System Administration for the Web: Week 4 Lab

3 October 2005

1 Notes

Some of you have complained that it's difficult to read Adobe Acrobat documents on the Sun Rays. If you feel that the lag is unbearable, you can print the lab using the printers located down the hall; your inst account has a print quota of 100 pages. If you want to save trees, we've also posted an HTML version of this lab on the class website.

You should be quite familiar with and capable of using Unix-based systems at this point. In this lab, as well as the last, I did not provide the syntax for any commands. From now on, it is your responsibility to read the man pages or use a search engine to determine the syntax for any of the commands presented in this lab.

2 Submitting the Lab

Email your responses, the output of a command, or the commands you used to perform a certain exercise (as appropriate) to `dima@ocf.berkeley.edu` along with your inst account login, your OCF login, and your preferred contact email address.

3 Lab for Week 4

3.1 FTP

As explained in lecture, File-Transfer Protocol (FTP) is one of the most common ways of distributing files over the Internet. Many FTP servers allow anonymous access – that is, access to the general public. In this section, you're going to connect to an FTP server and retrieve a file.

As mentioned last week, the world's most popular DNS server is BIND, the Berkeley Internet Name Domain. As you can probably figure out from its name, BIND was developed at UC Berkeley. We're going to download version 9.3.1 of BIND.

[1] Use the *ftp* command to connect to `ftp.isc.org`, the FTP server that hosts the BIND program. For anonymous access, you must often provide "anonymous" as the login and a random email address as the password.

[2] Once you have successfully connected to the FTP server, you will be able to get help by inputting – you guessed it – *help*. Use *help* to determine what kind of commands you can use.

[3] List the files in the current directory on the FTP server.

[4] There shouldn't be anything interesting in the default directory. The BIND program files can be found in the `isc/bind9/9.3.1/` directory. Navigate to that path.

[5] Once you're there, download the `bind-9.3.1.tar.gz` file.

[6] Suppose you wanted to get `bind-9.3.1.tar.gz` and `bind-9.3.1.tar.gz.asc` from the FTP server. You could use the command you used in the previous exercise twice, but there is an easier way. Figure out how to get multiple files at once.

FTP works in both directions. You can download files from a server, and you can upload files to a server.

[7] Change to another directory. Try to upload a file into that directory. What happens?

In most FTP server configurations, anonymous users are not allowed to upload files. If they were allowed to upload files, anybody could post any obscene or illegal file they wanted and get the server administrator into trouble or consume all the server's disk space. In the final project of this course, you'll be setting up an FTP server which will allow authenticated users to download and upload files, so it's important that you understand how FTP works.

[8] Disconnect from `ftp.isc.org`.

3.2 Archive Manipulation

In Section 3.1 you downloaded a copy of BIND. The file you downloaded had the extension *tar.gz*, which means that it is a gzipped tar archive. In the UNIX world, *gzip* is the equivalent of zip compression, and *tar* is a method of packing files together.

[1] Use *gzip* to decompress the BIND file.

You should now have a file named `bind-9.3.1.tar`. A tar archive is a way of packing multiple files and directories together into a single file. Unfortunately, *tar* does not support compression, so people often compress tar archives with *gzip* or *bzip2*.

[2] Extract `bind-9.3.1.tar` using *tar*.

[3] Navigate to the `bind-9.3.1` directory that was formed when you extracted the tar archive.

[4] There should be a `make` directory inside the `bind-9.3.1` directory. Create a tar archive named `make.tar` of the make directory and move the archive to your home directory.

[5] Delete the `bind-9.3.1` directory and any other files related to BIND except for the archive that you just created. The most efficient way to do this is to use the *rm* command with the *recursive* and *force* parameters and a *wildcard*. Ask your instructor for assistance if any of these terms confuse you.

3.3 SSH

Last week we had you use SSH to log into your OCF account from Soda Hall. SSH was described as a nifty tool that allows you to log into remote servers over an encrypted connection, among other things. We'll be introducing those "other things" in this section.

SSH is a package of three tools: *ssh*, *scp*, and *sftp*. You used the first tool last week to log into your OCF account. *scp* is short for Secure CoPy, and *sftp* stands for Secure FTP.

[1] *scp* can be used to copy files between computers. Use *scp* to copy `make.tar` to your OCF account. Delete `make.tar` from your inst account.

[2] *sftp* operates very much like regular FTP. Use *sftp* to connect to your OCF account and download `make.tar`.

The aforementioned operations are the most common uses for SSH, and most system administrators use SSH on a daily basis. When you're working on the final project, you'll find the need to transfer files to and perform actions on your server, and SSH will make it possible. Consequently, it is important that you become familiar and comfortable with SSH.

SSH has many other features such as *tunneling*. If you're interested, you can search the web for information on how to use these features.

3.4 UNIX Filesystems

UNIX filesystems have many features that are not present in Windows filesystems. For the purposes of this class, the only feature that you need to know is the *link*. In a UNIX filesystem, files and directories do not have to be unique – they can be placeholders that point to other files or directories. There are two types of links: *hard* and *soft*. We'll only focus on soft-links. If you want to know the difference between the two types of links, please consult your friendly search engine.

[1] Locate the command to create links between files. Hint: Using *apropos* is a good way to start.

[2] Create a new text file and create a link to that file.

[3] Edit the file by running a text editor on its link. Open the actual file in a text editor. Compare what you see.

[4] Create a new directory and create a link to that directory. Move the text file you just created into that directory.

[5] Print the contents of the directory using the link to that directory. What do you see?