc.)					View All

Bug 301039 depends on: Show dependency tree
Bug 301039 blocks: Show dependency graph

Votes: 0 Show votes for this bug Vote for this bug

Additional Comments:

[✓ Leave as UNCONFIRMED]
Confirm bug (change status to NEW)
Accept bug (confirm bug, change status to ASSIGNED)
Resolve bug, changing resolution to FIXED
Resolve bug, mark it as duplicate of bug #
Reassign bug to nobody@mozilla.org
and confirm bug (change status to NEW)
Reassign bug to owner and QA contact of selected component
and confirm bug (change status to NEW)
Commit

View Bug Activity | **Format For Printing**

FIGURE 18.2

A bug's status and resolution can be altered if you have the authority to do so.

Coding Practices

There are standards for programming in C/C++ that you must follow when fixing or enhancing the Mozilla products. A number of things that are important include

- **Use the correct coding style**—There is a code style book for programmers who are working on the Mozilla products. Many of these styles are based on the established C (and C++) styles. Go to http://www.mozilla.org/hacking/mozilla-style-guide.html and read this document about style. Better yet, print it for handy reference.

- **Use the reviewer's guide**—This guide sets the groundwork for code reviewing. Visit http://www.mozilla.org/projects/seamonkey/rules/code_review.html and read this document, even if you are not planning to be a code reviewer.
- **Follow the rules and tips for coding**—Again, visit the web page at http://www.mozilla.org/hacking/reviewers.html#rules-and-tips, which spells out exactly what is good practice and what is bad practice.
- **Make your coding portable**—This means you should not include platform-specific code unless it is absolutely unavoidable. (And it is very rare that it is unavoidable.) Visit http://www.mozilla.org/hacking/portable-cpp.html to read more about making your code portable.

A good starting point for coding is http://www.mozilla.org/hacking/coding-introduction.html. This page is a FAQ containing a lot of useful information. Read this document.

Setting Up and Building Firefox

Building Firefox (or Thunderbird) takes preparation. There are a number of pieces of the build environment that you must have. Everything necessary to build Firefox is available free; however, if you have the full Visual C++ (Visual Studio), you have some additional tools, such as the IDE/editor environment.

Several sites on the Internet deal with setting up the environment and building Mozilla products. One excellent site is http://gemal.dk/mozilla/build.html. A recommendation is to print these pages and use them as a checklist in your quest to build Firefox or Thunderbird.

Necessary Software

The first thing to do is to create a working folder. Eventually, you might need as much as 3.5GB of space, so ensure that the drive you use has the room. Next, in the root of the drive, create a folder named Mozilla. This folder will hold some of your tools and the files used to build Firefox.

To do a successful build, you will need a C/C++ compiler, such as Visual C++ or MinGW. The preferred compiler is Visual C++, and because it can be downloaded free from Microsoft, there is no reason to not use it. If you want to use MinGW, you can download it from the MinGW website at http://www.mingw.org/download/. The information at http://gemal.dk/mozilla/build.html describes which of the MinGW components are necessary and how to install them.

Now, let's get started!

You must first download and install the software for the build environment:

1. Go to http://msdn.microsoft.com/visualc/vctoolkit2003/ and download the Visual C++ 7.1 compiler package.
2. Install the Visual C++ package using the default options. The compiler will be installed in `%programfiles%\microsoft Visual C++ Toolkit 2003`.
3. Download and install the Microsoft Platform SDK (also called the Windows SDK). The SDK will be installed in `%programfiles%\microsoft Platform SDK\`. Again, accept any default option values.
4. Download and install the Microsoft .NET SDK from http://www.microsoft.com/downloads/details.aspx?familyid=262d25e3-f589-4842-8157-034d1e7cf3a3&displaylang=en. This is a large download, however, and it is not clear whether it is actually needed to build Firefox. As with the two previous examples, accept the default installation values.

NOTE You might be able to save some disk space by just installing the needed core, IE, and the Microsoft Data Access Components (MDAC) SDKs.

5. Download compiler-specific versions of glib and libIDL from ftp://ftp.mozilla.org/pub/mozilla.org/mozilla/libraries/win32. If you have installed the visual C++ from step 2, get the VC71 versions of these files (ftp://ftp.mozilla.org/pub/mozilla.org/mozilla/libraries/win32/vc71-glib-1.2.10-bin.zip and ftp://ftp.mozilla.org/pub/mozilla.org/mozilla/libraries/win32/vc71-libIDL-0.6.8-bin.zip). If you have previously installed Visual C++ 7.0, use the VC7 versions. We will install these files next.
6. Download and install the I386 macro assembler MASM from http://www.masm32.com/masmdl.htm. MASM is used to build the crypto components of Firefox. If you install MASM32, you will be installing other utilities, which can cause conflicts between MASM32's linker (LINK.EXE) and lib (LIB.EXE) programs and those supplied with your compiler. Therefore, use the linker and lib programs that come with your compiler; simply rename or delete the MSAM32 versions to resolve this problem.

You now need to install Cygwin from http://www.cygwin.com/setup.exe. Cygwin is a network-based installation, so only those components that you choose to install are downloaded to your computer.

In the Cygwin install, select the following options:

- **All Users**—The installation will be available to all users.
- **UNIX**—Cygwin will use Unix-compatible end-of-line characters.

In the next step of Cygwin's installation, you must select which packages are to be installed:

- **ash***—A Unix-like command-line interpreter shell.
- **bzip2***—An Open Source data compressor.
- **coreutils**—GNU core utilities. These utilities include fileutils, sh-utils, and textutils.
- **cvs**—A client used to download Mozilla source code.
- **cygutils**—A set of general-purpose utilities.
- **diffutils***—A set of file comparison utilities.
- **fileutils***—File and directory management utilities.
- **findutils**—A set of search utilities.
- **gawk***—A pattern matching language.
- **grep***—A text search in files tool.
- **gzip***—A powerful archive builder with a Windows interface.
- **make**—The system that controls building from a make file.
- **patch**—A tool used to modify a source file from a patch file.
- **Perl**—Support for the Perl language.
- **sed***—The search and replace program.
- **sh-utils***—Miscellaneous shell utilities.
- **textutils***—Additional text utilities.
- **unzip**—A command-line compressed archive file extractor, similar to PKZIP.
- **zip**—A command-line compressed archive file creator, similar to PKZIP.

Those packages marked with an asterisk (*) should already be selected, but confirm this.

The next part of the build environment is ActivePerl. ActivePerl is a download found at http://activestate.com/Products/Download/Register.plex?id=ActivePerl. Download the Zip file or the MSI file—both achieve the same results and are almost the same size. Unzip it to a temporary folder. After it's unzipped, the folder will contain a subfolder (ActivePerl) and Installer.bat, the ActivePerl installation program. Run Installer.bat, answering the prompts (the default values should be fine).

Finally, as you reach the end of the gathering of software, you have one more package that must be installed. These are the Netscape Wintools. Download them (http://ftp.Mozilla.org/pub/mozilla.org/mozilla/source/wintools.zip) and extract the buildtools folder from `Wintools.zip`. (As always, retain the folder structure.) You can place the buildtools folder in any convenient location, such as your `c:\mozilla` folder. This location will be referenced by your `buildsetup.bat` file, as described in Listing 18.1, later in this chapter.

NOTE The compiler-specific items (such as glib and libIDL) are dependent on the version of C++ being used. If you are using Microsoft .NET 2003/2005 or C++ 6.0, you need to get the proper versions. Visit ftp://ftp.mozilla.org/pub/mozilla.org/mozilla/libraries/win32/ to retrieve the Visual C++ 7.0, Visual C++ 7.1, and other versions of the supporting files. More help can be found on the Net, though, especially from some of the other third-party builders at http://forums.mozillazine.org.

Next in your `c:\mozilla` folder, create a subfolder named `moztools` (`c:\mozilla\moztools`). Then in the `buildtools` folder that you extracted from Netscape Wintools, enter the following commands:

```
set MOZ_TOOLS=c:\mozilla\moztools
cd c:\mozilla\buildtools\windows
install.bat
```

Now your `c:\mozilla\moztools` folder should have three subfolders (`bin`, `include`, and `lib`).

glib was downloaded from ftp://ftp.Mozilla.org/pub/mozilla.org/mozilla/libraries/win32/. Again, you must extract the contents (a folder named VC71) from this Zip file, preserving the folder structure. In `C:\`, create a subfolder named glib, and in that folder copy the VC71 folder you extracted.

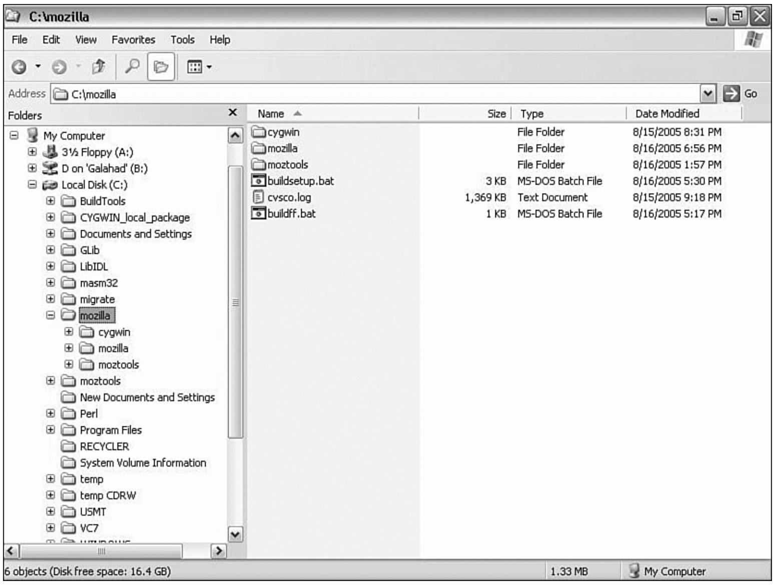
LibIDL was downloaded from ftp://ftp.Mozilla.org/pub/mozilla.org/mozilla/libraries/win32/. Again, you have to extract the contents (a folder named VC71) from this Zip file, preserving the folder structure. In `C:\`, create a subfolder named LibLDL, and in that folder copy the VC71 folder you extracted.

At this point, your Mozilla folder should look like the one shown in Figure 18.3.

In your `%programfiles%\microsoft Visual C++ Toolkit 2003\` folder is a batch file: `vcvars32.bat`. Copy this file to your `C:\mozilla` folder for easier access, if you want.

Next, you must create several new files. The first file is a batch command file you should name `buildsetup.bat`. This file can be created from Listing 18.1, but I strongly recommend you download it from the book's website because it is relatively complex.

If you must type in this file, pay particular attention to the line continuation characters and spaces. Be very careful to not insert spaces where there is a line continuation character; just append the line's text to the previous line's contents with no spaces or other characters. (The best way to get the book's files is to download them from the book's web page at http://www.quepublishing.com/title/0789734583.)



Name	Size	Type	Date Modified
cygwin		File Folder	8/15/2005 8:31 PM
mozilla		File Folder	8/16/2005 6:56 PM
moztools		File Folder	8/16/2005 1:57 PM
buildsetup.bat	3 KB	MS-DOS Batch File	8/16/2005 5:30 PM
cvsco.log	1,369 KB	Text Document	8/15/2005 9:18 PM
buildff.bat	1 KB	MS-DOS Batch File	8/16/2005 5:17 PM

6 objects (Disk free space: 16.4 GB) 1.33 MB My Computer

Figure 18.3

Right now your Mozilla folder has three subfolders.

Listing 18.1 `buildsetup.bat`

```
@ECHO OFF
REM we only want to process these settings one time.
➥Otherwise the path, lib, and include grow out of control

if .%BuildEnvironmentSet% == .YES goto runmake

set BuildEnvironmentSet=YES

REM SET BuildTools_=c:\mozilla\buildtools

REM SET BuildTools=c:\buildtools\buildtools

SET MozTools=c:\mozilla\moztools

REM if you copied this file to your c:\mozilla folder then you need to
REM change the path to c:\mozilla.
call "C:\Program Files\microsoft Visual C++ Toolkit 2003\vcvars32.bat"
```

LISTING 18.1 CONTINUED

```
set path=c:\mozilla\cygwin\bin;%moztools%\bin;
➥%BuildTools%\;c:\glib;c:\libild;c:\masm32\bin;%path%

SET LIB=C:\Program Files\microsoft Platform SDK\Lib;
➥C:\Program Files\microsoft.NET\SDK\v1.1\Lib;
➥C:\Program Files\Microsoft Visual C++ Toolkit 2003\lib;
➥C:\Program Files\microsoft Visual Studio .NET 2003\vc7\lib;
➥C:\masm32\LIB;C:\Program Files\microsoft Platform SDK\Lib\IA64;
➥C:\Program Files\microsoft Platform SDK\Lib\IA64\mfc

SET INCLUDE=C:\Program Files\Microsoft Visual C++ Toolkit 2003\include;
➥C:\Program Files\microsoft platform SDK\include;
➥C:\Program Files\microsoft.NET\SDK\v1.1\include;
➥C:\Program Files\microsoft Visual Studio .NET 2003\Vc7\include;
➥C:\Program Files\microsoft Platform SDK\include\Win64\crt;
➥C:\Program Files\microsoft Platform SDK\include\Win64\mfc;C:\masm32\
➥INCLUDE

REM SET GLIB_PREFIX=c:/mozilla/cygdrive/C/mozilla/glib/vc71
REM SET GLIB_PREFIX=c:/glib/vc71

REM SET LIBIDL_PREFIX=c:/mozilla/cygdrive/c/mozilla/libidl/vc71
REM SET LIBIDL_PREFIX=c:/libidl/vc71

rem SET MOZ_TOOLS=/cygdrive/c/mozilla/buildtools
SET MOZ_TOOLS=c:/mozilla/moztools

SET CVSROOT=:pserver:anonymous@cvs-mirror.Mozilla.org:/cvsroot

SET HOME=c:\mozilla\mozilla

SET CVS_RSH=ssh

SET MOZILLA_OFFICIAL=1

SET BUILD_OFFICIAL=1

ECHO Installation environment variables set!
goto endofbatch

:runmake
ECHO Installation environment variables were already set!
@ECHO ON

:endofbatch
@ECHO ON
```

Several lines in this file need special attention:

```
call "C:\Program Files\microsoft Visual C++ Toolkit 2003\vcvars32.bat"
```

This line calls the batch command file that sets the Microsoft Visual C++ environment variables. Using a call enables you to later add lines in your batch file after this line. Without using a call, `vcvars32.bat` would never return to your main `buildsetup.bat` command file. An explicit path is used to ensure that Windows will find the file.

```
SET GLIB_PREFIX=/cygdrive/C/mozilla/glib/vc71
```

This tells the build environment where the glib files are located. Notice that it uses forward slashes (/) instead of Windows's backslashes (\). Backslashes will cause the build to fail. This line is optional and can be remarked out if desired.

```
SET LIBIDL_PREFIX=/cygdrive/c/mozilla/libidl/vc71
```

This tells the build environment where the libIDL files are located. As noted previously, it uses forward slashes instead of Windows's backslashes. As with glib, this line is optional.

```
SET MOZ_TOOLS=/cygdrive/c/mozilla/buildtools
```

This tells the build environment where the Netscape Wintools are located. As with the two previous examples, it uses forward slashes instead of Windows's backslashes.

```
SET CVSROOT=:pserver:anonymous@cvs-mirror.mozilla.org:/cvsroot
```

This line tells CVS from where to obtain the source. Notice that it does not specify either Firefox or Thunderbird; it only gives the URL and the logon information (anonymous) for CVS.

```
SET HOME=c:\mozilla\mozilla
```

The home variable tells CVS where the downloaded source CVS retrieves will be located.

```
SET CVS_RSH=ssh
```

This specifies which external program will be used to access CVS when using the access method `:ext:`.

```
SET MOZILLA_OFFICIAL=1
```

This environment variable is used to allow you to run two or more builds using the same profile, which requires a build ID. The build ID is created when this variable is set to `1`.

```
SET BUILD_OFFICIAL=1
```

The `BUILD_OFFICIAL` environment variable turns on the build number in the title bar.

The next file you need is your `.mozconfig` file (see Listing 18.2). Notice the leading period—technically, this file has no name, only an extension. If you create this file without the period, it will not work. The `.mozconfig` file must show the build environment which project is being built. For our purposes, we might be building either Firefox or Thunderbird. The `MOZ_CO_PROJECT` will either be set to browser for Firefox or to mail for Thunderbird.

CAUTION Cygwin fixes the path environment variable, ensuring that the file and folder locations are compliant with the environment. However, Cygwin does not update any other environment variables.

LISTING 18.2 `.mozconfig`

```
#########################################
#Firefox is browser, Thunderbird is mail

. $topsrcdir/browser/config/mozconfig

export BUILD_OFFICIAL=1
export MOZILLA_OFFICIAL=1
mk_add_options BUILD_OFFICIAL=1
mk_add_options MOZILLA_OFFICIAL=1
mk_add_options MOZ_CO_PROJECT=browser

# If you want to create a non-static build, comment out the following
#lines
# The installer scripts can only create installers from static builds
ac_add_options —disable-shared
ac_add_options —enable-static

ac_add_options —enable-application=browser
ac_add_options —disable-activex
ac_add_options —disable-activex-scripting
#########################################
```

This file tells the make process to add `BUILD_OFFICIAL` and `MOZILLA_OFFICIAL` to the build. It also tells the generation process to create a static build.

You now have your build environment complete. Next, you need to obtain the Firefox source code using CVS. The process is as follows:

1. Open a command prompt by selecting Start, All Programs, Accessories, Command Prompt.
2. Change into your C:\mozilla folder and type **`buildsetup.bat`** to configure the environment.
3. Enter the command **`cvs login`**. At the password prompt, enter **`anonymous`**. You might receive an error message telling you that the `.cvspass` file does not exist; you can ignore this.

4. Enter the command **`cvs co mozilla/browser/config mozilla/client.mk`** and press Enter. This command retrieves one file—`client.mk`—and places this file in the subfolder mozilla inside your mozilla folder. If this subfolder does not exist, it is created. This new folder will be the home for your Firefox project.
5. Change into the new Mozilla folder created in step 4.
6. Enter the command **`make -f client checkout`**. This command tells make (a system that interprets a command file and calls other programs, something like batch or scripting) and uses CVS to fetch the entire source for Firefox. If you are on a slow connection, this command might take a while to finish; really fast connections usually take only about 5–10 minutes.

At this point, you have now installed the necessary software to build Firefox, installed all the required utilities, and retrieved the very latest (and perhaps, greatest) version of Firefox. The only thing left to do is to build it.

Performing a Build

There are several ways to build the project. One way is to run the makefile program from a command prompt. Type

```
make -f client.mk build
```

Of course, this command could be placed in a batch file along with a test to ensure that the `buildsetup.bat` file has also executed (see Listing 18.3).

Listing 18.3 BuildFireFox.bat

```
if .%BuildEnvironmentSet% == . goto runmake

echo doing build!
make -f client.mk build_all

goto endofall

:runmake
echo No build environment configured.

:endofall
```

This batch file checks to see whether the `buildsetup.bat` file has executed—if the `BuildEnvironmentSet` environment variable is set, `buildsetup.bat` has already executed. When it's sure that the build environment is set up, it calls the make utility to do the actual build of Firefox.

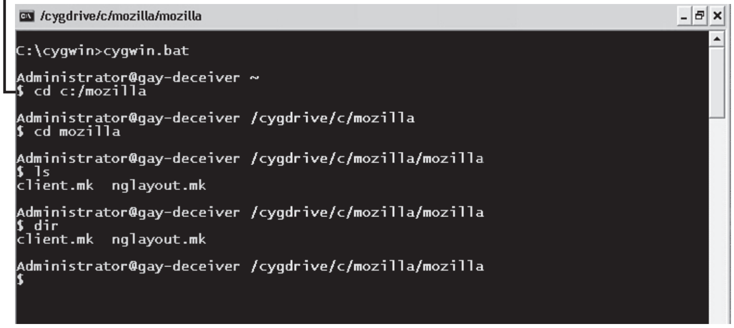
If you are more familiar with a Linux/Unix environment, you can use Cygwin, which you installed earlier in this chapter. That is one program we've not mentioned yet, and for a reason. You can, in a regular Windows command prompt, get the Firefox source. However, to build Firefox, you must use the Cygwin tools. One tool is a Unix shell emulator—just the thing for those of us who work with Windows but wish we were using Unix.

TIP If you want to try Cygwin and are not familiar with Unix (or Linux), you might initially have some problems with the concepts behind how Cygwin works. First, whereas Windows (and MS-DOS) uses a backslash to separate folders and filenames, Unix has always used the forward slash to do folder and file delimiting.

If you have not already done so, you must also configure the build environment using your `buildsetup.bat` file. Simply go to your `c:\mozilla` folder, enter the command **`buildsetup.bat`**, and press Enter.

Now, go to the Cygwin folder. It would be `C:\Cygwin` if you installed Cygwin using the defaults, although I installed Cygwin in a subfolder under `c:\mozilla`. In that folder is a single batch file: `cygwin.bat`. This is the command to start the Cygwin shell. Type **`Cygwin.bat`** and press Enter. You will get a prompt consisting of your *`Windows username@computer name`* and a folder specification (usually either a ~ for the root folder or the current folder). This prompt should look like that shown in Figure 18.4. (For the curious, Gay Deceiver is the name of the computerized car in Robert Heinlein's novel *Number of the Beast* and several other related stories.)

User input follows the $ prompt.



```
C:\cygwin>cygwin.bat

Administrator@gay-deceiver ~
$ cd c:/mozilla

Administrator@gay-deceiver /cygdrive/c/mozilla
$ cd mozilla

Administrator@gay-deceiver /cygdrive/c/mozilla/mozilla
$ ls
client.mk  nglayout.mk

Administrator@gay-deceiver /cygdrive/c/mozilla/mozilla
$ dir
client.mk  nglayout.mk

Administrator@gay-deceiver /cygdrive/c/mozilla/mozilla
$
```

FIGURE 18.4

Everything typed after the `$` *prompt was entered by the user.*

Also notice in Figure 18.4 that I entered the command `ls` and the command `dir`. Both commands do the same thing; `ls` is simply the Unix command for listing a directory's contents. I typed these commands prior to retrieving the Firefox source; otherwise, the results of listing the directory would fill the screen.

NOTE Mozilla recommends a Pentium 500MHz processor and 256MB of RAM to build Firefox. A faster processor and more RAM will significantly improve the build times, though. I personally do not recommend building Firefox on a notebook computer because the build process is very processor intensive and generates substantial heat.

TIP Do not expect your first try at building Firefox to be successful. If it is, that's great, but if not, do not be discouraged. Most of us spend several days getting everything working together before we get a successful build of Firefox. I've seen cases where it is best to simply start from scratch—that is, delete the source folder (the mozilla folder that is inside `c:\mozilla`) and refetch the source files.

We are finally ready to build Firefox. The `client.mk` file is the `make` command script file that both fetches the Firefox source code and builds Firefox. (Somewhat universal, isn't it?) What `client.mk` does depends on the options passed with the `make` command. If the `checkout` option is used, `make` fetches (or checks out) the source code. If the `build` option is used, `make` builds Firefox.

All you have to do now is type the following at Cygwin's `$` prompt:

```
make -f client.mk build_all
```

Whichever way you choose to build—from either a DOS command prompt window or a Cygwin window—your reward, if you've made no mistakes, will be a lengthy build process as the compiler is called to compile each source file, the linker is used to build the libraries and the executable, and other tasks are performed. Keep in mind that Firefox is a large program and this build process can therefore take considerable time on a slower computer. An example of build performance is a 2.2GHz Athlon64 with 2GB of memory, on which it takes approximately 28 minutes to build. A slower notebook with a 1.5GHz processor with 512MB of RAM took several hours to build Firefox. The first build takes the longest because subsequent builds usually have to rebuild only those parts of Firefox that have changed since the previous build.

Packaging Firefox for Distribution

The results of building the product are placed in the `\mozilla\mozilla\dist` folder. A second copy can be located elsewhere—for example, for Firefox a copy will also be in `\mozilla\mozilla\browser\app`.

The final step you must take to enable you to easily distribute the product is to create the installer (the same program you downloaded when you fetched Firefox or Thunderbird from Mozilla). This enables you to have a single file that can be shared among several users.

After your build has completed successfully, you can create the installer. The steps to do this are relatively straightforward:

1. First, if you have not already done so, run your `buildsetup.bat` file to initialize the Mozilla build environment.

2. Move to the folder `c:\mozilla\mozilla\browser\installer`.
3. Enter the following command, which creates the installation package:

   ```
   make installer
   ```
4. Your final installation executable will be located in the folder `c:\mozilla\mozilla\browser\installer`. (The filename can vary depending on which version or product you are creating, but it should be obvious based on the file's timestamp.)

NOTE To use the installer, you need to build a static build of Firefox or Thunderbird. This is controlled in the `.mozconfig` file's options `ac_add_options --disable-shared` and `ac_add_options --enable-static`. Refer to Listing 18.2 for an example of these two options.

You can now distribute your version of Firefox (or Thunderbird) by download or by other means, such as CD-R or network share.

NOTE Mozilla suggests that you uncheck the Quality Feedback Agent check box.

Check-in Requirements

The check-in process is basically a series of established steps that must be followed in order. Don't skip a step for any reason. From the beginning, the check-in process is as follows:

1. Write code, whether it is a patch for a bug or new code for additional (new) functionality. After your code is complete, you need to check out the latest version of Firefox from the CVS system. Without the latest version, the changes might break another patch or new component. If what you are doing is going to take a longer time (say, a week or more), you should check out the latest version from CVS from time to time.
2. Update your local tree (the source). Before checking out the source tree, first make sure that Tinderbox says the code is green—that is, it builds and tests okay. It is worthless to check out code while the tree is not green because your builds will not be reliable. Red means stop and green means go.
3. Have your work reviewed. It must be reviewed before it can be added to the tree. This review serves as a check to ensure that whatever was done is acceptable (for instance, that the coding style and syntax are correct). No code is ever added to the project without review. Reviews are done by the module owner or someone who is designated by the module owner.
4. Run prechecking tests. Established tests are already defined to test your code. Do not assume that your code will not have an adverse effect on other parts of the project. For information on precheck tests, visit http://www.Mozilla.org/quality/precheckin-tests.html.

5. Check in your work. Tinderbox must indicate that the tree is open (allows check-in) and not closed (no check-ins are allowed). No check-in is allowed when the tree is red—that is, when the project will not properly build.
6. Be responsible for your code. After your work has been checked in, you are responsible if it causes a problem! That is why you must do your precheck-in testing, and why the code must be reviewed.

It is important that, until the version that has your changes has properly built (the Tinderbox tree is green), you remain available. If something goes wrong, it is your responsibility to make it right.

Mozilla Source Code Secrets for Power Users

Here are a few ideas from the experts:

- The Mozilla Firefox and Thunderbird source and project can be visualized as a tree. From the trunk (the main project, such as Firefox) different versions of the product branch off. Usually, the tree has at least two branches: the currently released branch and the development branch.
- As time passes, new branches are added to the tree. Old, unnecessary branches are cut off.
- Code development at Mozilla follows a set process. All work is initiated as a bug using Bugzilla. Developers both at Mozilla and externally then undertake to resolve these bugs, either fixing the problems or adding functionality.
- To accommodate a large group of developers—many of whom are not under the direct control of Mozilla—a set of coding practices has been established. These practices must be followed for work to be accepted.
- Setting up a Firefox development environment takes some time and effort. Even experienced programmers take a few days to get to the point where they can build one of the Mozilla products.
- Several necessary products or tools must be used when building Firefox and other Mozilla products. These products or tools are all available freely on the Internet.
- Performing a build involves retrieving the latest source using CVS and then building the product. Changes need to be managed with patch files so they can be applied to new versions of the source.
- The check-in requirements help ensure that work on Firefox and other Mozilla products meets the necessary standards for programming style and syntax and that it does not break the product.