CIT 371 Lab 3: The Bash shell continued

This lab can be done with the Web Console or with SSH/PuTTY. See the Student VM access document for information on accessing your VMs.

In this lab, you will deepen your knowledge of Bash through the use of wildcards, command line editing and redirection. Log in to Coivcenter and start your VM1 VM (CentOS6) if needed. Log in as Student. Open a terminal window.

1. A variable is a name tagged to a piece of memory where you want to store some useful information. An *environment* variable is a variable that has been defined by the operating system or a program. The user may also define variables.
   1. To output all environment variables, enter the **env** command. *What are the first four environment variables shown?*  (NOTE: you can use **env | less** or **env | more** to pause the listing, or scroll up in the window) *What are the values of each of these variables? PWD, HOME, USER, HOSTNAME.* 
      1. **ORBIT\_SOCKETDIR,HOSTNAME, IMSETTINGS\_INTERGRATE\_DESKTOP, TERM. The values of each variable is PWD= /home/Student HOME= /home/Student USER= Student HOSTNAME= CentOS6Template.**
   2. Another way to view the value of a variable is to use the echo (output) instruction. Type **echo $HOME**. The $ means “get me the value of what is stored here”. Compare this to the output from **echo HOME**. Without the $, echo just outputs whatever is on the line, like the word HOME. One environment variable not displayed using env is PS1, which defines your prompt. Type **echo $PS1**. *What is the result?* It looks cryptic but things like \u stand for other variables like USER. \$ means “output a $”.
      1. **[\u@\ h \W] \$**
   3. To create a variable, type *name=value* where *name* is the variable name and *value* is the value you want to store there. The value should either be a string, in which case it can contain any characters desired, or an integer (whole) number. If the value on the right hand size contains a space, you must enclose it in either ‘’ or “”. We will use “”for now. Establish two variables, FIRST and LAST to store your first and last names. *What commands did you enter?*
      1. **FIRST=KALEB**
      2. **LAST=ALSTOTT**
   4. Using a single echo statement, output the values of your first and last names. *What command did you use? What was output?* Make sure your output has a blank space between your first and last names. *How did you accomplish that?*
      1. **echo $FIRST $LAST, the output was KALEB ALSTOTT. I accomplished the blank space by outputting the first and last name separate in the same command by using two $.**
   5. Write a single echo command to output your first and last name, your user name, and your home directory. *What command did you enter?*
      1. **Echo $FIRST $LAST $USER $HOME**
2. ~ and tab completion
   1. The ~ symbol represents “home directory”. Type **cd ~**. *Where are you?* Type **cd ~zappaf**. *What does your prompt now say?* Type **cd ~foxr/HUMOR**. *What directory are you in?*
      1. **You are in the home directory. My prompt now says [ Student@centOS6Template zappaf] $ which means we are in the zappaf directory. We are in the HUMOR directory.**
   2. Tab completion allows you to type in only the first part of a filename and let the Bash interpreter finish it for you. We will try this out using the command cat, which displays the contents of a text file. If you are not currently in ~foxr/HUMOR, switch to that directory. Type **cat bumperstickers**. Now try **cat b<tab>** meaning type cat b and then hit the tab key. *What happened?* Now try cat s<tab>. The response is a beep (you may or may not hear it depending on whether the speakers are on). This is the interpreter telling you that it could not uniquely identify a file that starts with ‘s’. Instead, type **cat sm<tab>**. This will work because there is only one file that starts with “sm”. Now try **cat s<tab><tab>** (press the tab key twice). *What happened?*
      1. **When typing cat s tab tab, we see the two files that start with s which are smileys and stupid-lawyers. Since there is two s files and not enough information we need to expand more onto to tab complete.**
   3. Change directory to /usr/bin (**cd /usr/bin**). Type **cat c<tab><tab>**. *What happens? How did this differ from doing* ***cat s<tab><tab>****?*
      1. **When typing cat c tab tab we see all the directories listed that start with a c which is a huge list and goes all the way down to more of a listing. When typing s tab tab it pops up with the option of “Display all 173 possibilities?” (y or n). The difference is the option it gives you of the yes or no.**
3. Wildcards are metacharacters (reserved characters or symbols) that can be used to match a number of items in file-based commands such as ls. The most common wildcard is \*, which means “match anything”.
   1. Change directory to ~/DUMMY-DIRECTORY. Type **ls \*.txt**. *What is displayed?* Type **ls** and *compare the results* to ls \*.txt.
      1. **file1.txt, file2.txt, file3.txt, file4.txt. When typed ls, we see all the directories not just the text files.**
   2. The ? matches any 1 character. Change directory to directory2. Type **ls** to see the files, there should be two of them. Type **ls aa?.\***. *Why did aa1.abC match but not aa10.bBC?*
      1. **Because we specified to list all files that have aa and then only ONE character followed by the period . and then match anything. So this matched with aa1.abc and not aa10.bBC because the ? only specifies only one character.**
   3. Change directory to ~/DUMMY-DIRECTORY/directory3/c1. Type **ls foo?**. (the ? is part of the command but not the period) *Why does foo not appear but foo2 and foo3 do?*
      1. **Foo does not appear because we are specifying for a file with foo followed by one character. Foo does not fit in with this specification.**
   4. The use of [ ] allows you to list a series of characters to match. For instance, [ab] means match anything that is an a or a b. Return to DUMMY-DIRECTORY and type **ls \*.[bc]\***. *Which files are listed? Why did aaa.abc not appear?* Explain what \*.[bc]\* means.
      1. **Aa6.bbc, aa7.bcc, aa8.bCC. The reason why aaa.abc did not appear is because we are looking for a file or directory being named, any characters .(period), and then anything that has b and or c followed by any character. What this ls command means is any character dot anything that has b or c and then followed by any character.**
   5. cd to ~zappaf. Type **ls –l** and you will find many directories. Type **cd ZAPPA**. Type **ls \*.\*** and then **ls \*** (with no period). *What are the differences?*
      1. **The difference between the ls \*.\* and ls \* is that the first listing will provide you with a listing of any charcter period followed by any charters while the second search will provide you with all files and texts in the ZAPPA file because of the ls (list) any character command.**

The characters \*, ? and [ ] are all used with wildcards but are also used in regular expressions, which we will explore in a few weeks. It is important to remember that \* and ? differ as wildcards.

1. Type **cd ~**. Many of the Linux commands allow for multiple arguments (parameters). For instance, type **ls DUMMY-DIRECTORY FILES** and the ls command will display both (remember to use tab completion, it will shorten the typing!) Another Bash shortcut is to place directories/subdirectories in { } to enumerate several different items to apply to a command. Type **cd DUMMY-DIRECTORY**. This directory contains subdirectories (directory2, directory3, directory4, dir5) some of which contain subdirectories. Let’s explore a command to list them all.
   1. Type  
      **ls {\*,directory2,directory3,directory3/{c1,c2},directory4,dir5}**. *What does this do? How many directories were output?* Notice how each directory name precedes the list of files in that directory. *What does the \* indicate in { }*?
   2. **This outputs the files in the directories listed in those specific command. There were 10 directories listed. The \* indicates any character defined in {}.**
   3. Type **ls –R** (this is a recursive directory listing meaning that it lists all of the contents of the current directory and then recursively all of the contents of its subdirectories (so that any subsubdirectories). Notice this accomplishes the same thing, so why use the { }? Because we might want to list a subset of the items. So we can precisely control the directories output by placing only those we are interested in in the { }. *Provide an ls statement using { } that outputs the contents of dir5, and c1 and c2 of directory3.* 
      1. **ls {\*,dir5, directory3/{c1,c2}}.**

Shut down your VM if desired. Disconnect from the VPN if you are using it and submit your lab report.