CIT 371 Lab 4: More Bash features

This lab can be done using the Web Console or SSH/PuTTY. See the VM Access document for more information on accessing your VMs.

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

1. Command Line Editing
   1. You want to rename a series of files using the mv (move or rename) command. In ~/DUMMY-DIRECTORY, there are 4 files named file1.txt, file2.txt, file3.txt and file4.txt. We want to rename them to end with .dat instead of .txt. We can’t issue a single mv statement to do a mass rename instruction. Instead, we rename each one at a time. The first command is **mv file1.txt file1.dat<enter>**. The novice Linux user would then retype this instruction, changing the number from 1 to 2, 2 to 3 and 3 to 4. With command line editing though, we can edit the instruction three times to save us time and effort. How? Type the above mv command exactly as shown. Next, type **control+p** (press the control key and while holding it down, press p). This recalls the last instruction (you can also do this using the up arrow). The last instruction, mv file1.txt file1.dat, is on the command line. We need to replace both 1’s with 2’s. Type **control+b** to back up 1 space. Repeat this until you reach the “.” before the .dat. Hit the **backspace** key to erase the “1”. Type **2**. Now type **escape+b** (hit the escape key and then type b) two times. This should leave you at the t in .txt. Type **control+b** to move back to the period and then backspace to erase the “1”. Type **2**. Press the <enter> key to enter the instruction (you don’t have to be at the end of the instruction to press <enter>). *Was this easier than retyping the entire instruction?*
      1. **Yes it was a lot easier being able to move with the keyboard and not retype.**
   2. Redo this again, changing file3.txt to file3.dat. Instead of using control+b repeatedly, use escape+b to move backward and escape+f to move forward to reach each period (and then backspace followed by 3). *What sequence of escape+b and escape+f did you use?*
      1. **I used 4 of escape+b and then 9 of the escape+b to change the files.**
   3. Redo this again, changing file4.txt to file4.dat. This time, after typing control+p, type control+a. *Where does this move you?* Use any combination of control+b, control+f, escape+b and escape+f to change this. *From steps 3a-c, did any of these make it easier than retyping the entire instruction? If so, which one(s) did you prefer?*
      1. **It moved back to the following command and then moves me to the front of the mv with the m. I felt like a was the easiest one to retype and was the simplest.**
   4. Create an empty file with **touch a\_file.txt**. We want to create a new file called b\_file.dat. *Explain how you would do this with command line editing.* Note: control+e moves the cursor to the end of the line.
      1. **Mv a\_file.txt b\_file.dat**
   5. Recall a previous instruction that has several different items on it separated by spaces. Experiment with these control characters and *explain what each does: control+k, control+y, escape+f, control+d, escape+d.* See table 2.6 on page 53 for an explanation of all available command line editing keystrokes.
      1. **Control+k = delete all characters from curser at end of the line**
      2. **Control+y = Yank. Return all deleted characters to curser position.**
      3. **Escape+f = move forward in the command line**
      4. **Control+d = delete character at curser**
      5. **Escpae+d = the date**
2. A pipe redirects the output of one command to be the input to another. The pipe symbol is |.
   1. cd to /etc. Type **ls**. A lot of files scroll past. To simplify this, we will use a program that pauses the information being placed on the window one screen at a time. There are two, less and more. Type **ls | more**. *What happens?* To move forward in the file, use the space bar, the enter key and the letter s and q. *What do each of the space bar, enter, s and q do?* Type **ls | less**. The less command is controlled by the same keys plus the arrow keys, page up and page down. Experiment with less and summarize how it differs from more. Type **ls | sort –r | less**. *What does this command do?*
      1. **Ls more provides a list of all text files and is continued with the more at the bottom of the page. The space bar jumps a whole page to the next list, enter is to jump one line, s skips forward one line, and q quits and exits back to the command. The biggest difference between using the less and more command is using the arrows and how much data you want to see. It is easier to look through lesser amounts of data if you know what you are looking for. The ls | sort –r | less command allows you to list in reverse order and have the less option to navigate.**
   2. cd to ~zappaf/ZAPPA. Type **cat c\* | grep “f$” | sort**. *Explain what you see for output* (do not copy the output, just describe it). *What does this instruction do?* 
      1. **The output looks like a poem or story with text. This instruction allows us to open up a file of text that starts with c followed by any character and then we use grep to search through text to find an f followed at the end of the line, and then sorted. Demonstrating on how we sorted these lines ending with myself, cuff, of , and of.**
   3. wc is a word counter, outputing the number of lines, words and characters of a file. cd to /etc. Use **ls –l** and count the number of files by piping it to **wc –l** (**-l** counts the number of lines). *What command did you enter? How many did you find?* 
      1. **ls -l | wc -l . And then there was 261 lines**
3. There are other forms of redirection which we explore here
   1. If you are not in the /etc directory, change to it. Type **ls > ~/etc-files.txt**. This command will perform an ls of the /etc directory and send the output to the file etc-files.txt in Student’s home directory (rather than displaying the result to the terminal window). *Why is the ~/ necessary in this instruction?* HINT: try it without the ~ and see what happens. Change directory to /bin. Type **ls >> ~/etc-files.txt**. The >> is an append. Type **less ~/etc-files.txt**. *How has the file changed?*
      1. **It is important to specify what pathway you are taking to that directory or else it will not work. The file has been added to the bottom and appended in the bin directory.**
   2. < is used to redirect input to originate from file instead of keyboard. As most Linux instructions obtain their input from files, < is not used very often (we will use it when we start writing our own scripts). Try it out though with **less < ~/etc-files.txt**. Now type **less ~/etc-files.txt**. *Is there any difference?*
      1. **No there are not any differences.**
   3. Type **cd ~** (Student’s home directory). << takes input from the keyboard. There are two ways to use it. ***command* << <enter>** causes command to execute, accepting input until you type **control+d**. ***command* << *word*** causes command to execute, accepting input until you type *word* <enter>. Type **wc << quit**. Your prompt is “>”,the interpreter waits for you to enter text. Type whatever you like (including 2 or more <enter>s) for a few lines, then type **quit <enter>**. The text you entered (excluding “quit”) is sent to wc. *What output did you get?* Use **man wc** to read what the output means.
      1. **4, 15, 73**
   4. We can combine << and > to create a text file. The cat command usually outputs the contents of a file to the terminal window. But the command **cat << quit > *filename*** inputs text from the keyboard until the user enters quit <enter>, and then sends the accumulated text to the file *filename*. Type **cat << quit > list.txt**. You are given the “>” prompt. Enter a list of grocery items on several lines (e.g., bananas, bread, milk, eggs, cheese). When done, type **quit**. *What output are you given?* Now type **less list.txt**. You should see your list. This is a simple way to create a textfile.
      1. **The output given was prompted to the next command, when less list.txt was inputted we are then presented with our grocery list.**
   5. Type the command **cat << quit | sort > list.txt**. Re-enter your list from part e, unsorted. Type **less list.txt**. *How does this version of the file differ? Explain the command you entered (what does each command and redirection symbol do).*
      1. **This file version is in order compared to the first list we looked at. The command cat << quit | sort > list.txt allows us to display the contents of the file list.txt . The << arrows > describe what’s going on. The << quit will allow the user to enter text to be inserted into the file while the | will sort the text. Finally, the > will input this text into the list.txt file.**
4. You tailor your Bash environment with variables and aliases. From the command line, changes are not permanent. There are scripts we can place our changes into for permanence.
   1. From Student’s home directory, type **cat .bashrc**. This script has an if statement to test if /etc/bashrc exists and if so, execute it. /etc/bashrc has a few instructions: create PS1 and PATH, set the umask value (this will be explained later in the semester) and execute any existing Bash shell scripts in /etc/profile.d. These are files that have the extension .sh. Type **cat .bash\_profile**. *Ignoring the if statement, what else does this script do?*
      1. **This is the user specific environment and startup programs.**
   2. Type **echo “alias..=‘cd ..’” >> .bashrc** *What does this command do?* Look at your .bashrc file. With this entry now in your .bashrc file, to have it take effect, type **source .bashrc**. The alias is now defined in your environment, test it by typing **..** *What happens?* Type **alias** and it should appear as the first alias in your list.
      1. **This command allows user specific aliases and functions in the .bashrc. When doing the command .. it moved me to the home directory.**
   3. *In which order do the two scripts .bash\_profile and .bashrc execute? How do you know?*
      1. **The order of the two scripts goes bash\_profile and then the .bashrc. We know this because when we man and see the if options it shows the pathways it can go down and in order to get to the .bashrc you have to open up the .bash\_profile.**

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