**CSC 209 UNIX Tools**

|  |  |  |
| --- | --- | --- |
|  | **Assignment 4** |  |

**Objectives: Access Permissions, Redirection, and Pipes**

**Note**: **Read** the entire assignment carefully and carry out the following tasks one by one. For some steps, I may provide the required UNIX command. For others, identify the UNIX commands you need. **UNIX is case sensitive**. Create folders and files exactly as indicated below, paying special attention to case. Complete the report as indicated below and submit the same. I do not want to see copy-and-paste of your computer output in your report. By requiring you to write I am hoping that you will remember the UNIX commands used and the results observed.

1. Login to holly server (or one of its other names) using your Webmail Net-ID and password.

Since we are not using the graphical user interface (GUI), you have to rely on your knowledge of UNIX commands. Take a note of how the command prompt placed by the shell appears and report below:

Shell prompt: [50] [etay1@courses2016:~]$

Type an appropriate command to print the working directory.

Command to print the working directory: pwd

Your login directory: /home/etay1

Use an appropriate command to switch to the **csc209** folder you created in Assignment # 1.

Command to switch from login directory to **csc209**: cd csc209

1. Complete the following steps:

Determine the current working directory.

Command to print the working directory: pwd

Your current working directory: /home/etay1/csc209

You should currently be located in **csc209**. Create a new folder named **asgn04** under **csc209**. Set the access permissions for **asgn04** as **rwx --- ---** by typing a command:

**chmod 700 asgn04**

Command to create the **asgn04** folder: mkdir asgn04

Command to set the access permissions: chmod 700 asgn04

Move to **asgn04** and determine the current working directory.

Command needed to switch from **csc209** directory to **asgn04**: cd asgn04

Command to print the working directory: pwd

Your current working directory: /home/etay1/csc209/asgn04

Shell prompt: [56] [etay1@courses2016:~/csc209/asgn04]$

**From now on, unless explicitly permitted, you should not move away from this asgn04 directory.**

**If you happen to log out in the middle of the exercise, be sure to use appropriate command(s)**

**and move to asgn04 before you continue.**

1. Display the contents of the file **.bashrc** under your login directory and determine the user mask (**umask**) set there.

Command to display **.bashrc**: cat ~/.bashrc

**umask** value set: # Control default access permission

umask 0066

Write the **rwx** permissions taken away by **umask** value above: -rw------

Create a new directory named **test04** under **asgn04**. Determine the default access permissions on the directory at the time of creation.

Command to create a new directory **test04**: mkdir test04

Command, long listing, of (only) **test04** showing its access permissions: ls -l

Access permissions on **test04** (write all 3 groups of 3 chars): drwx--x--x

Create a new directory named **test04** under the **test04** directory created in the above step. Remember that you are not allowed to change your directory unless explicitly allowed. Also note that you are creating a second directory named **test04** under **test04**. That is perfectly fine.

Command to create a second **test04**: mkdir -p test04/test04

1. Again, without changing directory, using the **touch** command, create a hidden file of size 0 named **.secret** under the lower level (second) **test04** directory.

Command to create **.secret**: touch test04/test04/.secret

Set access permissions of **.secret** so that no one, including you, has any permission.

Command to set permission: chmod 0 test04/test04/.secret

1. With one command, using relative path, do a directory listing in long format of the file **.secret** alone that you created in Step 3.

Command for directory listing of **.secret**: ls -l test04/test04/.secret

Since you are naming the file, there is no need for **–a** option even though it is a hidden file. Record the output below. Write the entire line.

---------- 1 etay1 domain users 0 Sep 27 23:45 test04/test04/.secret

1. Change the access permission for lower level (second) **test04** to remove the **x** permission on that folder. That is, it will end up with access permission **rw- --- ---**

Command to change permission: chmod 600 test04/test04

Attempt to switch to the lower level (second) **test04** directory using the **cd** command. The command will fail. Record the error message and reason for failure. (Use 2. Access Permissions)

Command used: cd test04/test04

Error message: -bash: cd: test04/test04: Permission denied

Reason for failure: removes directory permissions also

Attempt to redo the command to display all hidden and unhidden items in the lower level directory **test04** with a long listing format. The command will fail. Record the error message and reason for failure. (Use 2. Access Permissions)

Command used: ls -l test04/test04/.secret

Error message: ls: cannot access test04/test04/.secret: Permission denied

Reason for failure: same reason removes the directory permissions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Change the access permission for lower level (second) **test04** to add the **x** permission on that folder. That is, it will end up with access permission **rwx --- ---**

Command to change permission: chmod 700 test04/test04

Redo the command for a directory listing in long format for displaying the **.secret** file. The command will work fine.

Command used: ls -l test04/test04/.secret

Output: ---------- 1 etay1 domain users 0 Sep 27 23:45 test04/test04/.secret

Try to delete the file **.secret** using the **rm** command. You will receive a prompt. Respond “yes”.

Command to delete the file **.secret**: rm test04/test04/.secret

Prompt seen: rm: remove write-protected regular empty file ‘test04/test04/.secret’?

At this stage, the file has been deleted. The prompt could have been avoided if we used the **–f** option. In other words, to delete a file, you do not need any permission on the file itself.

Redo the command for a directory listing in long format of the **.secret** file. Now you will receive an error message as the file does not exist.

Command used: ls -l .secret

Output: ls: cannot access .secret: No such file or directory

1. Recreate the **.secret** file of size 0 as before under the lower level (second) **test04** directory. No need to change the access permissions for **.secret** now.

Command to create **.secret**: touch test04/test04/.secret

Attempt to remove the lower level (second) **test04** directory using the **rmdir** command. The command will fail.

Command used: rmdir test04/test04

Error message: rmdir: failed to remove ‘test04/test04’: Directory not empty

Attempt to remove the lower level (second) **test04** directory using the **rm** command with no options. The command will fail.

Command used: rm test04/test04

Error message: rm: cannot remove ‘test04/test04’: Is a directory

Now remove the lower level (second) **test04** directory and all the files and directories underneath by using the **–r** option for **rm** command tried above.

Command used: rm -r test04/test04

1. Now there is nothing below the top level (first) **test04** folder. Change its access permissions to remove the **w** permission. That is, at the end, it will have **r-x --- ---** permission.

Command to change permission: chmod 500 test04

Do a directory listing of (only) **test04** folder in long format to show its properties.

Command: ls -l test04

Output: total 0

Attempt to create a file named **.secret** of size 0 under **test04** directory (lower level) using the **touch** command. The command will fail. Record the error message and the reason for failure.

Command to create the **.secret** file: touch test04/.secret

Error message: touch: cannot touch ‘test04/.secret’: Permission denied

Reason for failure: we removed the permission to write in step 8 part 1

Because it is empty, the **test04** directory can be removed with a simple **rmdir** command. Remove the **test04** directory.

Command used: rmdir test04

Do a directory listing in long format showing all entries of the **asgn04** folder, which is your current working directory. There should be exactly two lines of output. Record the output.

Command used: ls -l

Output: total 0

We just removed both test04 and the lower level test04 so there shouldn’t be anything here

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. We want to create a file with exactly eleven lines, without using any editor, that counts from 0 to 10 in English and Spanish. Use the command **cat** with output redirection as follows:

**cat > countInSpanish**

Then type eleven lines precisely as follows: type **English version, tab key, Spanish version, enter key.** Do not enter anyextra blank or other characters.

zero cero

one uno

two dos

three tres

four cuatro

five cinco

six seis

seven siete

eight ocho

nine nueve

ten diez

After the eleven lines, type: **Control-D** to signal end of keyboard input.

Command to create the file **countInSpanish**: cat > countInSpanish

How to signal end of keyboard input? Control-D

1. To check that there are exactly eleven lines and to ensure that there are no trailing blanks, display the contents of the file with the **cat** command using two options: **–n** option and **–E** option.

Command to display file: cat -n -E

What character appears at the end of each line? $

Note that the $ symbol is used to show the end of the line. It is not part of the file itself.

11. Execute the **wc** command on the file and report the result observed.

Command: wc -wlc countInSpanish

Output observed: 11 22 112 countInSpanish

Number of lines: 11 Number of words 22 Number of bytes 112

1. Type **od** command with **–c** option on the file. It will display the file character by character and also interpret the control characters. You should see the tab and newline characters. (If you see any blank character in the file, you did not create the file as required. You have to redo the steps. For that, first delete the **countInSpanish** file and start all over at Step 9.)

Command: od -c

Visually count the number of new line, tab and blank characters seen. Also, is there any special character at the end of the file to mark the end of the file?

Number of new line characters seen: 11

Number of tab characters seen: 11

Number of blank characters seen: 16

Is there any special character at the end of the file to mark the end of the file? no

13. Display the top 6 lines of the file.

Command to display top 6 lines of the file: head -n 6 countInSpanish

Display the bottom 4 lines of the file.

Command to display the bottom 4 lines of the file: tail -n 4 countInSpanish

14. Type the following command: **head -9 countInSpanish | tail -3**

Here the symbol **|** is the vertical stroke over the backslash key, known in UNIX as a pipe symbol. Report what you observe.

Command with pipe typed: head -9 countInSpanish | tail -3

What did you observe?

six seis

seven siete

eight ocho

Explain why the command displayed only those three lines.

Calls the bottom 3 OF the top 9

15. Without changing directory, with one command, copy the file **helloCount.c** in my directory **~nyu/csc209/asgn04** into your **asgn04** directory but with the file name changed to **countHello.c**.

Command to copy file: cp ~nyu/csc209/asgn04/helloCount.c .

Rename your file **countHello.c** to **helloWithCount.c**

Command to rename file: mv helloCount.c helloWithCount.c

Verify the success of your copy and renaming with an **ls** command (long option) on that file.

Command to verify success in copying and renaming: ls -l helloWithCount.c

Output: -rwx------ 1 etay1 domain users 335 Sep 28 00:30 helloWithCount.c

16. Compile the program **helloWithCount.c** so that the object file is named **helloWithCount** and the standard error file is **redirected** to a file named **error**.

Use the command: **cc helloWithCount.c –o helloWithCount 2> error**

Note that the C compiler (**cc**) compiles the program file (source file). All syntax errors are redirected to standard error which is what we are redirecting to the file **error**. If there are no errors the object file is created as directed **(-o** option).

Display the contents of the file **error**

Command to display file: cat error

17. If there are no errors, the file error will be empty. In this case, you will see several errors reported. In the displayed output, you may see strange **â** characters. They are supposed to be webpage apostrophe characters. Ignore them. If you read the messages carefully, there are really only two errors, in lines 10 and 15. Identify them.

helloWithCount.c: In function ‘main’:

helloWithCount.c:10:8: error: expected ‘;’ before ‘return’

return -1;

^

helloWithCount.c:15:20: error: ‘countt’ undeclared (first use in this function)

for (i = 0; i < countt; i++)

^

helloWithCount.c:15:20: note: each undeclared identifier is reported only once for each function it appears in

Display the C program file **helloWithCount.c** with line numbers.

Command: cat -n helloWithCount.c

18. The **pico** editor is intuitive to use. Unfortunately it may not be available in all UNIX systems. I recommend you learn to use other editors, such as **vi** and even **Emacs**. The **nano** editor is very similar to **pico**. In any case, for this part, use **nano** to edit the program file and fix the two obvious errors. To start **nano**, type **nano** followed by the name of file to be edited. The menu bar at the bottom of the **nano** window should tell you how to save the file after correction and how to exit.

Command to edit source file: nano helloWithCount.c

How to write out? Control o How to exit? Control x

19. Attempt to recompile the code again, exactly as in Step 16. You will receive an error message as you are attempting to redirect error messages to a file named **error** that already exists. Normally, UNIX would have overwritten the file when you redirect like this (much like the **cp** command). However, because we have set the **noclobber** option in the **.bashrc** file, such accidental erasure of files is protected for us.

Error message:

helloWithCount.c: In function ‘main’:

helloWithCount.c:10:8: error: expected ‘;’ before ‘return’

return -1;

^

helloWithCount.c:15:20: error: ‘countt’ undeclared (first use in this function)

for (i = 0; i < countt; i++)

^

helloWithCount.c:15:20: note: each undeclared identifier is reported only once for each function it appears in

Allow temporary overwriting in this command by typing:

**cc helloWithCount.c –o helloWithCount 2>| error**

**I tried this and nothing seemed to occur -> Rest of the lab is dependent on this it seems**

See the vertical stroke after **>** symbol.

Display the contents of the file **error**

Command to display file: cat error this isn’t right is it?

It should be empty, indicating no more syntax errors. (If not, repeat Steps 18 and 19.)

20. Execute the compiled object code created in the previous step, without supplying any arguments. The code will run and show the correct usage.

Command to execute the compiled object code with no arguments: ./helloWithCount.c

Output observed:

./helloWithCount.c: line 4: syntax error near unexpected token `('

./helloWithCount.c: line 4: `int main (int argc, char\* argv[ ])'

Re-execute the command with your first name and 10 as the two command line arguments.

Command to run again with arguments: ./helloWithCount.c elijah 10 10

Describe the output:

./helloWithCount.c: line 4: syntax error near unexpected token `('

./helloWithCount.c: line 4: `int main (int argc, char\* argv[ ])'

Re-execute the command with your first name and 10 as the two command line arguments, but providing them in the wrong order, that is, number first and then your name.

Command to run again with arguments: ./helloWithCount.c 10 10 elijah

Describe the output:

./helloWithCount.c: line 4: syntax error near unexpected token `('

./helloWithCount.c: line 4: `int main (int argc, char\* argv[ ])'

Use the source code, helloWithCount.c to determine why it didn’t work:

Syntax error in line 4 and possibly a misspelling of a declared variable so it failed to reference it properly.

(No points off on this last effort)