**CSC 209 UNIX Tools**

|  |  |  |
| --- | --- | --- |
|  | **Assignment 5** |  |

**Objectives: Redirection, Pipes, and Filename Wildcards**

**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 holly2 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 **asgn05** under **csc209**. Set the access permissions for **asgn05** as **rwx --- ---** by typing a command:

**chmod 700 asgn05**

Command to create the **asgn05** folder: mkdir asgn05

Command to set the access permissions: chmod 700 asgn05

Move to **asgn05** and determine the current working directory.

Command needed to switch from **csc209** directory to **asgn05**: cd asgn05

Command to print the working directory: pwd

Your current working directory: /home/etay1/csc209/asgn05

Shell prompt: [58] [etay1@courses2016:~/csc209/asgn05]$

**From now on, unless explicitly permitted, you should not move away from this asgn05 directory.**

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

**and move to asgn05 before you continue.**

2. Use an editor to create a file named **fruits** with ten lines, each line naming one fruit, specifically in this order: orange, apple, banana, peach, kiwi, pineapple, grape, strawberry, cherry, and mango. The file should have exactly ten lines, i.e., no blank lines at the end.

Command to create the file: pico fruits

Use the **wc** command with suitable option to check the number of lines. Report output.

Command to check the number of lines: wc -l fruits

Output of the above command: 10 fruits

If the number of lines is not ten, do this step again.

3. Use **cat** command with output redirection to create a file named **vegetables** with the following six lines, each line naming one vegetable, specifically in this order: okra, eggplant, beans, peas, carrot, and pepper. (See Assignment 4) The file should have exactly six lines, i.e., no blank lines at the end.

Command to create the file: cat > vegetables

How did you terminate keyboard input? Control D

Use the **wc** command with suitable option to check the number of lines. Report output.

Command to check the number of lines: wc -l vegetables

Output of the above command: 6 vegetables

If the number of lines is not six, do this step again

4. Write one command that works as follows: Concatenate the contents of the two files fruits and vegetables and redirect the output to file named **tops**.

Command to create the file **tops**: cat fruits vegetables > tops

Display the contents of **tops** with line numbers and check if it contains 16 lines as expected.

Command to display the file: cat -n tops

What was displayed? Describe: both lists: fruits and vegetables

Write the items in the order of appearance:

1 orange

2 apple

3 banana

4 peach

5 kiwi

6 pineapple

7 grape

8 strawberry

9 cherry

10 mango

11 okra

12 eggplant

13 beans

14 peas

15 carrot

16 pepper

Use the **wc** command with suitable option to check the number of lines. Report output.

Command to check the number of lines: wc tops

Output of the command: 16 16 110 tops

5. Write one command that works as follows: invoke **sort** command with input redirection, taking input from **tops**, and with the output is redirected to **sortops**.

Command to create the file **sortops**: sort < tops > sortops

Display the contents of **sortops** with line numbers and check if it contains 16 lines as expected.

Command to display the file: cat -n sortops

What was displayed? Describe: the same list except alphabetized

Write the items in the order of appearance:

1 apple

2 banana

3 beans

4 carrot

5 cherry

6 eggplant

7 grape

8 kiwi

9 mango

10 okra

11 orange

12 peach

13 peas

14 pepper

15 pineapple

16 strawberry

Use the **wc** command with suitable option to check the number of lines. Report output.

Command to check the number of lines: wc sortops

Output of the command: 16 16 110 sortops

6. Use the **ls** **–1** command – **here the option is digit one and not lower case L** – to display all non-hidden entries in **asgn05**, one per line. Count the number of non-hidden entries and report.

Command to display non-hidden entries in **asgn05**, one per line: ls -1

Number of non-hidden entries under **asgn05**: 4

List the non-hidden entries:

fruits

sortops

tops

vegetables

We know that **wc –l** command – here the option is lower case L – will count the number of lines in its input. Using the two commands seen in this step, write one compound command with a pipe that displays a single number indicating the number of non-hidden entries in **asgn05**.

Command with pipe to display the number of non-hidden entries under **asgn05**:

ls -1 | wc -l

Output of the command: 5

7. Write one compound command with a pipe that works as follows: invoke **sort** command with input redirection, taking input from the file **tops.** The output is piped to **head** command with an option **-9** to choose the top nine lines.

Command with pipe: sort < tops | head -9

Write the items in the order of appearance:

apple

banana

beans

carrot

cherry

eggplant

grape

kiwi

mango

If the list is not **apple . . mango**, do this step again.

Pipe the output of the above command to the **tail** command with option **-2** to choose the bottom two lines.

Command with two pipes: sort < tops | head -9 | tail -2

If the list is not **kiwi mango**, do this step again.

Now, write one compound statement with input direction, two pipes, and output redirection that works as follows: invoke **sort** command with input redirection, taking input from the file **tops.** The output is piped to **head** command with an option to choose certain number of lines. The output of the head command is piped to **tail** command to choose a certain number of lines. The output of the tail command is redirected to a file named **mid4**. The file **mid4** should contain only the four items in the middle of **sortops** listed above (i.e., lines 7-10 of **sortops**). Determine the options to be used for the **head** and **tail** commands to get the job done.

Command with input redirection, two pipes, and output redirection to create **mid4**:

sort < tops | head -10 | tail -4 sortops >| mid4

Display the contents of **mid4** and check if it contains 4 lines as expected.

Command to display **mid4**: cat mid4

List the items in the order of appearance:

peas

pepper

pineapple

strawberry

8. Use the **cal** command to display the calendar for July 2021 and redirect the output to a file named **summer21first**.

Command to redirect July 2021 calendar: cal 7 2021 > summer21first

Provide another command to display the calendar for June 2021 and redirect the output to the same file **summer21first**.

Do you see any error message? Explain why you have the error message.

Command to redirect June 2021 calendar: cal 6 2021 > summer21first

Error reported: -bash: summer21first: cannot overwrite existing file

Reason for error: can’t be overwritten > file already exits

Give a command that will rewrite the **summer21first** with the calendar for June 2021, without reporting any error, i.e., temporarily overriding the **noclobber** option.

Command that will redirect as required without error cal 6 2021 >| summer21first

9. Write another command to display the calendar for July 2021 and append the output to the file **summer21first.**

Command to append the July 2021 calendar: cal 7 2021 >> summer21first

Write another command to display the calendar for August 2021 and append the output to the file **summer21first**.

Command to append August 2021 calendar: cal 8 2021 >> summer21first

10. Accomplish the entire task of putting the calendar for June - August 2021 in a file by writing three **cal** commands for each of the three months of 2021, with these commands separated by semicolons and grouped together with parentheses, and finally redirecting the output of the entire group of commands to a file named **summer21second**.

One line command to put the calendar for June-August 2021 in a file:

(cal 6 2021; cal 7 2021; cal 8 2021) >| summer21second

11. Compare the output of **summer21first** and **summer21second** to see if they are identical. Use **cmp** command. If the files are identical, the command will provide no response.

Command to compare the two files **summer21first** and **summer21second**:

cmp summer21first summer21second

Result of comparison: nothing happens

Are the two files identical? yes

Use **pico** and edit **summer21first**. Insert a blank line after the line displaying August 2021. Now compare the two files and report your finding.

Command to edit **summer21first**: pico summer21first

Command to compare the files: cmp summer21first summer21second

Result of the comparison:

summer21first summer21second differ: byte 1, line 1

12. By connecting two commands by **||** or **&&** we can indicate conditional execution.

In **cmd1 || cmd2** we are looking for the successful execution of either **cmd1** or **cmd2**. Hence, if **cmd1** is successful, we skip **cmd2**. In other words, execute **cmd2** only if **cmd1** fails.

In **cmd1 && cmd2** we are looking for the successful execution of both **cmd1** and **cmd2**. Hence, if **cmd1** fails, we skip **cmd2**. In other words, execute **cmd2** only if **cmd1** succeeds.

For each of the following four cases, think first and record your expected output, including error messages. Here, **bingo** is an invalid UNIX command. On the other hand, **echo** is a valid command that will succeed.

|  |  |
| --- | --- |
| **Command** | **Expected Output** |
|  |  |
| **echo first || echo second** | -------------------------🡪 first |
| **echo first && echo second** | -------------------------🡪 second |
| **bingo || echo second** | -------------------------🡪 second |
| **bingo && echo second** | -------------------------🡪 nothing |

Type the command and report the output.

|  |  |
| --- | --- |
| **Command** | **Observed Output** |
|  |  |
| **echo first || echo second** | -------------------------🡪 first |
| **echo first && echo second** | -------------------------🡪 first -------------------------🡪 second |
| **bingo || echo second** | -------------------------🡪 second |
| **bingo && echo second** | -------------------------🡪 not found |

13. With one **touch** command, create five hidden files: **.image1.jpg**, **.image2.jpg**, **.image3.jpg**, **.image4.jpg**, and **.image5.jpg**

Command to create hidden files: touch .image1.jpg .image2.jpg .image3.jpg .image4.jpg .image5.jpg

With one touch command createsix non-hidden files with names: **image.jpg**, **image6.jpg**, **image7.jpg**, **image8.jpg**, **image9.jpg**, and **image10.jpg**.

Command to create non-hidden files: touch image.jpg image6.jpg image7.jpg image8.jpg image9.jpg image10.jpg

With one touch command create a non-hidden file named: **2image2**

Command to create **2image2** file: touch 2image2

Create a directory by the name **d**

Command to create the directory: mkdir d

Without changing your directory, using a touch command, create a file named **test.jpg** inside the folder **d**.

Command to create a file inside the folder: touch d/test.jpg

14. Type the command **ls image[5-7]\*** and report what you see.

Output of **ls image[5-7]\***

image6.jpg image7.jpg

(list all entries)

Type **ls image[13579]\***  and report what you see.

Output of **ls image[13579]\*** image10.jpg image7.jpg image9.jpg

(list all entries)

15. Write one **ls** command with wildcard(s) to provide a listing of all non-hidden files in **asgn05** whose name begins with **image**

Command to list non-hidden files whose name begins with **image**

ls image\*

Output of the above: image10.jpg image6.jpg image7.jpg image8.jpg image9.jpg image.jpg

(list all entries)

16. Write one **ls** command with wildcards to provide a directory listing of all (hidden as well as non-hidden) files in **asgn05**, whose name ends with **.jpg**

Command to list all (hidden as well as non-hidden) files whose name ends with **.jpg**

ls .\*.jpg \*.jpg

Output of the above:

image10.jpg .image2.jpg .image4.jpg image6.jpg image8.jpg image.jpg

.image1.jpg .image3.jpg .image5.jpg image7.jpg image9.jpg

(list all entries; there should be 11)

17. Write one **ls** command with option **-1** (digit one) and with wildcards to provide a directory listing of all (hidden as well as non-hidden) files that include the string **image** in the name (in the beginning, middle or at the end). Note that the option **-1** (digit one) for the **ls** command will place each entry in a separate line.

Command to list all (hidden as well as non-hidden) files with the string **image** in the name:

ls -1 .\*image\* \*image\*

Output of the above:

2image2

image10.jpg

.image1.jpg

.image2.jpg

.image3.jpg

.image4.jpg

.image5.jpg

image6.jpg

image7.jpg

image8.jpg

image9.jpg

image.jpg

(list all entries

Write one compound command with a pipe to count the number of hidden as well as non-hidden files that includes the string **image** in the name (in the beginning, middle or at the end). Note that the option **-1** (digit one) for the **ls** command will place each entry in a separate line. So we simply have to count the number of lines.

Command with pipe to count the number of hidden as well as non-hidden files with the string **image** in the name:

ls -1 .\*image\* \*image\* | wc -l

Output of the above: 12

18. Type the command **ls \***  and report what you see. Which files are listed?

Output of **ls \*** command:

2image2 image6.jpg image9.jpg sortops tops

fruits image7.jpg image.jpg summer21first vegetables

image10.jpg image8.jpg mid4 summer21second

d:

test.jpg

(list all entries)

Are hidden files listed? No

Type the command **ls ?** and report what you see. Which files are listed?

Output of **ls ?** command: test.jpg

(list all entries)

Are hidden files listed? no

19. Type the command **echo \*** and report what you see. What files are listed?

Output of the command **echo \***

2image2 d fruits image10.jpg image6.jpg image7.jpg image8.jpg image9.jpg image.jpg mid4 sortops summer21first summer21second tops vegetables

Identify the difference between the output of **ls \*** and **echo \***

echo \* is alphabetized and ls \* is not

Type the command **echo \\***

Output of the command **echo \\***  \*

1. Bash is only one of the shells available in a UNIX system. There are other shells such as Bourne Shell, C Shell, Korn Shell, etc. On holly2, the AT&T UNIX Bourne shell (**sh**) is not available. We are using **bash**, which is an enhanced and compatible version of sh (http://www.gnu.org/software/bash/). Likewise, the Berkeley UNIX C shell (**csh**) is not available on holly2. Instead, we can use **tcsh**, which is an enhanced, but completely compatible version of csh (http://www.tcsh.org/Welcome).

Most of the basic commands are identical in these shells; but there are also small differences. Further they use different sets of initialization files. To use **tcsh**, type the command **/bin/csh** What is the prompt for **csh** that you see?

Command to switch to **tcsh**: ­­­­­­­­­­­­­­­­­­­ /bin/csh

Prompt in **tcsh**:

[1] [etay1@courses2016]%

Type **cd ~; pwd**

You will move to your login directory and print the absolute path to it.

Output: /home/etay1

Exit **tcsh** by typing the command **exit**.

Command to exit **tcsh**: exit

You are back in bash. Type **pwd** again and see that you are back in **asgn05**.

Output of **pwd** command: [78] [etay1@courses2016:~/csc209/asgn05]$