COMPUTER POWER USER

Tips & Tricks

Click To Print

Tips & Tricks
May 2008 • Vol.8 Issue 5
Page(s) 98-99 in print issue

Warm Up To Penguins

Fort Apache: The Server

Linux has two faces it shows to the world. The one we talk about most often here is the desktop and what you can do with it. But long before people thought about desktop use, Linux and its forebearer Unix were server OSes extraordinaire. And although the average home user is unlikely to need to set up a Sendmail daemon or use Named to serve DNS requests, there are a few services that may be worth installing or turning on.

A big one is Httpd, the Apache Web server. There are a number of good reasons to run a Web server on your Linux box. For one, if you host a Web site on a remote system, you can use a local Linux box running Apache to test it out. Once you have Httpd installed, you can also learn how to write Perl, PHP, Ruby, or Java Web services, among many others. And there are lots of useful packages, from blogging systems to calendar servers, which run on top of Apache. You can even set up a shared calendar for your family using Apache.

Depending on the options you selected when you first installed Linux, you may already have Apache installed. Still, it never hurts to check. The worst thing that will happen is that you'll be told Apache is up-to-date. As always, details differ for difference distributions, but the general ideas are the same. This time around, we'll be using Ubuntu 7.10.

If you open Synaptic Package Manager, you can easily find the apache2 package (it's right near the top of the list). Clicking this will install Apache. Synaptic will inform you that a number of other packages need to be installed, as well, which is fine. Apply the request and sit back.

Once you've installed apache2, you'll see that it's actually up and running. By default, Apache is added to your /etc/init.d directory, which means it's going to launch every time you boot the system. You can see this by opening up a browser on the Linux system and typing **http://localhost/** as the URL. Because we don't have any content yet, all you will see is a fairly boring directory listing.

Let's begin by creating a welcome page. The root directory for your Apache content is /var/www, and if you look there now, there's not much to see:

```
james@james-ubuntu-test:~$ ls -F /var/www

apache2-default/
```

If you're curious, apache2-default has a single file in it, which displays the string "It Works!" if you browse to it. We're going to create an index.html in /var/www itself. As shipped, if Apache looks in a directory and sees a file called index.html or index.htm, it will grab the contents of that file and return it if only the directory is specified in the URL. So when we navigate to http://localhost/, Apache will see that there's no file name specified in the URL. As such, it will look in the directory for an index file. If it finds it, it returns that. If not, it simply returns a directory listing. You can actually configure Apache to generate a 404 error instead. This isn't a bad idea: Letting people see your server's directory listings is a *bit* of a security hole.

Speaking of security, we want to fix the permissions on /var/www. With Ubuntu 7.10, the directory and all the files under it are owned by root:root (user root, group root). The Apache server itself runs as www-data:www-data, for security. The www-data user and group should have been added during the install. We want to do three things: add a group called “www” and add ourselves to that group; change /var/www’s ownership to www; and make any files created in the /var/www directory or its subdirectories group-writable.

Here’s the logic. We want to be able to edit files in the content hierarchy without having to be root, but we don’t want the Web server itself to be able to modify the Web content. If it could, someone could exploit the server and place infected files on our Web server. So by making the directories be owned by a group different from the one that the server runs as, but that we are a member of, we’re protected.

The actual set of commands is fairly simple:

```
$ sudo addgroup www

$ sudo adduser <yourname> www

$ sudo chgrp -R /var/www www

$ sudo chmod -R g+w /var/www

$ sudo chmod g+s /var/www
```

Welcome CPU Readers!

Once Apache is installed, creating static content is just a matter of creating an HTML file.

These commands do the following, in order: add a www group; add you to that group (replacing <yourname> with your username); recursively change all the files and directories under and including /var/www to be group-writable; and set the “sticky” group bit on /var/www, which says that any files or directories created underneath it should receive www as their group.

Now we’re ready to actually author some content. First, you have to log out and log back in again to access the www group added to your user settings. Fire up an editor of your choice (even a text editor will do) and save the following to a file called index.html, which you’ll save into /var/www:

```
<html><head>

<title>Welcome CPU Readers!</title>

</head>

<body>

<h1>Welcome CPU Readers!</h1>

</body>

</html>
```

Now if you point your browser to http://localhost/, you should get a friendly greeting. If all you want to do is create static Web content, you’re all set. Create files in /var/www or in subdirectories you create, and you can access them via URLs. And anyone on your local LAN should be able to access them, as well.

Just for fun, let’s finish off by writing a little perl program to generate some Web content. Create a new file in the /var/www directory called counter.cgi. It should have the following content:

```
#!/usr/bin/perl

print “Content-type: text/html\n\n”;

print “<html><head><title>I’m a program!</title></head><body>”;

for ($a = 0; $a < 10; $a++) {
```

```
print “This is line $a<br/>\n”;

}

print “</body></html>”;
```

Now make it runable by turning on the execute bit

```
$ chmod a+x counter.cgi
```

If you browse to http://localhost/counter.cgi, you’ll get a nasty surprise, however. Instead of the program running, you’ll get the contents of the file itself. This happens because CGI scripts have not been enabled. Letting random programs run can be a security risk, so you have to explicitly permit it.

In the current release of Apache, you can find the configuration files in /etc/apache2. In older versions or different releases, you may find them in /etc/httpd. The structure of the directories and files underneath varies from version to version, so we’ll describe it for Apache 2.2.4.

In 2.2.4 under Ubuntu, there’s a directory in /etc/apache2 called modules-enabled. This has the software and configuration files for the Apache modules you’ve enabled. You can enable new features by copying files from the modules-available directory to modules-enabled. To turn on CGI, start by editing the file called mime.types. Look for this line “#AddHandler cgi-script .cgi” and remove the “#**.**” Then save the file.

You also need to set a flag on the directory itself, you can find the configuration options for the “default” Web site in /etc/apache2/sites-enabled; it should be called something like “000-default.” Find the line that reads “Options Indexes FollowSymLinks MultiViews” and tack “ExecCGI” to the end (with a space between”MultiViews” and “ExecCGI.” Then, save the file and restart the server by typing **$ sudo /etc/init.d/apache2 restart** in the console. If you did everything right, you should be able to open http://localhost/counter.cgi and see Apache counter from 0 to 9.

by James Turner

Infinite Loop: Where There’s Smoke, There’s Horseradish

A fire alarm works great . . . if you can hear it. But what if you’re hearing-impaired? Bypassing what seems like the obvious logical choice of substituting bright flashing lights for a piercing screech, researchers at Japan’s Shiga University have developed a system that disperses the pungent aroma of horseradish to rouse the sleeping in the event of a fire. In a recent test, 13 of 14 subjects woke within two minutes, declaring their amazement at the device and a profound hunger for yellowfin tuna sashimi.

Source: www.wctv.tv/APNews/headlines/16188567.html