

 Search the Catalog Learning Debian
GNU/Linux

Learning Debian GNU/Linux

By [Bill McCarty](#)

1st Edition September 1999

1-56592-705-2, Order Number: 7052

360 pages, \$34.95 , Includes CD-ROM

◀ PREVIOUS

Chapter 12
Setting Up a Linux-Based
WAN

NEXT ▶

12.2 Installing and Configuring a Web Server

Installing and configuring a web server is not much more difficult than installing an FTP server. Once your web server is up and running, other Internet users can view documents you publish on your Linux system.

12.2.1 Configuration

Configuring a web server can be as easy or as difficult as you choose. Like other web servers, Apache provides seemingly countless options. Fortunately, Debian GNU/Linux automatically configures Apache when you install it. However, the configuration options selected by the install scripts may not suit your needs. In that case, you can modify the configuration files.

Apache's configuration files reside in the directory */etc/apache*. For historical reasons that no longer apply, Apache has three configuration files:

`access.conf`

Specifies what hosts and users are allowed access to what documents and services

`httpd.conf`

Specifies options that govern the operation of the `httpd` daemon

`srm.conf`

Specifies how your server's documents are organized and formatted

Currently, you can place Apache configuration commands in any of these files. However, each of these files must exist, even if it is empty; otherwise, the `httpd` daemon will refuse to run. As distributed, the files contain a default configuration. Before starting the web server, you should revise the `ServerName` option of the *httpd.conf* file. The three following subsections describe other options that you may wish to specify. You can scan them to see what options are available and specify options that interest you. A more complete description of the options is available in Apache's online documentation. Also, the Apache web site (<http://www.apache.org/>) provides a tutorial on Apache configuration. To change an option, simply open the related file by using your favorite text editor, change the file as you wish, and save the file.

Bright Data Ltd. v. Oxylabs, UAB
IPR2023-01441 | Oxylabs EX2005

The subsections assume some familiarity with HTML and web servers. If you find that some options are obscure, don't fret; your web server will serve ordinary HTML pages even if you set no options other than `ServerName`.

12.2.1.1 The `access.conf` file

The `access.conf` file specifies a default set of permissions that govern access to documents and services. It then specifies sets of permissions that override the default permissions for particular documents and services. The usual practice is to specify a quite restrictive set of default permissions, relaxing these permissions to provide access to particular documents and services.

The file contains a mixture of comments (lines beginning with `#`) and directives. Comments are ignored by the server.

The default permissions are specified as follows:

```
<Directory />
Options None
AllowOverride None
</Directory>
```

The paired tags `<Directory>` and `</Directory>` enclose a list of options that pertain to the / directory, the directory specified in the `<Directory>` tag. The options are:

- `Options None`, which specifies that no special server features are enabled for the specified directory or its subdirectories.
- `AllowOverride None`, which specifies that access specifications cannot be overridden by an `.htaccess` file.

[Table 12.2](#) describes special server features that are available.

Table 12.2: Special Server Features

Option	Description
<code>ExecCGI</code>	Execution of CGI scripts is permitted in this directory.
<code>FollowSymLinks</code>	The server will follow symbolic links in this directory.
<code>Includes</code>	Server-side includes are permitted.
<code>IncludesNOEXEC</code>	Server-side includes, except <code>#exec</code> and <code>#include</code> , are permitted in this directory.
<code>Indexes</code>	If the directory contains no index file (for example, <code>index.html</code>), the server will prepare a formatted index.
<code>MultiViews</code>	Content-negotiated MultiViews are permitted in this directory. MultiViews permit, for example, a client browser to select a document in a particular language from a set of documents.
<code>SymLinksIfOwnerMatch</code>	The server will follow symbolic links for which the target file or directory has the same owner as the link.

Unless the specification for a directory specifies `AllowOverride None`, you can override the specified options by placing an `.htaccess` file in the directory or one of its subdirectories. The `.htaccess` file

can contain specifications of the same sort as the *access.conf* file; the server applies the specifications in the *.htaccess* file in preference to those specified in the *access.conf* file.

After the restrictive default specifications come some more relaxed specifications:

```
<Directory /home/httpd/html>
Options Indexes Includes FollowSymLinks
AllowOverride None
order allow,deny
allow from all
</Directory>
```

These specifications apply to the directory */home/httpd/html* and its subdirectories. Here, the *Indexes*, *Includes*, and *FollowSymLinks* options are specified. As for the root directory, use of *.htaccess* files is forbidden, via *AllowOverride None*. Here, unlike the specification for the root directory, the hosts allowed to access documents and services are specified. The *order allow,deny* directive specifies that any *deny* directives will be applied after any *allow* directives, and will therefore take precedence. No *deny* directives appear in this specification; the *allow from all* directive permits any host to access documents and services within the */home/httpd/html* directory and its subdirectories.

Another specification allows execution of CGI scripts within the */home/httpd/cgi-bin* directory and its subdirectories:

```
<Directory /home/httpd/cgi-bin>
AllowOverride None
Options ExecCGI
</Directory>
```

The final specification in the default configuration lets the local host access HTML documents within the */usr/doc* directory and its subdirectories:

```
<Directory /usr/doc>
order deny,allow
deny from all
allow from localhost
Options Indexes FollowSymLinks
</Directory>
```

12.2.1.2 The *srm.conf* File

The *srm.conf* file specifies the organization and format of documents provided by your web server. As was the case with the *access.conf* file, you don't need to make any changes to the *srm.conf* file, though you may wish to do so.

The *DocumentRoot* directive specifies the directory that contains your HTML files. When a web client accesses the root directory, the server actually fetches files from the directory specified as *DocumentRoot*:

```
DocumentRoot /home/httpd/html
```

The *UserDir* directive specifies the name of the subdirectory that the server appends to a user's home directory when a client makes a *~user* request:

```
UserDir public_html
```

This directive specifies that a reference to *~user* will be translated to a reference to */home/user/public_html*.

The `DirectoryIndex` directive specifies the name of the file (or names of the files) used a directory indexes:

```
DirectoryIndex index.html index.shtml index.cgi
```

The `FancyIndexing` directive specifies whether icons are used to produce fancy directory indexes:

```
FancyIndexing on
```

The `AddIcon` and `AddIconByType` directives associate icons with files of given types:

```
AddIconByType (TXT,/icons/text.gif) text/*  
AddIcon /icons/binary.gif .bin .exe
```

The default configuration includes many such directives. If you wish to add support for a new type of file, you may want to add a directive associating an icon with the new file type.

The `DefaultIcon` directive specifies the icon used for file types not explicitly associated with an icon:

```
DefaultIcon /icons/unknown.gif
```

The `ReadmeName` directive specifies the name of a file used by the server to produce readme entries:

```
ReadmeName README
```

The server will first look for the file `README.html` and then for the file `README`.

The similar `HeaderName` directive specifies the name of a file that the server will prepend to a generated index:

```
HeaderName HEADER
```

The `IndexIgnore` directive specifies a set of file names that should not be included in a generated index. These are often specified by using wildcard characters:

```
IndexIgnore .?}* *~ *# HEADER* README* RCS
```

The `AccessFileName` directive specifies the name of the file that, if present, overrides access control specifications for a directory:

```
AccessFileName .htaccess
```

The `TypesConfig` directive identifies the *mime.types* file, which describes multimedia files:

```
TypesConfig /etc/mime.types
```

The `DefaultType` directive specifies the default MIME type for documents:

```
DefaultType text/plain
```

The `AddEncoding` directive instructs compatible browsers to uncompress information as it's downloaded:

```
AddEncoding x-compress Z  
AddEncoding x-gzip gz
```

The `Redirect` directive lets you provide a forwarding address for documents that have moved. The default configuration includes no Redirects, which have this simple form:

```
Redirect old-URL new-URL
```

The `Alias` directive lets you refer to a directory by using an alias. For example, the following directive provides a more convenient way of referring to the </home/httpd/icons> directory:

```
Alias /icons/ /home/httpd/icons/
```

The `ScriptAlias` directive lets users refer to the CGI directory as simply </cgi-bin/>:

```
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
```

The following sets of specifications work around problems with several browsers. First, two directives that disable keepalive for browsers that do not support it:

```
BrowserMatch "Mozilla/2" nokeepalive  
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
```

The final directives force use of HTTP version 1.0 responses for several browsers that do not support HTTP 1.1:

```
BrowserMatch "RealPlayer 4\.0" force-response-1.0  
BrowserMatch "Java/1\.0" force-response-1.0  
BrowserMatch "JDK/1\.0" force-response-1.0
```

12.2.1.3 The `httpd.conf` File

The `httpd.conf` file specifies options related to the `httpd` daemon. You should specify the `ServerName` option before starting your web server.

The `ServerType` directive specifies whether the web server is started via `inetd` or standalone:

```
ServerType standalone
```

The `port` directive specifies the port on which the web server listens for client requests:

```
Port 80
```

The `HostnameLookups` directive specifies whether clients are logged by IP address (off) or hostname (on):

```
HostnameLookups off
```

The `User` and `Group` directives specify the `userid` and `group` under which the `httpd` daemon runs. The daemon initially runs as `root` and then switches to the specified `userid` and `group`. The default configuration specifies the `userid` as `nobody`, a standard Unix `userid` that has very limited permissions. A user or process running as `nobody` can access files only in ways permitted to all users; generally, this means the user or process cannot modify files. The `group nobody` has similarly constrained privileges:

```
User nobody  
Group nobody
```

The `ServerAdmin` directive specifies the email address of the server administrator:

```
ServerAdmin root@localhost
```

The `ServerRoot` directive specifies the directory that contains the configuration, error, and log files:

```
ServerRoot /etc/httpd
```

The `BindAddress` directive provides for virtual hosts. It specifies the IP address to which the server should listen. It is normally disabled by a comment token:

```
#BindAddress *
```

The `ErrorLog` directive specifies the location of the error log file:

```
ErrorLog logs/error_log
```

The `LogLevel` directive specifies the verbosity of the server log. Possible values include `debug`, `info`, `notice`, `warn`, `error`, `crit`, `alert`, and `emerg`:

```
LogLevel warn
```

The `LogFormat` directive specifies format names that can be used with the `CustomLog` directive:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access_log common
```

For more information about `LogFormat` and `CustomLog`, see the Apache documentation.

The `PidFile` directive specifies the file in which the server should log its process id:

```
PidFile /var/run/httpd.pid
```

The `ScoreBoardFile` directive specifies the name of the file used to store internal server process data:

```
ScoreBoardFile /var/run/httpd.scoreboard
```

The `LockFile` directive specifies the file used to providing locking. You'll need to change this option only if you NFS mount the directory used by the server for logging:

```
#LockFile /var/lock/httpd.lock
```

The `ServerName` directive specifies the hostname of the system on which your server runs. Depending on your network configuration, you may not need to specify this directive:

```
ServerName host.domain.com
```

The `UseCanonicalName` directive specifies whether the server will return a canonical URL formed from the `ServerName` and `Port` directives (on) or the hostname and port supplied by the client (off):

```
UseCanonicalName on
```

The `CacheNegotiatedDocs` directive instructs browsers not to cache documents; it is usually disabled by prefixing it with a comment token (`#`):

```
#CacheNegotiatedDocs
```

The `Timeout` directive specifies the maximum number of seconds the server will wait for certain responses, such as the next packet in a sequence of TCP packets:

```
Timeout 300
```

The `KeepAlive` directive specifies that connections are persistent; that is, that a client can make multiple requests per connection:

```
KeepAlive On
```

The `MaxKeepAliveRequests` directive specifies the maximum number of requests permitted during a persistent connection:

```
MaxKeepAliveRequests 100
```

The value 0 denotes an unlimited number of requests.

The `KeepAliveTimeout` directive specifies the maximum number of seconds during which the server will wait for the next request:

```
KeepAliveTimeout 15
```

The `MinSpareServers` and `MaxSpareServers` directives respectively fix the minimum and maximum number of spare server processes the server will create. Having an available server expedites handling of an incoming request:

```
MinSpareServers 8  
MaxSpareServers 20
```

The `StartServers` directive specifies the initial number of server processes:

```
StartServers 10
```

The `MaxClients` directive specifies the maximum number of simultaneous server processes. When this number is reached, requests from additional clients are locked out:

```
MaxClients 150
```

The default values of these options are higher than necessary for most workstation users. Your system will perform more efficiently (and perhaps more reliably) if you change the directives to specify the following values:

```
MinSpareServers 3  
MaxSpareServers 6  
StartServers 3  
MaxClients 20
```

The `MaxRequestsPerChild` directive specifies the number of requests a child process can handle before expiring. This ensures that processes are periodically recreated, minimizing problems due to software errors such as memory leaks:

```
MaxRequestsPerChild 100
```

The `ProxyRequests` directive specifies whether the proxy server is enabled; it is normally disabled:

```
#ProxyRequests On
```

When the proxy server is active, the following directives specify various caching options. They are normally disabled:

```
#CacheRoot /var/cache/httpd  
#CacheSize 5  
#CacheGcInterval 4  
#CacheMaxExpire 24
```

```
#CacheLastModifiedFactor 0.1
#CacheDefaultExpire 1
#NoCache a_domain.com another_domain.edu joes.garage_sale.com
```

The `Listen` directive lets you bind Apache to a specific IP address or port, in addition to the default IP address and port. It is generally disabled:

```
#Listen 3000
#Listen 12.34.56.78:80
```

The `<VirtualHost>` and `</VirtualHost>` tags enclose a series of options that establish a virtual host, useful if your system has multiple IP addresses. The options can include any of the options described in this subsection. The tags and options are normally disabled:

```
#<VirtualHost host.some_domain.com>
#ServerAdmin webmaster@host.some_domain.com
#DocumentRoot /www/docs/host.some_domain.com
#ServerName host.some_domain.com
#ErrorLog logs/host.some_domain.com-error_log
#TransferLog logs/host.some_domain.com-access_log
#</VirtualHost>
```

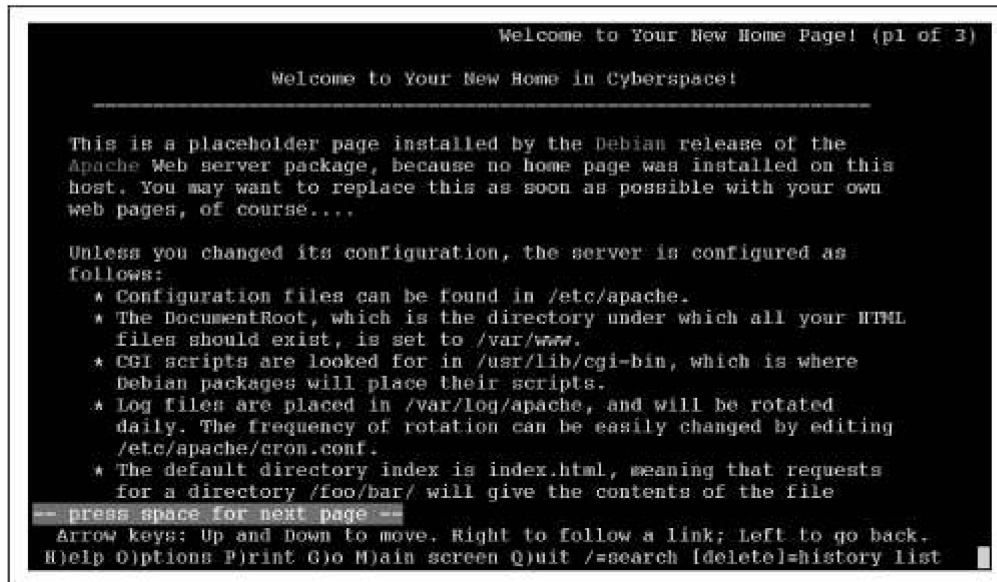
12.2.2 Startup and Use

The install script will automatically start your web server as soon as you install it. If you installed the `lynx` browser, you can use it to test your web server. Issue the command:

```
lynx http://localhost
```

You should see a screen that resembles [Figure 12.1](#).

Figure 12.1: The Apache start page viewed by lynx



If you prefer, you can view the start page by using Netscape Navigator, which supports the graphics embedded in the page. The result should resemble [Figure 12.2](#).

Figure 12.2: The Apache start page viewed by Netscape Navigator



Once you can access your web server locally, try accessing it from a remote computer. This should be as simple as forming a URL that includes the fully qualified hostname of your system (that is, the host and domain names); for example, <http://mysystem.mydomain>.

[◀ PREVIOUS](#)
[12.1 An FTP Server](#)
[HOME](#)
[BOOK INDEX](#)
[NEXT ▶](#)
[12.3 Using a Mail Server](#)

Back to: [Learning Debian GNU/Linux](#)

[oreilly.com Home](#) |
 [O'Reilly Bookstores](#) |
 [How to Order](#) |
 [O'Reilly Contacts International](#) |
 [About O'Reilly](#) |
 [Affiliated Companies](#) |
 [Privacy Policy](#)

© 2001, O'Reilly & Associates, Inc.