By Vikram Vaswani

Table of Contents

Hooking Up

If you've been using the Internet for any length of time, you already know about FTP, the File Transfer Protocol. Built on the TCP/IP protocol and having a pedigree dating back to the early days of the Internet, FTP has long been one of the most reliable ways of transferring files from one host to another on the Internet - even today, many years after the Internet went mainstream, it's still in use, by multinational corporations and home users alike, as a means of secure and robust file distribution.

Now, setting up an FTP server to make files available over a network isn't very difficult at all - in fact, it's pretty simple, so long as you keep a few basic rules in mind. Over the course of this article, I'm going to show you what I mean by taking you through the process of installing and configuring an FTP server for your internal (or external) network. Regardless of whether you're looking to simplify file sharing between users on your LAN, distribute software to Webheads across the globe, or just tinker around with a new piece of code, you're bound to find something interesting in here.

The Sales Pitch

There are numerous FTP servers available on the Web, and they come in many different flavours. Some are for Windows and some are for *NIX, some are feature-packed and others are stripped down to just the basics, some are open-source and others have to be paid for...the permutations go on and on. For our little experiment here today, though, I'm going to be using the proFTPD server, a free, open-source implementation of the FTP protocol that also happens to be one of the most widely used, stable, easy-to-use and secure FTP servers available today.

Developed by a team of open-source programmers, proFTPD offers numerous advantages over its brethren. It's extremely simple to compile, install and configure, supports a wide range of platforms, and has excellent documentation and help files. In addition to supporting the FTP protocol defined in RFC 959, it also comes with numerous additional features, including Apache-style configuration files, the ability to hide files and directories, resumption of broken downloads and file and directory aliases. Don't think all this comes at the cost of stability, though - proFTPD is so robust that extremely popular sites like SourceForge, Slackware, LinkSys and kernel.org use it to power their FTP services.

Anonymous FTP is a service offered by many sites that host software for download. In an anonymous FTP system, users can log in to the system using the special "anonymous" account and gain access to the files stored there for download. proFTPD supports anonymous FTP out of the box, and also comes with built-in support for virtual hosting and restricted guest accounts.

Keeping in mind that activating an FTP server opens a back door into the system, proFTPD comes with numerous features designed to ensure that the security of your system is not breached, including logging of all transfers and setting access levels on a per-directory basis. Users can be "jailed" in specific directories to minimize the impact of any damage they may cause, and all commands can be logged to maintain an audit trail of user activity.

Intrigued? Wanna see it in action? Flip the page, and let's get installing!

Source Control

The first order of business to install proFTPD on the Linux box you plan to use as a server. Drop by the official proFTPD Web site at http://www.proftpd.org/and get yourself the latest stable release of the software (this tutorial uses proFTPd 1.2.8).
Once you've downloaded the source code archive to your Linux server (mine is named "olympus.melonfire.com"), log in as "root"

```
$ su -
Password: ****
```

and extract the source to a temporary directory.

```
$ cd /tmp
$ tar -xzvf /home/me/proftpd-1.2.8.tar.gz
```

Next, configure the package using the provided "configure" script,

```
$ cd /tmp/proftpd-1.2.8
$ ./configure --prefix=/usr/local/ftpd
```

and compile and install it.

```
$ make
$ make install
```

Unless you specified a different path to the "configure" script, proFTPD will have been installed to the directory "/usr/local/ftpd".
You can verify this by doing a quick directory scan of that directory - here's what you should see.

```
$ ls -lR /usr/local/ftpd/{bin,etc,sbin}
/usr/local/ftpd/bin:
total 28
-rwxr-xr-x 1 root root 8108 May 7 10:24 ftpcount
-rwxr-xr-x 1 root root 4940 May 7 10:24 ftptop
-rwxr-xr-x 1 root root 11552 May 7 10:24 ftpwho
/usr/local/ftpd/etc:
total 4
-rw-r--r-- 1 root root 1817 May 7 10:24 proftpd.conf
/usr/local/ftpd/sbin:
```

```
total 320
-rwxr-xr-x 1 root root 5356 May 7 10:24 ftpshut
lrwxrwxrwx 1 root root 7 May 7 10:24 in.proftpd ->
proftpd
-rwxr-xr-x 1 root root 313312 May 7 10:24 proftpd
```

Once you've got proFTPd installed, the next step is to configure it. Let's look at that next.

Start Me Up

Unlike its competitors, some of which require four or more configuration files, proFTPd is controlled via a single configuration file, usually located in "/usr/local/ftpd/etc/proftpd.conf". Pop open this file, and put the following lines of code into it:

```
# set server parameters
ServerName "ProFTPD"
ServerAdmin "admin@localnet.com"
ServerType standalone
Port 21
# set default umask
Umask 022
# set the user and group for the server process
User nobody
Group nobody
```

If you've ever used the Apache Web server, you'll notice a marked similarity between the configuration file above and Apache's "httpd.conf" file. This is no accident - proFTPD was inspired by the Apache approach, and uses a similar directive-based technique to configure the server.
Most of the directives above should be fairly self-explanatory - the ServerName directive sets the name of the server (as displayed to connecting users), the ServerAdmin directive sets the email address of the server's administrator and the Port directive sets the port the server will run on. The ServerType directive specifies whether the server is active at all times, or whether it is awakened on demand from the "inetd" daemon. The User and Group directives set the user and group owning the server process, while the Umask directive sets the file mask for files and directories created by FTP users.
With the configuration completed, how about starting the server and playing with it a little?

```
$ /usr/local/ftpd/sbin/proftpd
```

If all goes well, proFTPd should start up and run in the background as a daemon.
You can verify this by checking the list of running processes:

```
$ ps ax | grep proftpd
28429 ? S 0:00 proftpd: (accepting connections)
```

You can also test if the daemon is, in fact, running by a quick telnet to port 21 (the default FTP port) - you should see something like this:

```
$ telnet localhost 21
Trying 127.0.0.1...
```

```
Connected to localhost.
Escape character is '^]'.
220 ProFTPD 1.2.8 Server (ProFTPD) [olympus.melonfire.com]
telnet> quit
Connection closed.
```

Now, how about logging in and messing around a bit?

Going Home

Your Linux system should already have an FTP client installed - start it up, give it the name of the server to connect to ("localhost", in this example), and log in as a user with an existing account on the system.

```
$ ftp localhost
Connected to localhost (127.0.0.1).
220 ProFTPD 1.2.8 Server (ProFTPD) [olympus.melonfire.com] Name
(localhost:joe): joe 331 Password required for joe.
Password: *******
230 User joe logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

How about looking around a little?

```
ftp> ls
227 Entering Passive Mode (127,0,0,1,4,139).
150 Opening ASCII mode data connection for file list
-rw------- 1 joe joe 9144 May 7 05:47 mbox
-rw-rw-r-- 1 joe joe 966281 May 7 04:44 proftpd-1.2.8.tar.gz
226 Transfer complete.
ftp>
```

Let’s see if you can upload and download files.

```
ftp> bin
200 Type set to I.
ftp> get mbox
local: mbox remote: mbox
227 Entering Passive Mode (127,0,0,1,4,143).
150 Opening BINARY mode data connection for mbox (9144 bytes)
226 Transfer complete.
9144 bytes received in 0.00042 secs (2.1e+04 Kbytes/sec)
ftp> put outfile
local: outfile remote: outfile
227 Entering Passive Mode (127,0,0,1,4,145).
150 Opening BINARY mode data connection for outfile
226 Transfer complete.
ftp>
```

How about moving around the file system?

```
ftp> cd /
250 CWD command successful.
ftp> ls -l
227 Entering Passive Mode (127,0,0,1,36,136)
150 Opening ASCII mode data connection for /bin/ls.
total 193
drwxr-xr-x 2 root 4096 Apr 28 15:33 bin
drwxr-xr-x 4 root 1024 Apr 28 17:32 boot
drwxr-xr-x 20 root 118784 May 6 11:52 dev
drwxr-xr-x 41 root 4096 May 6 16:46 etc
drwxr-xr-x 17 root 4096 May 6 16:47 home
drwxr-xr-x 2 root 4096 Jun 22 2001 initrd
drwxr-xr-x 7 root 4096 Apr 28 17:29 lib
drwx------ 2 root 16384 Apr 28 17:17 lost+found
drwxr-xr-x 2 root 4096 Aug 27 2002 misc
drwxr-xr-x 4 root 4096 Apr 28 12:03 mnt
drwxr-xr-x 2 root 4096 Aug 23 1999 opt
dr-xr-xr-x 80 root 0 May 6 2003 proc
drwxr-x--- 5 root 4096 May 3 18:23 root
drwxr-xr-x 2 root 8192 Apr 28 17:31 sbin
drwxrwxrwt 3 root 4096 May 6 16:20 tmp
drwxr-xr-x 15 root 4096 Apr 28 17:19 usr
drwxr-xr-x 18 root 4096 May 5 17:42 var
226 Transfer complete.
ftp> cd /home
250 CWD command successful.
ftp> ls -l
227 Entering Passive Mode (127,0,0,1,4,157).
150 Opening ASCII mode data connection for file list
drwx------ 4 joe users 4096 Apr 28 11:02 joe
drwx------ 4 john users 4096 May 5 09:32 john
drwx------ 4 sarah users 4096 Jan 26 16:12 sarah
226 Transfer complete.
ftp>
```

All done? Log out.

```
ftp>bye
221 Goodbye.
```

Now, you'll have seen, from the above demonstration, that the logged-in user can not only view his or her home area, but also other parts of the directory tree. Since this is generally considered a serious security hole, the first order of business is to configure proFTPD to "jail" users to their home area and prevent them from moving around the rest of the system. Luckily, doing this is fairly simple - just add the lines

```
# jail users to their home areas
DefaultRoot ~
```

to your "proftpd.conf" file, and restart the server.

```
$ killall -HUP proftpd
```

Now, try logging in again. This time, when you attempt to move up and out of your home area, you'll see that proFTPD does not permit you to do this.

```
$ ftp localhost
Connected to localhost (127.0.0.1).
220 ProFTPD 1.2.8 Server (ProFTPD) [olympus.melonfire.com] Name
(localhost:joe): joe 331 Password required for joe.
Password: *******
230 User joe logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
Using binary mode to transfer files.
ftp> ls -l
227 Entering Passive Mode (127,0,0,1,4,161).
150 Opening ASCII mode data connection for file list
-rw------- 1 joe joe 9144 May 7 05:47 mbox
-rw-rw-r-- 1 joe joe 966281 May 7 04:44 proftpd-1.2.8.tar.gz
226 Transfer complete.
ftp> pwd
257 "/" is current directory.
ftp> cd /
250 CWD command successful.
ftp> ls -l
227 Entering Passive Mode (127,0,0,1,4,163).
150 Opening ASCII mode data connection for file list
-rw------- 1 joe joe 9144 May 7 05:47 mbox
-rw-rw-r-- 1 joe joe 966281 May 7 04:44 proftpd-1.2.8.tar.gz
226 Transfer complete.
ftp> cd /home
550 /home: No such file or directory
```

```
ftp> cd /bin
550 /bin: No such file or directory
ftp> bye
221 Goodbye.
```

Just incidentally, proFTPD’s default settings do not allow "root" to log in, even with the correct password. The reason is that "root" is just too powerful a user to be permitted access via FTP; permitting "root" login opens up a security hole that might be exploited by determined hackers to gain super-user access to the system.
If, despite the warning above, you still want to allow "root" login to the FTP server, you can do so by adding the RootLogin directive to the configuration file, as below:

```
RootLogin on
```

Simple, huh? Now, how about setting up the server to handle anonymous FTP.

Signed, Anonymous

At this point in time, your proFTPD server is not configured for anonymous FTP - a fact amply demonstrated by the output of the server when you try to log in as the special "ftp" or "anonymous" users. Typically, the "ftp" user on the system is configured without a password - which makes it impossible for any user to log in at this time. Therefore, the server needs to be configured to recognize the "ftp" and "anonymous" users, and grant them access to the appropriate public area on the server.
Setting up an anonymous FTP server with proFTPD is simplicity itself - all you need to do is use the special <Anonymous>...</Anonymous> block, which contains configuration parameters for the operation of your FTP server.
In order to enable anonymous FTP, pop open your "proftpd.conf" file and add the following code block to it:

```
# set root directory for anonymous users to /home/ftp
<Anonymous/home/ftp>
# set the user and group for the server process
User ftp
Group ftp
# alias "anonymous" login to "ftp"
UserAlias anonymous ftp
# restrict "anonymous" users from writing data
<Directory *>
<Limit WRITE>
DenyAll
</Limit>
</Directory>
</Anonymous>
```

Obviously, you should make sure that an entry exists for the user "ftp" in your system’s password file, and that the directory "/home/ftp" exists before activating this configuration. Once you’re done that, restart the server and try logging in again as an anonymous user.

```
$ ftp localhost
Connected to localhost (127.0.0.1).
220 ProFTPD 1.2.8 Server (ProFTPD) [olympus.melonfire.com] Name
(localhost:joe): ftp 331 Anonymous login ok, send your complete
email
address as your password.
Password: *******
230 Anonymous access granted, restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (127,0,0,1,4,199).
```

```
150 Opening ASCII mode data connection for file list
drwxr-xr-x 3 ftp ftp 4096 Apr 28 06:45 pub
226 Transfer complete.
ftp> cd pub
250 CWD command successful.
ftp> ls
227 Entering Passive Mode (127,0,0,1,4,207).
150 Opening ASCII mode data connection for file list
-r-xr-xr-x 1 ftp ftp 8820072 Jul 15 2002 ar500enu.exe
226 Transfer complete.
ftp> get ar500enu.exe
local: ar500enu.exe remote: ar500enu.exe
227 Entering Passive Mode (127,0,0,1,4,209).
150 Opening BINARY mode data connection for ar500enu.exe (8820072
bytes) 226
Transfer complete. 8820072 bytes received in 2.05 secs (4.2e+03
Kbytes/sec)
ftp> put mbox
local: mbox remote: mbox
227 Entering Passive Mode (127,0,0,1,4,211).
550 mbox: Permission denied
ftp> cd /
250 CWD command successful.
ftp> cd ..
250 CWD command successful.
ftp> pwd
257 "/" is current directory.
ftp> bye
221 Goodbye.
```

As you can see, the system now gives you access and locates you in the "/home/ftp" directory. You have the ability to download existing files, though not upload new ones, and you can move around freely in the "/home/ftp" hierarchy, but not outside it. In other words, your basic anonymous FTP, good to go!

Giving Back

By default, only users with real accounts on the system are allowed to upload files to the FTP server - and even they are limited to uploads in their home area. Anonymous users, as demonstrated on the previous page, do not have the ability to upload files to the server. This is a fairly reasonable safety precaution if your server is exposed to the public Internet, because you never know what malicious files might be uploaded to your server; however, if your FTP server is running on a closed network, you might want to enable file upload for anonymous users also (perhaps to enable file sharing between users at different locations).
In order to do this, you should update the <Anonymous>...</Anonymous> block to look like this:

```
# set root directory for anonymous users to /home/ftp
<Anonymous/home/ftp>
# set the user and group for the server process
User ftp
Group ftp
# alias "anonymous" login to "ftp"
UserAlias anonymous ftp
# restrict "anonymous" users from writing data
<Directory *>
<Limit WRITE>
DenyAll
</Limit>
</Directory>
# allow writes to the /home/ftp/incoming directory
# but do not allow reads
<Directory incoming>
<Limit READ WRITE>
DenyAll
</Limit>
<Limit STOR>
AllowAll
</Limit>
</Directory>
</Anonymous>
```

In case you haven’t yet figured it out, the <Limit>...</Limit> block places restrictions on the commands that can be executed by a user, with the AllowAll and DenyAll directives specifying whether all clients can or cannot use those commands. For finer-grained control, proFTPD also provides the Allow and Deny directives, which permit you to set allow/deny rules on the basis of host or network name, rather than globally for all clients.
Now, create a directory in "/home/ftp" named "incoming", restart the server and try uploading a file anonymously into that directory:

```
$ ftp localhost
Connected to localhost(192.168.3.1).
220 ProFTPD 1.2.8 Server (ProFTPD) [olympus.melonfire.com] Name
(localhost:joe): ftp 331 Anonymous login ok, send your complete
email
address as your password.
Password: *******
230 Anonymous access granted, restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,3,1,4,239).
150 Opening ASCII mode data connection for file list
drwxr-xr-x 2 ftp ftp 4096 May 7 08:41 incoming
drwxr-xr-x 3 ftp ftp 4096 Apr 28 06:45 pub
226 Transfer complete.
ftp> cd incoming
250 CWD command successful.
ftp> ls -l
227 Entering Passive Mode (192,168,3,1,4,227).
150 Opening ASCII mode data connection for file list
226 Transfer complete.
ftp> put mbox
local: mbox remote: mbox
227 Entering Passive Mode (192,168,3,1,4,231).
150 Opening BINARY mode data connection for mbox
226 Transfer complete.
9144 bytes sent in 0.00124 secs (7.2e+03 Kbytes/sec)
ftp> ls
227 Entering Passive Mode (192,168,3,1,4,235).
150 Opening ASCII mode data connection for file list
-rw-r--r-- 1 ftp ftp 9144 May 7 08:41 mbox
226 Transfer complete.
ftp> put mbox
local: mbox remote: mbox
227 Entering Passive Mode (192,168,3,1,4,237).
550 mbox: Overwrite permission denied
ftp> cd /
250 CWD command successful.
ftp> put mbox
local: mbox remote: mbox
227 Entering Passive Mode (192,168,3,1,4,233).
550 mbox: Permission denied
ftp> bye
221 Goodbye.
```

As you can see, proFTPD now permits uploads, but only into the "incoming" directory; uploads to any location elsewhere in the anonymous FTP area will fail. Additionally, anonymous users cannot download files from the "incoming" directory.
In case you'd like a slightly greater level of security for your anonymous FTP server, you can force anonymous users to provide an actual password to gain entry, rather than the default anything-goes email address. In order to enable this, simply add the

```
AnonRequirePassword on
```

line inside the <Anonymous>...</Anonymous> block of your configuration file, and proFTPD will only allow anonymous access to users who enter the system password for the "ftp" user.

Timberrrrrrrrr!

By default, proFTPD logs all messages to the system logger, usually "/var/log/messages". You can alter the level of logging via the SystemLog directive,

```
# set system logging level
SyslogLevel debug
```

or even tell proFTPD to log all file transfers by means of the TransferLog directive.

```
# set transfer log
TransferLog /var/log/xferlog
```

With this configuration, proFTPD logs all transfers to "/var/log/xferlog". This data may then be read by an automated tool to calculate transfer statistics and bandwidth usage.
Here's a snippet of the system log,

```
May 7 17:03:58 olympus proftpd[7616]: olympus.melonfire.com
(localhost.localdomain[127.0.0.1]) - FTP session closed.
May 7 17:06:03 olympus proftpd[11531]: olympus.melonfire.com
(localhost.localdomain[127.0.0.1]) - FTP session opened.
May 7 17:06:10 olympus proftpd[11531]: olympus.melonfire.com
(localhost.localdomain[127.0.0.1]) - FTP session closed.
May 7 17:06:22 olympus proftpd[4096]: olympus.melonfire.com -
received SIGHUP --
master server rehashing configuration file
May 7 17:06:24 olympus proftpd[11933]: olympus.melonfire.com
(localhost.localdomain[127.0.0.1]) - FTP session opened.
```

and here's a snippet of the transfer log.

```
Wed May 7 11:20:00 2003 0 olympus.melonfire.com 9144 /home/joe/mbox
b
_ o r joe ftp 1 * c
Wed May 7 11:20:19 2003 0 olympus.melonfire.com 0
/home/joe/outfile b _ i r joe ftp 1 * c
Wed May 7 12:42:11 2003 2 olympus.melonfire.com 8820072
/home/ftp/pub/winapps/acroread/ar500enu.exe b
_ o a a@a.com ftp 1 * c
Wed May 7 14:11:32 2003 0 olympus.melonfire.com 9144
/home/ftp/incoming/mbox b _ i a a ftp 1 * c
```

You can perform so-called extended logging, in which all commands sent to the server are recorded to a file, with the ExtendedLog directive. This directive also allows you to specify the type of commands that are tracked - for example, use the keyword AUTH for authentication commands, RETR for file retrieval commands, or ALL for all commands (a complete list of supported keywords is available in the proFTPD documentation).

```
# set extended log
ExtendedLog /var/log/proftpdlog ALL
```

Here's a snippet of this log:

```
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:03:34 +0530]
"SYST"
215 -
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:03:39 +0530]
"PASV"
227 -
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:03:39 +0530]
"LIST"
226
123
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:03:42 +0530] "CWD
pub" 250 -
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:03:49 +0530]
"QUIT" 221 -
olympus.melonfire.com UNKNOWN nobody [07/May/2003:17:03:54 +0530]
"USER upload" 331 -
olympus.melonfire.com UNKNOWN upload [07/May/2003:17:03:56 +0530]
"PASS (hidden)" 230 -
olympus.melonfire.com UNKNOWN upload [07/May/2003:17:03:56 +0530]
"SYST" 215 -
olympus.melonfire.com UNKNOWN upload [07/May/2003:17:03:58 +0530]
"QUIT" 221 -
olympus.melonfire.com UNKNOWN nobody [07/May/2003:17:06:26 +0530]
"USER
ftp" 331 -
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:06:26
+0530]
"PASS a" 230 -
olympus.melonfire.com UNKNOWN ftp [07/May/2003:17:06:26
+0530] "SYST" 215 -
olympus.melonfire.com UNKNOWN ftp
[07/May/2003:17:06:31
+0530] "QUIT" 221 -
```

You can also set the server's debugging level with the DebugLevel directive, which must be followed by a number between 0 and 9 - this can come in handy if you experience problems with server startup or operation.

```
DebugLevel 9
```

Another way to obtain debugging output is to start the server with the "-d" parameter, followed by a number between 0 and 9. Consider the following example, which demonstrates:

```
$ /usr/local/ftpd/sbin/proftpd -nd3
- parsing '/usr/local/ftpd/etc/proftpd.conf' configuration
- <Directory *>: adding section for resolved path '*'
- <Directory incoming>: adding section for resolved path '/incoming'
olympus.melonfire.com - ProFTPD 1.2.8 (stable)
(built Wed May 7 10:22:32 IST 2003) standalone mode STARTUP
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
mod_cap/1.0: capabilities '= cap_chown,cap_net_bind_service+ep'.
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching POST_CMD command 'PASS (hidden)' to mod_log
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching POST_CMD command 'PASS (hidden)' to mod_ls
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching POST_CMD command 'PASS (hidden)' to mod_auth
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching LOG_CMD command 'PASS (hidden)' to mod_log
olympus.melonfire.com(olympus.melonfire.com[127.0.0.1]) -
dispatching PRE_CMD command 'SYST' to mod_core
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching PRE_CMD command 'SYST' to mod_core
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching CMD command 'SYST' to mod_core
olympus.melonfire.com (olympus.melonfire.com[127.0.0.1]) -
dispatching LOG_CMD command 'SYST' to mod_log .
```

Getting Virtual

proFTPD also comes with built-in support for so-called "virtual FTP", which comes in handy if you want to provide FTP service on multiple interfaces, IP addresses or ports on the same physical machine. proFTPD's <VirtualHost>...</VirtualHost> block allows you to specify different configuration options for each IP address, or to have the FTP server service connections on different ports on the same machine. Typically, you can use most of the same directives inside a <VirtualHost>...</VirtualHost> block that you would use outside it (although there are a few restrictions). This allows you to configure and customize each virtual host separately - as the following example illustrates:

```
<VirtualHost 192.168.0.77>
# set server parameters
ServerName "Olympus"
ServerAdmin "admin@some.other.localnet.com"
Port 22
# set default umask
Umask 022
# set the user and group for the server process
User nobody
Group nobody
# jail users to their home areas
DefaultRoot ~
# set root directory for anonymous users to /home/ftp
<Anonymous /home/hosts/olympus/ftp>
# set the user and group for the server process
User ftp
Group ftp
# alias "anonymous" login to "ftp"
UserAlias anonymous ftp
# restrict "anonymous" users from writing data
<Directory *>
<Limit WRITE>
DenyAll
</Limit>
</Directory>
</Anonymous>
</VirtualHost>
```

The code block above tells proFTPD to listen for connections on port 22 of the interface identified by IP address 192.168.0.77. Anonymous FTP is enabled for this virtual host, though users are not permitted to upload files, and regular system users are automatically jailed to their home areas.

Passing Messages

You can customize the messages displayed by proFTPD via its numerous display directives, some of which are outlined below:

```
# set authentication messages
AccessGrantMsg "You said the magic word!"
AccessDenyMsg "Leave and never darken my door again!"
# set connect message (before login)
DisplayConnect /usr/local/ftpd/connect.msg
# set per-directory message
DisplayFirstChdir .msg
# set login and logout messages
DisplayLogin login.msg
DisplayQuit logout.msg
```

Note that the files pointed to by the DisplayFirstChdir, DisplayLogin and DisplayQuit directives must all be located under the root directory space defined for the FTP user. Files located outside this space will not be processed. In the event that the server cannot find a specified file, it will display no message for the corresponding action.

You can also customize server timeouts and number of possible connections with a range of different directives. For example, the TimeoutIdle directive specifies the time the server will wait before disconnecting an idle connection, while the TimeoutLogin directive sets the time it will wait for connected clients to log in (all timeout values are specified in seconds).

```
# disconnect the client if idle for 2 minutes
TimeoutIdle 60
# disconnect the client if not logged in successfully for 2 minutes
TimeoutLogin 60
```

The TimeoutNoTransfer directive sets the time the server will wait for clients which have not issued any transfer commands before disconnecting them.

```
# disconnect the client if 2 minutes elapse with no transfer
TimeoutNoTransfer 120
```

The TimeoutSession directive sets a maximum time limit for each client session - the server will disconnect the client once this duration has elapsed.

```
# disconnect the client after 10 minutes
TimeoutSession 600
```

You can limit the maximum number of connections to the server with the MaxClients directive,

```
# set max number of clients at any time
MaxClients 50
```

or apply more fine-grained rules with the MaxClientsPerHost and MaxClientsPerUser directives, which allow you to restrict connections based on source host and source user respectively.

```
# set max connections per client and per user
MaxClientsPerHost 5
MaxClientsPerUser 5
```

You can limit the number of login attempts per connection with the MaxLoginAttempts directive,

```
# set max number of login attempts
MaxLoginAttempts 2
```

and place restrictions on the size of uploaded and downloaded files with the MaxRetrieveFileSize and MaxStoreFileSize directives.

```
# set max file size for downloads and uploads MaxRetrieveFileSize 10
Mb MaxStoreFileSize 4 Mb
```

Finally, you can permit resumption of broken file transfers with the AllowRetrieveRestart and AllowStoreRestart directives, which tell the server that it should allow clients to restart interrupted downloads and uploads.

```
AllowRetrieveRestart on
AllowStoreRestart on
```

Ending On A High Note

And that’s about it for the moment. In this article, I introduced you to the proFTPD server, explaining its important features and guiding you through the process of compiling and installing it on your Linux box. With the server installed, I then showed you how to configure it to support file transfers by both system users and anonymous users. In addition to a detailed explanation of the basic configuration file options, I also explained some of the security issues related to file transfer and filesystem access by remote users, and showed you the proFTPD configuration directives to minimize the security risks associated with opening up your system in this manner.

With your server now operational, I then moved on to a discussion of some of proFTPD’s other features, showing you how to configure the server to support FTP service on multiple network interfaces and ports, log all transfers and commands, control the display of messages, set limits for client connection and timeout values, and create rules to allow or deny access to the server.

However, everything I’ve discussed in this article is only the tip of the iceberg - proFTPD comes with many more configuration directives, which allow you extensive, fine-grained control over the way the server operates. If you’re planning on deploying proFTPD on your network, and you’re serious about doing a good job, you should also take a look at the following links:

The proFTPD Web site, at http://www.proftpd.org/

The proFTPD FAQ, at http://www.proftpd.org/docs/faq/linked/faq.html

The proFTPD manual, at http://proftpd.linux.co.uk/localsite/Userguide/linked/userguide.html

Example configuration files, at http://www.proftpd.org/docs/example-conf.html

The exhaustive list of proFTPD configuration directives, at http://www.proftpd.org/docs/directives/configuration_full.html

proFTPD mailing lists for support and troubleshooting, at http://www.proftpd.org/lists.html

Until next time...happy FTP-ing!

Note: Examples are illustrative only, and are not meant for a production environment. Melonfire provides no warranties or support for the source code described in this article. YMMV!