CHAPTER 3

Installing and Configuring Apache

In this second of three installation-related chapters, you will install the Apache web server and familiarize yourself with its main components, including log and configuration files.

In this chapter, you will learn

- How to install the Apache server
- How to make configuration changes to Apache
- Where Apache log and configuration files are stored

Current and Future Versions of Apache

The installation instructions in this chapter refer to Apache HTTPD server version 2.0.58, which is the current production version of the software. The Apache Software Foundation uses minor release numbers for updates containing security enhancements or bug fixes. Minor releases do not follow a set release schedule; when enhancements or fixes are added to the code and thoroughly tested, the Apache Software Foundation releases a new version with a new minor version number.

It is possible that by the time you purchase this book, the minor version number will have changed, to 2.0.59 or beyond. If that is the case, you should read the list of changes, which is linked from the download area at http://httpd.apache.org/download.cgi, for any changes regarding the installation or configuration process, which makes up the bulk of this chapter.

Although it is unlikely that any installation instructions will change between minor version updates, you should get in the habit of always checking the changelog of software that you install and maintain. If a minor version change does occur during the time you are reading this book, but no installation changes are noted in the changelog, simply make a mental note and substitute the new version number wherever it appears in the installation instructions and accompanying figures.

Choosing the Appropriate Installation Method

You have several options when it comes to getting a basic Apache installation in place. Apache is open source, meaning that you can have access to the full source code of the software, which in turn enables you to build your own custom server. Additionally, pre-built Apache binary distributions are available for most modern UNIX platforms. Finally, Apache comes already bundled with a variety of Linux distributions, and you can purchase commercial versions from software vendors such as Covalent Technologies and IBM. The examples in this chapter will teach you how to build Apache from source if you are using Linux/UNIX, and how to use the installer if you plan to run Apache on a Windows system.

Building from Source

Building from source gives you the greatest flexibility because it enables you to build a custom server, remove modules you do not need, and extend the server with third-party modules. Building Apache from source code enables you to easily upgrade to the latest versions and quickly apply security patches, whereas updated versions from vendors can take days or weeks to appear.

The process of building Apache from the source code is not especially difficult for simple installations but can grow in complexity when third-party modules and libraries are involved.

Installing a Binary

Linux/UNIX binary installations are available from vendors and can also be downloaded from the Apache Software Foundation website. They provide a convenient way to install Apache for users with limited system administration knowledge, or with no special configuration needs. Third-party commercial vendors provide prepackaged Apache installations together with an application server, additional modules, support, and so on.

The Apache Software Foundation provides an installer for Windows systems—a platform where a compiler is not as commonly available as in Linux/UNIX systems.

Installing Apache on Linux/UNIX

This section explains how to install a fresh build of Apache 2.0.58 on Linux/UNIX. The general steps necessary to successfully install Apache from source are as follows:

1. Download the software
2. Run the configuration script
3. Compile the code and install it

The following sections describe these steps in detail.

Downloading the Apache Source Code

The official Apache download site is located at http://httpd.apache.org/download.cgi. You can find several versions of the Apache source code, packaged with different compression methods. The distribution files are first packed with the `tar` utility and then compressed either with the `gzip` tool or the `compress` utility. Download the `*.tar.gz` version if you have the `gunzip` utility installed on your system. This utility comes installed by default in open source operating systems such as FreeBSD and Linux. Download the `*.tar.Z` file if `gunzip` is not present in your system. (It isn't included in the default installation of many commercial UNIX operating systems.)

The file you want to download will be named something similar to `httpd-2.0.VERSION.tar.Z` or `httpd-2.0.`*`VERSION`*`.tar.gz`, where *`VERSION`* is the most recent release of Apache. For example, Apache version 2.0.58 is downloaded as a file named `httpd-2.0.58.tar.gz`. Keep the downloaded file in a directory reserved for source files, such as `/usr/src/` or `/usr/local/src/`.

Uncompressing the Source Code

If you downloaded the tarball compressed with `gzip` (it will have a `tar.gz` suffix), you can uncompress it using the `gunzip` utility (part of the `gzip` distribution).

Tarball is a commonly used nickname for software packed using the `tar` utility.

By the Way

You can uncompress and unpack the software by typing the following command:

```
# gunzip < httpd-2.0*.tar.gz ¦ tar xvf -
```

If you downloaded the tarball compressed with `compress` (it will have a `tar.Z` suffix), you can issue the following command:

```
# cat httpd-2.0*.tar.Z ¦ uncompress ¦ tar xvf -
```

Uncompressing the tarball creates a structure of directories, with the top-level directory named `httpd-2.0_VERSION`. Change your current directory to this top-level directory to prepare for configuring the software.

Preparing to Build Apache

You can specify which features the resulting binary will have by using the `configure` script in the top-level distribution directory. By default, Apache will be compiled with a set of standard modules compiled statically and will be installed in the `/usr/local/apache2` directory. If you are happy with these settings, you can issue the following command to configure Apache:

```
# ./configure
```

However, in preparation for the PHP installation in Chapter 4, "Installing and Configuring PHP," you will need to make sure that `mod_so` is compiled into Apache. This module, named for the UNIX shared object (`*.so`) format, enables the use of dynamic modules such as PHP with Apache. To configure Apache to install itself in a specific location (in this case, `/usr/local/apache2/`) and to enable the use of `mod_so`, issue the following command:

```
#./configure --prefix=/usr/local/apache2 --enable-module=so
```

The purpose of the `configure` script is to figure out everything related to finding libraries, compile-time options, platform-specific differences, and so on, and to create a set of special files called *makefiles*. Makefiles contain instructions to perform different tasks, called *targets*, such as building Apache. These files will be read by the `make` utility, which will carry out those tasks. If everything goes well, after executing `configure`, you will see a set of messages related to the different checks just performed and will be returned to the prompt:

```
...
configure ok
creating test/Makefile
config.status: creating docs/conf/httpd.conf
...
config.status: executing default commands
#
```

If the `configure` script fails, warnings will appear, alerting you to track down additional software that must be installed, such as compilers or libraries. After you install any missing software, you can try the `configure` command again, after deleting the `config.log` and `config.status` files from the top-level directory.

Building and Installing Apache

The `make` utility reads the information stored in the makefiles and builds the server and modules. Type `make` at the command line to build Apache. You will see several messages indicating the progress of the compilation, and you will end up back at the prompt. After compilation is finished, you can install Apache by typing `make`

`install` at the prompt. The makefiles will install files and directories and return you to the prompt:

```
...
Installing header files
Installing build system files
Installing man pages and online manual
...
make[1]: Leaving directory `/usr/local/bin/httpd-2.2.0'
#
```

The Apache distribution files should now be in the `/usr/local/apache2` directory, as specified by the `--prefix` switch in the `configure` command. To test that the `httpd` binary has been correctly built, type the following at the prompt:

```
# /usr/local/apache2/bin/httpd -v
```

You should see the following output (your version and build date will be different):

```
Server version: Apache/2.0.58
Server built:   May 12 2006 11:47:22
```

Unless you want to learn how to install Apache on Mac OS X or Windows, skip ahead to the "Apache Configuration File Structure" section to learn about the Apache configuration file.

Installing Apache on Mac OS X

Lucky you, Apache is already installed on Mac OS X! By default, the Apache server binary is located at `/usr/sbin/httpd`. Configuration files such as `httpd.conf`, the master configuration file for Apache, are found in `/etc/httpd`. Because Apache is ready to go (and is fully prepared to use PHP), skip ahead to the "Apache Configuration File Structure" section to learn about the Apache configuration file and how to use it.

Installing Apache on Windows

Apache 2.0 runs on most Windows platforms and offers increased performance and stability over the Apache 1.3 versions for Windows. You can build Apache from source, but because not many Windows users have compilers, this section deals with the MSI installer version.

Before installing Apache, you'll probably want to make sure that you are not currently running a web server (for instance, a previous version of Apache, Microsoft Internet Information Server, or Microsoft Personal Web Server) on your machine. You might want to uninstall or otherwise disable existing servers. You can run several web servers, but they must run in different address and port combinations.

Before downloading the installer, take a moment—a very important moment—and look for a statement on the downloads page (found at http://httpd.apache.org/download.cgi) that says "If you are downloading the Win32 distribution, please read these important notes." The direct URL to these notes is http://www.apache.org/dist/httpd/binaries/win32/README.html.

The Apache Software Foundation maintains this page for the benefit of Windows users who want to run a version of the Apache server. On this page, there are notes for nearly every flavor of Windows still in use, and as such it will be in your best interest to read the information that is presented. I guarantee that if you are running Apache either as a production or development server, you find relevant information on the notes page.

When you're ready to begin the installation, look for the link labeled Win32 Binary (MSI Installer). After you download the installer, double-click the file to start the installation process. You will get a welcome screen, as shown in Figure 3.1.

Click Next to continue the installation process, and you will be prompted to accept the Apache license. Basically the license says that you can do whatever you want with the software—including making proprietary modifications—except claim that you wrote it, but be sure to read the license so that you fully understand the terms.

FIGURE 3.1 The Windows installer welcome screen.

After you accept the license, the installer presents you with a brief introduction to Apache. Following that, it asks you to provide basic information about your computer, as shown in Figure 3.2. This includes the full network address for the server (for instance, `mycomputer.mydomain.com`) and the administrator's email address. The server name is the name your clients will use to access your server, and the

administrator email address will be added to error messages so that visitors know how to contact you when something goes wrong.

Also on this screen, you are prompted to select which installation shortcuts should be installed—those for starting Apache as a service or those for starting Apache manually. Installing Apache as a service causes it to run every time Windows is started, and you can control it through the standard Windows service administration tools. Installing Apache for the current user requires you to start Apache manually and set the default port on which Apache listens to requests to 8080 (instead of 80). Select the appropriate radio button and click Next to continue.

By the Way

If your machine does not have a full network address, use `localhost` or `127.0.0.1` as the server name.

FIGURE 3.2
The basic information screen.

The next screen enables you to choose the type of installation, as shown in Figure 3.3. A Typical installation means that Apache binaries and documentation will be installed, but headers and libraries will not. This is the best option to choose unless you plan to compile your own modules.

A Custom installation enables you to choose whether to install header files or documentation. After selecting the target installation directory, which defaults to `c:\Program Files\Apache Group`, the program will proceed with the installation process. If everything goes well, it will present you with the final screen shown in Figure 3.4.

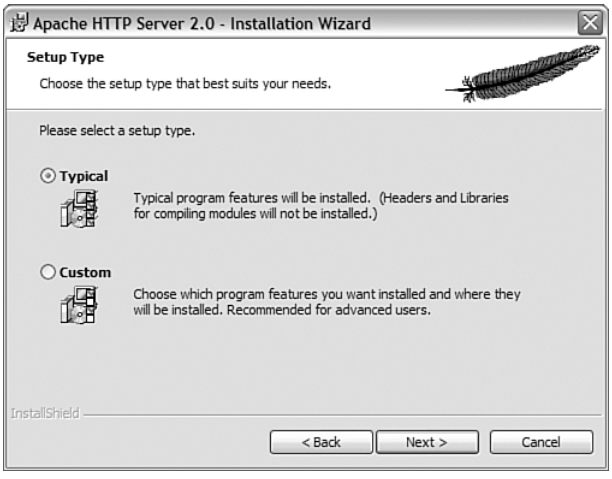

FIGURE 3.3
The installation type selection screen.

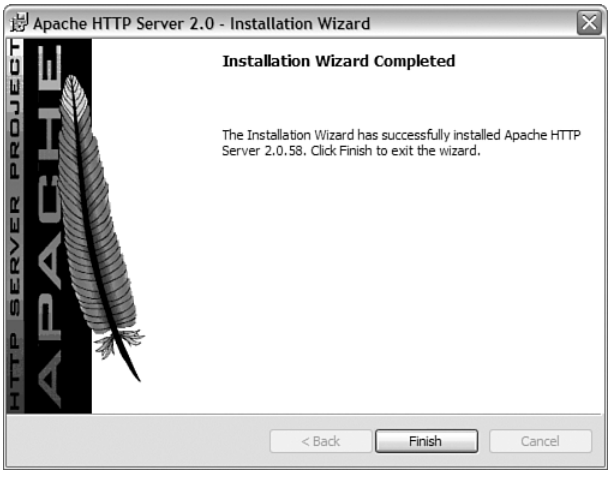

FIGURE 3.4
The successful installation screen.

In the next section, you'll learn about the Apache configuration file, and eventually start up your new server.

Apache Configuration File Structure

Apache keeps all its configuration information in text files. The main file is called `httpd.conf`. This file contains directives and containers, which enable you to customize your Apache installation. *Directives* configure specific settings of Apache, such as authorization, performance, and network parameters. *Containers* specify the context to which those settings refer. For example, authorization configuration can refer to the server as a whole, to a directory, or to a single file.

Directives

The following rules apply for Apache directive syntax:

- The directive arguments follow the directive name.
- Directive arguments are separated by spaces.
- The number and type of arguments vary from directive to directive; some have no arguments.
- A directive occupies a single line, but you can continue it on a different line by ending the previous line with a backslash character (\).
- The pound sign (#) should precede the directive, and must appear on its own line.

The Apache server documentation offers a quick reference for directives at http://httpd.apache.org/docs/2.0/mod/quickreference.html. You'll soon learn about some of the basic directives, but you should supplement your knowledge using the online documentation.

The Apache documentation for directives typically follows this model:

- **Description**—This entry provides a brief description of the directive.
- **Syntax**—This entry explains the format of the directive options. Compulsory parameters appear in italics, optional parameters appear in italics and brackets.
- **Default**—If the directive has a default value, it will appear here.
- **Context**—This entry details the containers or sections in which the directive can appear. Containers are explained in the next section. The possible values are `server config`, `virtual host`, `directory`, and `.htaccess`.
- **Override**—Apache directives belong to different categories. The Override field is used to specify which directive categories can appear in `.htaccess` per-directory configuration files.
- **Status**—This entry indicates whether the directive is built in Apache (`core`), belongs to one of the bundled modules (`base` or `extension`, depending on whether they are compiled by default), is part of a Multi-Processing Module (`MPM`), or is bundled with Apache but not ready for use in a production server (`experimental`).
- **Module**—This entry indicates the module to which the directive belongs.
- **Compatibility**—This entry contains information about which versions of Apache support the directive.

Further explanation of the directive follows these entries in the documentation, and a reference to related directives or documentation might appear at the end.

Containers

Directive containers, also called *sections*, limit the scope for which directives apply. If directives are not inside a container, they belong to the default server scope (`server config`) and apply to the server as a whole.

These are the default Apache directive containers:

- `<VirtualHost>` —A `VirtualHost` directive specifies a virtual server. Apache enables you to host different websites with a single Apache installation. Directives inside this container apply to a particular website. This directive accepts a domain name or IP address and an optional port as arguments. You will learn more about virtual hosts in Chapter 29, "Apache Performance Tuning and Virtual Hosting."

- `<Directory>`, `<DirectoryMatch>` — These containers allow directives to apply to a certain directory or group of directories in the file system. `Directory` containers take a directory or directory pattern argument. Enclosed directives apply to the specified directories and their subdirectories. The `DirectoryMatch` container allows regular expression patterns to be specified as an argument. For example, the following allows a match of all second-level subdirectories of the `www` directory that are made up of four numbers, such as a directory named after a year and month (`0906` for September 2006):

  ```
  <DirectoryMatch "^/www/.*/[0-9]{4}">
  ```

- `<Location>`, `<LocationMatch>` — These containers allow directives to apply to certain requested URLs or URL patterns. They are similar to their `Directory` counterparts. `LocationMatch` takes a regular expression as an argument. For example, the following matches directories containing either `"/my/data"` or `"/your/data"`:

  ```
  <LocationMatch "/(my¦your)/data">
  ```

- `<Files>`, `<FilesMatch>` — Similar to `Directory` and `Location` containers, `Files` sections allow directives to apply to certain files or file patterns.

Containers surround directives, as shown in Listing 3.1.

LISTING 3.1 Sample Container Directives

```
 1: <Directory "/some/directory">
 2: SomeDirective1
 3: SomeDirective2
 4: </Directory>
 5: <Location "/downloads/*.html">
 6: SomeDirective3
 7: </Location>
 8: <Files "\.(gif¦jpg)">
 9: SomeDirective4
10: </Files>
```

Sample directives *SomeDirective1* and *SomeDirective2* will apply to the directory `/some/directory` and its subdirectories. *SomeDirective3* will apply to URLs referring to pages with the `.html` extension under the `/downloads/` URL. *SomeDirective4* will apply to all files with `.gif` or `.jpg` extensions.

Conditional Evaluation

Apache provides support for conditional containers. Directives enclosed in these containers will be processed only if certain conditions are met.

- `<IfDefine>`—Directives in this container will be processed if a specific command-line switch is passed to the Apache executable. The directive in Listing 3.2 will be processed only if the `-D`*`MyModule`* switch is passed to the Apache binary being executed. You can pass this directly or by modifying the `apachectl` script, as described in the "Apache-Related Commands" section later in this chapter.

  `IfDefine` containers also allow the argument to be negated. That is, directives inside a `<IfDefine !`*`MyModule`*`>` section—notice the exclamation point before the *MyModule* name—will be processed only if no `-D`*`MyModule`* parameter is passed as a command-line argument.

- `<IfModule>`—Directives in an `IfModule` section will be processed only if the module passed as an argument is present in the web server. For example, Apache ships with a default `httpd.conf` configuration file that provides support for different MPMs. Only the configuration belonging to the MPM compiled into Apache will be processed, as you can see in Listing 3.3. The purpose of the example is to illustrate that only one of the directive groups will be evaluated.

LISTING 3.2 `IfDefine` Example

```
1: <IfDefine MyModule>
2: LoadModule my_module modules/libmymodule.so
3: </IfDefine>
```

LISTING 3.3 `IfModule` Example

```
 1: <IfModule prefork.c>
 2: StartServers         5
 3: MinSpareServers      5
 4: MaxSpareServers     10
 5: MaxClients          20
 6: MaxRequestsPerChild  0
 7: </IfModule>
 8:
 9: <IfModule worker.c>
10: StartServers         3
11: MaxClients           8
12: MinSpareThreads      5
13: MaxSpareThreads     10
14: ThreadsPerChild     25
15: MaxRequestsPerChild  0
16: </IfModule>
```

The `ServerRoot` Directive

The `ServerRoot` directive takes a single argument: a directory path pointing to the directory where the server lives. All relative path references in other directives are relative to the value of `ServerRoot`. If you compiled Apache from source on Linux/UNIX, as described earlier in this chapter, the default value of `ServerRoot` is `/usr/local/apache2`. The `ServerRoot` for Mac OS X users defaults to `/Library/WebServer`. If you used the Windows installer, the `ServerRoot` is `C:\Program Files\Apache Group`.

Per-Directory Configuration Files

Apache uses per-directory configuration files to allow directives to exist outside the main configuration file, `httpd.conf`. These special files can be placed in the filesystem. Apache will process the content of these files if a document is requested in a directory containing one of these files or any subdirectories under it. The contents of all the applicable per-directory configuration files are merged and processed. For example, if Apache receives a request for the `/usr/local/apache2/htdocs/index.html` file, it will look for per-directory configuration files in the `/`, `/usr`, `/usr/local`, `/usr/local/apache2`, and `/usr/local/apache2/htdocs` directories, in that order.

Watch Out!

> Enabling per-directory configuration files has a performance penalty. Apache must perform expensive disk operations looking for these files in every request, even if the files do not exist.

Per-directory configuration files are called `.htaccess` by default. This is for historical reasons; they were originally used to protect access to directories containing HTML files.

The directive `AccessFileName` enables you to change the name of the per-directory configuration files from `.htaccess` to something else. It accepts a list of filenames that Apache will use when looking for per-directory configuration files.

To determine whether a directive can be overridden in the per-directory configuration file, check whether the `Context:` field of the directive syntax definition contains `.htaccess`.

Apache directives belong to different groups, specified in the `Override` field in the directive syntax description. Possible values for the `Override` field are as follows:

- `AuthConfig`—Authorization directives
- `FileInfo`—Directives controlling document types
- `Indexes`—Directives controlling directory indexing
- `Limit`—Directives controlling host access
- `Options`—Directives controlling specific directory features

You can control which of these directive groups can appear in per-directory configuration files by using the `AllowOverride` directive. `AllowOverride` can also take an `All` or a `None` argument. `All` means that directives belonging to all groups can appear in the configuration file. `None` disables per-directory files in a directory and any of its subdirectories. Listing 3.4 shows how to disable per-directory configuration files for the server as a whole. This improves performance and is the default Apache configuration.

LISTING 3.4 Disabling Per-Directory Configuration Files

```
1: <Directory />
2: AllowOverride none
3: </Directory>
```

Apache Log Files

Apache includes two log files by default. The `access_log` file is used to track client requests. The `error_log` is used to record important events, such as errors or server restarts. These files don't exist until you start Apache for the first time. The files are named `access.log` and `error.log` in Windows platforms.

The `access_log` File

When a client requests a file from the server, Apache records several parameters associated with the request, including the IP address of the client, the document

requested, the HTTP status code, and the current time. Listing 3.5 shows sample log file entries. Chapter 26, "Logging and Monitoring Server Activity," will show you how to modify which parameters are logged.

LISTING 3.5 Sample `access_log` Entries

```
1: 127.0.0.1 - - [26/Jan/2006:09:43:37 -0700] "GET / HTTP/1.1" 200 1494
2: 127.0.0.1 - - [26/Jan/2006:09:43:40 -0700] "GET /manual/ HTTP/1.1" 200 10383
```

The `error_log` File

The `error_log` file includes error messages, startup messages, and any other significant events in the life cycle of the server. This is the first place to look when you have a problem with Apache. Listing 3.6 shows sample `error_log` entries.

LISTING 3.6 Sample `error_log` Entries

```
1: [Tue Jan 26 09:42:59 2006] [notice] Parent: Created child process -2245
2: [Tue Jan 26 09:42:59 2006] [notice] Child -2242: Child process is running
3: [Tue Jan 26 09:42:59 2006] [notice] Child -2242: Acquired the start mutex.
4: [Tue Jan 26 09:42:59 2006] [notice] Child -2242: Starting 250 worker threads.
```

Additional Files

The `httpd.pid` file contains the process ID of the running Apache server. You can use this number to send signals to Apache manually, as described in the next section.

The `scoreboard` file, present configuration files on Linux/UNIX Apache, is used by the process-based MPMs to communicate with their children.

In general, you do not need to worry about these files.

Apache-Related Commands

The Apache distribution includes several executables. This section covers only the server binary and related scripts. Chapter 25, "Restricting Access to Your Applications," and Chapter 29, "Apache Performance Tuning and Virtual Hosting," cover additional utilities included with the Apache distribution.

Apache Server Binary

The Apache executable is named `httpd` in Linux/UNIX and Mac OS X, and `apache.exe` in Windows. It accepts several command-line options, which are described in Table 3.1. You can get a complete listing of options by typing `/usr/`

`local/apache2/bin/httpd -h` on Linux/UNIX, by typing `/usr/sbin/httpd –h` on Mac OS X, or by typing `apache.exe -h` from a command prompt on Windows.

TABLE 3.1 `httpd` Options

Option	Meaning
`-D`	Allows you to pass a parameter that can be used for `<IfDefine>` section processing
`-l`	Lists compiled-in modules
`-v`	Shows version number and server compilation time
`-f`	Allows you to pass the location of `httpd.conf` if it is different from the compile-time default

After Apache is running, you can use the `kill` command on Linux/UNIX and Mac OS X to send signals to the parent Apache process. Signals provide a mechanism to send commands to a process. To send a signal, execute the following command:

```
# kill -SIGNAL pid
```

In this syntax, *`pid`* is the process ID and *`SIGNAL`* is one of the following:

- `HUP`—Stop the server
- `USR1` or `WINCH`—Graceful restart; which signal to use depends on the underlying operating system
- `SIGHUP`—Restart

If you make some changes to the configuration files and you want them to take effect, you must signal Apache that the configuration has changed. You can do this by stopping and starting the server or by sending a restart signal. This tells Apache to reread its configuration.

A normal restart can result in a momentary pause in service. A graceful restart takes a different approach: Each thread or process serving a client will keep processing the current request, but when it is finished, it will be killed and replaced by a new thread or process with the new configuration. This allows seamless operation of the Web server with no downtime.

On Windows, you can signal Apache using the `apache.exe` executable:

- `apache.exe -k restart`—Tells Apache to restart
- `apache.exe -k graceful`—Tells Apache to do a graceful restart
- `apache.exe -k stop`—Tells Apache to stop

You can access shortcuts to these commands in the Start menu entries that the Apache installer created. If you installed Apache as a service, you can start or stop Apache by using the Windows service interface: In Control Panel, select Administrative Tasks and then click the Services icon.

Apache Control Script

Although it is possible to control Apache on Linux/UNIX using the `httpd` binary, it is recommended that you use the `apachectl` tool. The `apachectl` support program wraps common functionality in an easy-to-use script. To use `apachectl`, type the following:

```
# /usr/local/apache2/bin/apachectl command
```

In this syntax, *command* can be `stop`, `start`, `restart`, or `graceful`. You can also edit the contents of the `apachectl` script to add extra command-line options.

Some OS distributions provide you with additional scripts to control Apache; please check the documentation included with your distribution.

Starting Apache for the First Time

Before you start Apache, you should verify that the minimal set of information is present in the Apache configuration file, `httpd.conf`. The following sections describe the basic information needed to configure Apache and to start the server.

Check Your Configuration File

You can edit the Apache `httpd.conf` file with your favorite text editor. In Linux/UNIX and Mac OS X, this probably means `vi` or `emacs`. In Windows, you can use Notepad or WordPad. You must remember to save the configuration file in plain text, which is the only format Apache will understand.

There are only two parameters that you might need to change to enable you to start Apache for the first time: the name of the server and the address and port to which it is listening. The name of the server is the one Apache will use when it needs to refer to itself (for example, when redirecting requests).

Apache can usually figure out its server name from the IP address of the machine, but this is not always the case. If the server does not have a valid DNS entry, you might need to specify one of the IP addresses of the machine. If the server is not connected to a network (you might want to test Apache on a standalone machine), you can use the value 127.0.0.1, which is the loopback address. The default port value is 80. You might need to change this value if there is already a server running in the machine at port

80, or if you do not have administrator permissions—on Linux/UNIX and Mac OS X systems, only the `root` user can bind to privileged ports (those with port numbers lower than 1024).

You can change both the listening address and the port values with the `Listen` directive. The `Listen` directive takes either a port number or an IP address and a port, separated by a colon. If only the port is specified, Apache will listen on that port at all available IP addresses in the machine. If an additional IP address is provided, Apache will listen at only that address and port combination. For example, `Listen 80` tells Apache to listen for requests at all IP addresses on port 80. `Listen 10.0.0.1:443` tells Apache to listen only at 10.0.0.1 on port 443.

The `ServerName` directive enables you to define the name the server will report in any self-referencing URLs. The directive accepts a DNS name and an optional port, separated by a colon. Make sure that `ServerName` has a valid value. Otherwise, the server will not function properly; for example, it will issue incorrect redirects.

On Linux/UNIX and Mac OS X platforms, you can use the `User` and `Group` directives to specify which user and group IDs the server will run as. The `nobody` user is a good choice for most platforms. However, there are problems in the HP-UX platform with this user ID, so you must create and use a different user ID, such as `www`.

Starting Apache

To start Apache on Linux/UNIX, change to the directory containing the `apachectl` script and execute the following command:

```
# /usr/local/apache2/bin/apachectl start
```

Mac OS X users can type the following at the prompt:

```
# /usr/sbin/httpd
```

To manually start Apache on Windows, click the Start Apache in Console link in the Control Apache Server section, within the Apache HTTP Server 2.0.55 program group in the Start menu. If you installed Apache as a service, you must start the Apache service instead.

If everything goes well, you can access Apache using a browser. A default installation page will be displayed, such as the one shown in Figure 3.5. If you cannot start the web server or an error page appears instead, please consult the "Troubleshooting" section that follows. Make sure that you are accessing Apache in one of the ports specified in the `Listen` directive—usually port 80 or 8080.

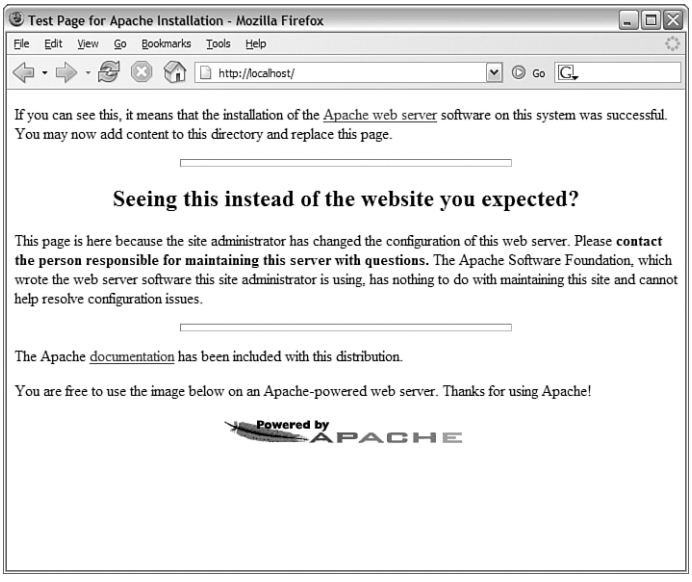

FIGURE 3.5
Apache default installation page.

Troubleshooting

The following subsections describe several common problems that you might encounter the first time you start Apache.

Already an Existing Web Server

If there is already a server running on the machine and listening to the same IP address and port combination, Apache will not be able to start successfully. You will get an entry in the error log file indicating that Apache cannot bind to the port:

```
[crit] (48)Address already in use: make_sock: could not bind to
➥address 10.0.0.2:80
[alert] no listening sockets available, shutting down
```

To solve this problem, you need to stop the running server or change the Apache configuration to listen on a different port.

No Permission to Bind to Port

You will get an error if you do not have administrator permissions and you try to bind to a privileged port (between 0 and 1024):

```
[crit] (13)Permission denied: make_sock: could not bind to address 10.0.0.2:80
[alert] no listening sockets available, shutting down
```

To solve this problem, you must either log on as the administrator before starting Apache or change the port number (8080 is a commonly used nonprivileged port).

Access Denied

You might not be able to start Apache if you do not have permission to read the configuration files or to write to the log files. You will get an error similar to the following:

```
(13)Permission denied: httpd: could not open error log file
    ➥ /usr/local/apache2/logs/error_log.
```

This problem can arise if Apache was built and installed by a different user than the one trying to run it.

Wrong Group Settings

You can configure Apache to run under a certain username and group. Apache has default values for the running server username and group. Sometimes the default value is not valid, and you will get an error containing `setgid: unable to set group id`.

To solve this problem on Linux/UNIX and Mac OS X, you must change the value of the `Group` directive in the configuration file to a valid value. Check the `/etc/groups` file for existing groups.

Summary

This chapter explained different ways of getting an Apache 2.0 server installed and running on your Linux/UNIX, Mac OS X, or Windows machine. It covered both binary and source installation and explained the basic build-time options. Additionally, you learned the location of the server configuration files and the syntax of the commands used to modify your Apache configuration. You learned about the two main log files: `access_log` and `error_log`. Finally, you saw how to start and stop the server using the Apache control scripts or the Apache server binary on Linux/UNIX, Mac OS X, and Windows platforms.

Q&A

Q ***How can I start a clean build?***

A If you need to build a new Apache from source and do not want the result of earlier builds to affect the new one, it is always a good idea to run the `make clean` command. That will take care of cleaning up any existing binaries, intermediate object files, and so on.

Q ***Why are per-directory configuration files useful?***

A Although per-directory configuration files have an effect on server performance, they can be useful for delegated administration. Because per-directory configuration files are read every time a request is made, there is no need to restart the server when a change is made to the configuration.

You can allow users of your website to make configuration changes on their own without granting them administrator privileges. In this way, they can password-protect sections of their home pages, for example.

Q *What do you mean by a valid `ServerName` directive?*

A The DNS system is used to associate IP addresses with domain names. The value of `ServerName` is returned when the server generates a URL. If you are using a certain domain name, you must make sure that it is included in your DNS system and will be available to clients visiting your site.

Workshop

The workshop is designed to help you anticipate possible questions, review what you've learned, and begin putting your knowledge into practice.

Quiz

1. How can you specify the location where you want to install Apache?
2. What is the main difference between `<Location>` and `<Directory>` sections?
3. What is the difference between a restart and a graceful restart?

Answers

1. Linux/UNIX users can use the `--prefix` option of the `configure` script. If an existing installation is present at that location, the configuration files will be preserved but the binaries will be replaced. On Windows, this location is set in the installation wizard.
2. `Directory` sections refer to file system objects; `Location` sections refer to elements in the address bar of the web page.
3. During a normal restart, the server is stopped and then started, causing some requests to be lost. A graceful restart allows Apache children to continue to serve their current requests until they can be replaced with children running the new configuration.

Activities

1. Practice the various types of server shutdown and restart procedures.
2. Make some configuration changes, such as different port assignments and `ServerName` changes.