Laboratory #4: Filesystem shell commands

Unix (420-321-VA) - Winter 2021
Teacher: Tassia Camoes Araujo

Goals:

1. Explore environment variables
2. Practice file system shell commands
3. Practice commands connection, redirection, expansion

Instructions

Part I: Shell and Environment variables

Shell variables are a set of name/value pairs that are available in your shell. You can set a variable by typing in a terminal <*variable_name*>=<*value*>. You can use the command *echo $var_name* to check a variable value (attention to the $ before the variable name). If a shell variables is *exported*, it means the value is also accessible to child processes in the system (another shell). The exported variables are called *environment variables*.

A few variables for you to check: USER, DESKTOP_SESSION, HOME, PWD, OLDPWD, PATH

Related commands:

export <*variable_name*>=<*value*>

echo $*variable_name*

printenv *variable_name*

For the hands-on practice, use ";" to execute multiple commands in single command line. You will need the record of commands and output for the lab report.

1. Print the value of the variable PS1. Read the section PROMPTING of the bash manual and try to understand what the escape characters mean.
2. Change your prompt to use a few of the available options for the PS1 variable. Research online for hints on how to setup a custom prompt for your taste.
3. Take a screenshot of the resulting prompt.
4. Move to the directory /tmp, then move back to your home directory.
5. Print the values of the variables PWD and OLDPWD.
6. Execute the commands *pwd* and "*cd -*", explain their output.
7. Now change the values of the variables PWD and OLDPWD to any existing path.
8. Execute again the commands *pwd* and "*cd -*", and describe how they are affected by those environment variables.

Part II: File system practice

1. Create the directory /*tmp/420-321-F20/lab4* and enter this directory.
2. Create a file named ‘sample.txt’ using *touch*. Run the command *stat* on the file.
3. Use *touch* again on the same file with no arguments, and run stat to observe what changes.
4. Touch the same file again, now with the option -a, then run stat. Explain what changes.
5. Touch again, now with the option -m, then run stat. Explain what changes.
6. List the file attributes (long list option of *ls*).
7. Use a wildcard to list all files that end with ‘.txt’ files in the current directory.

Part III: Connecting shell commands

In a shell, you can use special characters to connect commands (“|”), send the output of a command to a file (“>”), perform multiple tasks in sequence (“;”), and expand the result of commands or arithmetic expressions.

1. Print the string *“date”* and send the output to the file *lab4.sh.*
2. Print the string *“echo Running my lab#4 script”* and append to *lab4.sh.*
3. Execute the file with *“bash lab4.sh”*.
4. Create a hardlink named *“hard.sh”* and a symbolic link named *“soft.sh”*, both pointing to the same file *lab4.sh*.
5. Use a shell command to show that the files lab4.sh and their links have the same content.
6. Execute all your script files and tell if there is any difference in their output.
7. List the content of the directory with *“ls -l”*. Note the number that appear just after the permissions of each file. That is the number of hard links pointing to a physical location (inode). Note also the file types, the first character that appears, just before the permission of each file.
8. Remove the file *lab4.sh* and list the directory content with *“ls -l”* again.
9. Describe the changes you see from the previous run of *ls*.
10. Restore the file *lab4.sh* by creating a hard link from *“hard”* to *lab4.sh*
11. Add the line “*sleep 3”* at the end of *lab4.sh.*
12. Run all your files again, and explain if there is any difference in the output.
13. Add a line to the script to use another command or environment variable that will identify the user running the script.
14. Run the script and record the output.
15. Use brace expansion to create files *file1.txt*, *file2.txt* and *fie3.txt* in a single command.

16. Combine the commands *cat* and *grep* to catch all occurrences of the string "d*ate"* in the current directory.
17. Combine the previous command with *wc* to count how many occurrences there are.
18. Combine the commands *ls* and wc to count how many files that end with ".txt"
19. Use the command *echo* with *command expansion* to print how many files there are in the current directory.
20. Adapt the previous command, to count how many scripts you have in this directory.

Part IV: Recording commands

This time you'll collect the list of commands yourself. Each new command practiced this week should be included in your ever-growing commands table. You'll be asked to include the updated table with all commands description in the next lab report.

Part V: Deliverables

1. Open LibreOffice to create your lab document.
2. Include a header with course name, section, your student name, the license of your work (suggestion: one of the creative common licenses).
3. For each part, show your sequence of commands and output.
4. Export your file as PDF and upload it to Omnivox.

Thanks!