System Administration Decal

Intermediate Lab #1

February 1, 2010

Introduction

This week, we learned about the shell and various typical UNIX commands we can use within the shell. Additionally, we explored some special features of the shell, including output and input redirection, pipes, and substitution. In this lab, you will practice these skills while learning even more about the shell, in a series of themed exercises. Please email your responses to mgasidlo+decal@ocf.berkeley.edu and remember that you can always email me with questions about the lab. Remember to use the slides as a resource for solving the exercises also.

IMPORTANT: If you are using your cs198 account to do these exercises, I highly recommend that you use the 'bash' shell. The default shell used for Berkeley CS accounts is not adequate for many of the idioms we have learned. You can start the bash shell after logging in with 'exec bash'. If you have a Mac, it uses bash by default. I won't stop you from using csh or tcsh if you really *really* want to, but consider yourself warned. Or, if you're feeling adventurous, try the 'zsh' shell.

Some notes which may help you:

- This lab will ask you to make extensive use of *man*. By default, *man* uses *more* to display *man* pages. For a more pleasant experience using *man*, you can issue the command "export PAGER=less" from the command line. To make this persistent, issue the command "echo 'export PAGER=less' >> ~/.bash_profile" from the command line.
- To switch your login (i.e. default) shell to bash, run the command "ssh update" and follow the instructions to change your shell.
- You can (and probably should) change the password on your cs198 account. To do this, run the command "ssh update" at the command line and follow the instructions to change your password.
- Some of you may notice as you Google around that there exist repositories of *man* pages on the internet. While it may be tempting to rely exclusively on these, there will be times when you (*gasp*) have no internet access. In these circumstances, you will need to know how to use *man* from the command line. In addition, many programs have different versions on different operating systems which have different command line options and may behave very differently. Make sure you're reading the *man* pages for the version of the program you're actually using. (The pages accessed by running *man* will always—assuming everything's installed and configured correctly—display the correct information for the version installed on that machine)
- Those of you using Macs may notice that some programs (especially vi and less) don't quite work properly when you ssh into another system. To solve this, run "export TERM=xterm" at the command line.

I. Review of Basic Commands and Shell Idioms

In a sentence or two, describe what each of the following command lines does precisely. You may have to use "man" pages to look up what certain commands and options do, as described in lecture. (You remember command-line options, right?) For some commands that are **built in** to the shell, there won't be any man pages, so you can also use your CS lab account to try these commands out. Also, try removing the options from the command lines to see what changes. Failing that, maybe Google will help you.

1. ls –l a???bc
2. rmdir * (**Hint**: rmdir's exact behavior is not immediately obvious. Be careful.)
3. mkdir –p a/b/c
4. ls | grep 'pookie' | wc –l
5. mkdir $(<dirs)
6. find >file 2>/dev/null

II. The wget command

The *wget* command is used to retrieve web pages and other files hosted on web servers on a command-line interface. Familiarize yourself with this tool by issuing *man wget* from a terminal to read its manual page. Then, answer these questions (the man page is useful for many of these):

1. If you run *wget* on a URL, you will notice that it prints a lot of output, including a text-based progress bar. Come up with **at least two ways** to inhibit its output, either using the capabilities of the shell or by passing options to *wget*.
2. Figure out how to make *wget* print the contents of a URL to standard output. Use this option to create a **pipe sequence** that searches www.cnn.com for the term 'sports'.
3. What is special about the way *wget* prints its informational status messages/progress bar? **Hint**: try using output redirection to hide one or more output channels.

III. Exit codes

In UNIX environments, all programs report whether they have failed or succeeded when they terminate using an **exit code**. You might think of it this way: running a program is like calling a function in a piece of code, and checking to see what its return value is.

1. Find a way to display the exit code of the program that was last run in the shell using Google (or your favorite search engine.) Using the Web to find answers is instrumental even for seasoned sysadmins; it's impossible to know *everything* off the top of your head. In your answer, include the query you used and the web page from which you got the answer.
2. Try using *wget* to download a URL that does not exist. Using your answer to #1, determine what exit codes are commonly used for success and failure.

IV. Job Control

The shell has the ability to manage many running programs at once, and stop and continue them at your whim. In this section of the lab, you will learn how to control this power. To begin, use your knowledge from exercise II to start downloading a large file from

somewhere. (A few megabytes is enough – you just need to buy yourself time.)

1. While it is downloading, hit Control and Z at the same time. (From now on we'll abbreviate this to ^Z; similarly Control + C will be ^C.) Clearly describe what happens.
2. Hitting ^Z will return you to a prompt. Describe the output of the *jobs* command, and then try using the *fg* or *bg* commands. Figure out what all three of these do and how they pertain to job control. You shouldn't need a search engine to do this. Just experiment!
3. Use this procedure to run at least three programs at once. Supply the output of the *jobs* command when this is happening.
4. Now start your download again, but this time interrupt it using ^C. Compare the behavior of ^C and ^Z.
5. Every time we have run programs up until now, they have started in the foreground (i.e. you are not returned to a prompt immediately.) Find a way to start a program in the background.