System Administration for Beginners

Week 3 Notes

September 29, 2008

1 Review

1.1 Logging In

The desktop environment you are using, if you logged in using default settings, is rather slow and not exactly the best environment to do work with. Before you log on to the machine, click on the "Session" button and select "Java Desktop Environment". This GUI is a bit easier to work around with and may look a bit familiar with other operating systems that you have seen.

Although there are many tools and options available through the GUI, try to exclusively limit yourself to the terminal when entering commands and opening up programs. This will serve as good practice on the command-line; most of the time a system administrator will be working with a command-line instead of a GUI.

1.2 Review of Basic Commands

By now you should be familiar with what some of the basic commands that deal with file and directory manipulation. Using the command-line, you should also be able to start programs (like a web browser) and terminate them by choosing the correct process from a list. Although you may not remember exactly what each command does, you know the appropriate commands and the resources to look up what something does.

1.3 Review of UNIX File Permissions

Last week's laboratory covered briefly on the concept of UNIX file permissions. It is *extremely* important that you understand file permissions because they are one of the pillars of UNIX security. If file permissions are incorrectly set, you will expose your data to malicious users which could result in the compromise of your system.

On UNIX-based systems, there are three types of permissions you can grant: execute, read, and write. These permissions are represented by the numerical values 1, 2, and 4, respectively. These enumerated values make it easy to grant a combination of permissions by simply adding up the values for the permissions that you want to grant and assign the resulting value as the permission.

For example, to grant write and read permissions, you would assign $2+4 = 6$ as the permission. The maximum value of a permission is 7 (execute, read, and write) and the minimum value is 0 (none at all).

Each file in UNIX has three categories of permissions associated with it: owner, group, and other. You can set permissions independently for each of these categories of users.

You set permissions by using `chmod` with a group of 3 numbers representing the permissions for the owner, group, and world, respectively. For example, to grant all permissions to the owner, read, and execute permissions to the group, and no permissions to everyone else, you would use the following command:

```
chmod 750 some_file
```

To help familiarize yourself with permissions, today's lab will have some exercises dealing with file permissions.

2 UNIX Tools for System Administrators

2.1 Filesystem Tools

2.1.1 Links

A useful filesystem feature that is present in practically all UNIX-based operating systems is the link. A link is merely a pointer to another file or directory that is virtually indistinguishable from the original file. In this way, a UNIX link can be considered a much more powerful version of a Microsoft Windows shortcut.

There are two types of links: hard and soft. The difference between the two is primarily technical (if interested, however, you may look up its `man` page) and is not very important at the moment. For now, always use symbolic links.

The command to create a link is `ln`. To create a symbolic link, you also need to specify the -s parameter:

```
ln -s original_file link_to_file
```

To unlink files, use `unlink` on the link.

2.1.2 tar

In the old days of UNIX, backups were performed using cassette tapes. While that technology has been replaced with more reliable means of performing backups, the commands that interfaced with the tapes are still present in UNIX-like operating systems. One such command is `tar`.

`tar` is similar to Windows-equivalent of WinZip or WinRAR. They package files together into an archive that can be easily transferred over the Internet, or, originally, onto tape media. Unlike WinZip and WinRAR, however, `tar` does not compress the data it archives – it merely combines them together into one package.

Many programs are available for UNIX-like systems are distributed in tar archives. Sometimes these archives are independently compressed using another program such as `gzip` or `bzip2`. You can use `tar` to package the contents of a directory into a single file that you can copy to removable media. To create a tar archive of a certain directory, use the following syntax:

```
tar -cf archive.tar directory_to_archive/
```

To extract a tar archive:

```
tar -xf archive.tar
```

To list the contents of an archive:

```
tar -tf archive.tar
```

2.1.3 Various Filesystem Tools

du determine the amount of disk space being used by a directory.

```
du -h some_directory/
```

gzip compress and decompress files using the gzip (.gz) algorithm.

```
gzip file_to_compress
gzip -d compressed_file.gz
```

bzip2 compress and decompress files using the bzip2 (.bz2) algorithm

```
bzip2 file_to_compress
bzip2 -d compressed_file.gz
```

2.2 Command-Line Tools

2.2.1 Pipes

One of the reasons for UNIX's success was its innovative features. The pipe is one of these innovative features that we can use to make life easier. In a nutshell, a pipe is a tool that takes the output of one command and feeds it as the input to another command. You create a pipe by using the `|` character.

For example, suppose you used the `du` command to list the disk space used by a large directory with many sub-directories. There may be so much output that your terminal scrolls for some time; using the scroll bars on the terminal may not be an option as the output is truncated at the top. Rather than allowing your terminal to scroll uncontrollably, you can pipe the output of `du` to the input of less. Recall that less is used for reading text files. To create the pipe that connects the output of one command to the input of another, you would type the following into the terminal:

```
du some_directory/ | less
```

2.2.2 Redirecting Output

Using the pipe was a way of redirecting output; instead of displaying the output on the screen, we took it and sent it directly into the input of another command. In some cases, we want to save the output in a file. For example, you may want to save the output of a command as a log for future analysis. To redirect the output of a command from the terminal to a file, you use > or >>. You use one greater-than symbol (>) when you want to redirect the output of a command to a file and overwrite the file's contents. You use two greater-than symbols (>>) when you want to redirect the output of a command to a file by appending it to the end. For example, to redirect the output of du to a file named du_output, you would use the following command:

```
du some_directory/ > du_output
```

2.2.3 screen

System administrators often need to use multiple terminals at the same time. Rather than opening many terminal windows, they use screen. screen allows system administrators to create multiple virtual terminals that they can switch between using key combinations.

Besides having the ability to create multiple terminals in one, screen also has the ability to detach from a terminal and reconnect to it later. For example, if you ran a command that would take a very long time to complete, you would not be able to logout of the system because the system would kill the process as you logged out. You can get around this by using screen: you start screen, execute your command, and detach. Once detached, you can logout of the system and your process will continue running. Whenever you want to check on the progress of your process, you can re-attach to your screen.

It is not necessary for you to use screen. It is, however, a very useful tool that many system administrators use on a daily basis, so we recommend that you try it out. Since screen maintains a persistent state even after you have logged off, you can keep your work "saved" for later when you disconnect. The following is a quick list of screen commands; please read the man page if you want to learn the rest:

- ctrl-a c – create a new screen
- ctrl-a k – kill current screen
- ctrl-a n – move to next screen
- ctrl-a 0 – show a listing of all screens

2.2.4 Various Command-Line Tools

history display the commands you've entered
clear clears your display

2.3 Tools You Definitely Need to Know

2.3.1 SSH

The Internet is a very dangerous place. On the outside, most people are oblivious about the dangers when connecting wirelessly. Hackers can easily monitor your web sessions; especially over a wireless Internet connection. Consequently, system administrators try to use encrypted network tools whenever possible.

A Secure Shell client, or SSH, is a network tool for remotely logging into a remote system. It is the tool that you would use to connect to the computers here at Soda Hall using your home computer. If you have an OCF account, you can use SSH to connect to the systems at the OCF using the following command:

```
ssh your_login@ocf.berkeley.edu
```

If you do not yet have your OCF account, you can use SSH to login to your inst account using the following command:

```
ssh cs198-XX@solar.cs.berkeley.edu
```

You might get a warning about encryption keys. If prompted, input 'yes' and continue. If your login is successful, you will have opened a shell into your OCF account. Any commands you execute in the shell will execute directly on the OCF's computers and it's output, depending on how it is redirected, will display on your terminal. When finished, you may disconnect at any time using the command `exit` to exit out of the shell.

SSH also includes some tools for transferring files between computers over an encrypted connection: `sftp` and `scp`. `sftp` is very similar to File Transfer Protocol (FTP), which some of you may have used to upload webpages or files, and uses the same syntax as `ssh`. `scp` is like a network version of `cp`. To copy a file from your inst account to your OCF account, use the following command:

```
scp file_to_copy login@ocf.berkeley.edu:some_path
```

The text after the colon in the last argument to `scp` specifies either an absolute or relative path to the directory in which you want to store the file. If unspecified, it automatically defaults to your home directory, but be sure to always include the colon. Otherwise, it will simply act like `cp` by creating a file called "`login@ocf.berkeley.edu`".

2.3.2 Various Useful Commands You Need to Know

grep search text for a specific string

```
cat some_file | grep look_for_this
```

find search for files

```
find .  -name document.txt -print
```

tail print out the last 10 lines of a certain file

```
tail very_long_log
tail -n 20 some_log
tail -f another_log
```

uptime shows how long the system has been on and the current load

who show who is logged into the system

w shortcut for `who` and `uptime`

last show the last few logins on the system

2.4 Other Useful Tools

2.4.1 rsync

`rsync` is a tool for mirroring files and directories across two systems. It is primarily used as a backup tool, but it can be used for all sorts of operations where it is necessary to keep two sets of data in sync.

A remarkable feature of `rsync` is that it will intelligently detect which files have been modified and only transmit the modifications to the receiving server, resulting in a significant reduction in bandwidth transfer required for an update. `rsync` also support encryption as well as on-the-fly encryption.

Though the details will not be discussed here, an important piece of advice is to pass the `-n` parameter when running `rsync` for the first time on a directory; this will run through the process but will not actually do anything. By performing a dry-run, you can review the operations, analyze the output, and make corrections to the parameters if necessary.

2.4.2 lynx

`lynx` is a command-line browser. It is rather useful when you need to look up information quickly and you know where to go. Modern command-line browsers have image capabilities, but most will display only text.

3 Connecting from Home

To connect to the computers at Soda Hall or the OCF using your home computer, you will need to obtain an SSH client. If you are using Microsoft Windows, you can download the SSH Secure Shell program from the website `http://software.berkeley.edu`. An excellent alternative for Windows users is the PuTTY SSH client. Mac OS X users should have an SSH client already built in; look for the Terminal under "System" .

Regardless of which SSH client you use, you will need the following information: a hostname and a login name. To connect to the computers at Soda Hall, you can use `solar.cs.berkeley.edu`. Your login name is the inst account you received (`cs198-XX`). To connect to the OCF, the hostname and login name are generally different: you'll want to use `ocf.berkeley.edu` as the host name and the login name you choose when signing up for your account.

Once you've connected, your client will ask for your account password. Type it in. If your login is successful, you should have a shell open to either your inst or OCF account. Any commands you type into this shell should work just as if you were physically in Soda Hall and executing them on the computers here.

There is on drawback, however. Unless you have setup your own X server at home, any programs that require a GUI will not work. For example, if you attempt to start Firefox using your shell, you will probably get an error.

The choice of server you want to connect to is ultimately up to you. Both of your accounts are restricted in a sense that you do not have full control of the system; you might find, however, that the OCF provides more access to different programs and utilities that the instructional labs do not. We will soon deal with account types and permissions in the upcoming weeks.