**CSC 209 UNIX Tools**

|  |  |  |
| --- | --- | --- |
|  | **Assignment 1** |  |

**Objectives: Login, logout, setting up initialization files, and simple UNIX commands**

**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 encourage you to **type** the commands rather than copy-n-paste. It is not only because it can force you to write and remember the UNIX commands used and the results observed, but also because that copy-n-paste often causes errors for text encoding issues.

1. In the next step, you will login on a UNIX server (also called a host) named **courses**. It is a Dell server machine that runs Red Hat Linux. Since this machine replaced an existing one named holly, this server is also known by the name **holly** or **holly2**. On the Internet, the fully qualified domain name for this machine is **courses.brockport.edu**

The aliases **courses.brockport.edu** or **www.itss.brockport.edu** may also be used to refer to the same machine.

To login, use any Windows personal computer in Drake. For command line login, you will normally use the **telnet** software and access the UNIX server. However, telnet protocol transmits passwords in plain text form which may be captured and misused. Hence, telnet access to Courses has been blocked for security reasons. You need to use a secure shell client, such as **PuTTY**, a free executable SSH and Telnet client tool, <http://en.wikipedia.org/wiki/PuTTY>, or <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> ). If you have a home computer with Internet access, you may want to download, install, and configure PuTTY for access to courses.brockport.edu or holly2.brockport.edu

In Drake, look under Internet Utilities for PuTTY and launch it. Choose the configuration Courses stored on it and click Load. Then click Open to connect to the UNIX server.

Write your NetID: etay1

1. Login to courses server using your Webmail Net-ID and password.
   1. Since we are not using the graphical user interface (GUI), you have to rely on your knowledge of UNIX commands. In order to indicate that the command interpreter (**shell**) is ready to receive your next command, it puts out a **prompt**. Take a note of how the command prompt placed by the shell appears and report below:

Shell prompt: [etay1@courses2016 ~]$

* 1. In many cases, user accounts are created to invoke the **C Shell** (**csh**; actually a more advanced **tcsh**) or a Bash (bash; **Bourne Again Shell**) login. In order to see what shell you are currently using, type the command **echo $0** and report:

Shell used: Bash

* 1. As you login, you will be placed in your login (home) directory in the file system. Type the command **echo $HOME** to print your login (home) directory.

Login (home) directory:­­­­­­­­­­­­­­­­­­­­­­­­­­/home/etay1

It will be a string of the form **/home/semester\_of\_joining/Net\_ID**

* 1. You may move away from the login (home) directory to other directories later. But at the time of login, the working directory is the same as your login (home) directory. Type the command **pwd** to print your working directory:

Current working directory: /home/etay1

1. On a UNIX system, there are a variety of shells available for use. The most advanced one is the Bourne Again Shell (**bash**). We will use bash in this course and in order to ensure that all students have similar set up, copy the configuration files I have in my login directory to your login directory, as shown below.

Type the following five commands, one by one, exactly as shown below. Before you press the enter key, make sure you are making no mistakes. Note that the file names for these files start with period. Also see the tilde (**~**) symbol in front of my Net\_ID. As each command is executed, if done properly, the shell will display the command prompt to ask for the next command. If you see any error message, stop and check with me.

**cp ~nyu/.cshrc ./.cshrc**

**cp ~nyu/.login ./.login**

**cp ~nyu/.bashrc ./.bashrc**

**cp ~nyu/.bash\_logout ./.bash\_logout**

**chmod 600 .cshrc .login .bashrc .bash\_logout**

1. Type the command given below and report the output. Note that the symbol after the hyphen is lower case L and not digit 1. Write the entire line rather than copy-n-paste, do not skip any field. ls is used to show the information about files and directories.

**ls –l .cshrc .login .bashrc .bash\_logout ls**

-rw------- 1 etay1 domain users 97 Aug 31 10:26 .bash\_logout

-rw------- 1 etay1 domain users 536 Aug 31 10:26 .bashrc

-rw------- 1 etay1 domain users 226 Aug 31 10:25 .cshrc

-rw------- 1 etay1 domain users 81 Aug 31 10:25 .login

1. Logout by typing the command **logout**
2. Log back in on courses server. The prompt would have changed now. Besides we are using **Bourne Again Shell** (**bash**) now.

Shell prompt:

The prompt will have an event number within square brackets. Every command you type will get a number, so we can recall a specific command later. The prompt will also include within square brackets, your Net\_ID and machine name, separated by @. This sequence is followed by a colon and your current working directory. Thus, by looking at this part, you can always be sure if you are in the right directory before you type a command. At this stage, you will see the tilde (~), which is the notation for your home (login) directory. More precisely, **~nyu** stands for the login (home) directory of the user named **nyu**. If no user name is given, it is the login (home) directory of the person logged in. Finally, you will see a **$** sign and a blank space as the last two characters of the prompt.

As a result of the change made to the initialization files **.cshrc**, **.login**, **.bashrc**, and **.bash\_logout**, the environment has been customized to be the same for all of us. If you do not see this change, repeat Steps 1-4 carefully before proceeding further.

1. Let us try a number of simple commands. Type each one of them and write the output seen.

|  |  |
| --- | --- |
| **Command** | **Output** |
| **echo $0** | -bash |
| **echo $HOME** | /home/etay1 |
| **pwd** | /home/etay1 |
| **date** | Wed Aug 31 10:32:11 EDT 2022 |
| **echo $USER**  **whoami** | etay1  etay1 |
| **hostname** | courses2016.brockport.edu |
| **uname** | Linux |
| **cat .login** | Exec /bin/bash -l |
| **wc .login** | 3 17 81 .login |

1. Of the ten commands in the list above, five of them (**pwd**, **date**, **whoami**, **hostname**, and **uname**) did not specify any argument. The other five had one argument. None had any options. (Options are preceded by hyphen and are single characters in UNIX.) The general format of a UNIX command is as follows:

**command name followed by zero or more options, followed by zero or more arguments**

This is often written formally as

**command [ option ] ... [ argument ] ...**

The square bracket around **option** means options are not always required; i.e., there may be no options. The ellipses ( **...** ) mean that options may be repeated. Likewise, square bracket around **argument** means arguments are not always required; i.e., there may be no arguments. The ellipses ( **...** ) mean that arguments may be repeated.

For example, the command:

**wc –l -w .cshrc .login .bashrc .bash\_logout**

has the command name (**wc**) followed by two options (**-l** and **-w**), followed by four arguments (**.cshrc**, **.login**, **.bashrc**, **.bash\_logout**).

The purpose of the **wc** command is to print the number of lines, words, and bytes in files. In ASCII files, the end of the line is denoted by the ASCII character **LF** (line feed or newline) with binary code **0001010**. Since it is a control character that cannot be displayed, it is denoted as '\**n'**. Each ASCII character occupies one byte. Thus, the number of characters in the file is the same as the number of bytes of the file.

The **wc** command, with the options, **-l** and **–w**, it prints only the number of lines and number of words. Note: It is allowed to combine the two options as **–lw** Type the command as shown above and report the findings below.

|  |  |  |
| --- | --- | --- |
| **File** | **# of lines** | **# of words** |
| **.cshrc** | 9 | 36 |
| **.login** | 3 | 17 |
| **.bashrc** | 27 | 63 |
| **.bash\_logout** | 4 | 17 |
| **Total** | 43 | 133 |

1. UNIX provides extensive online help in the form of manual pages. The command **man wc** will help you look up the manual pages regarding the command **wc** and determine all the options. While reading the man pages, press the enter key to advance the display by one line; press the space bar to move it up by one screen; finally type **q** to quit the man page display. Try the command now to determine the option to be used to print the length of the longest line.

Option identified: -L, --max-line-length

print the length of the longest line

Execute the **wc** command with this option to determine the length of the longest line for the two files **.login** and **.bashrc**

Exact command typed, with options and arguments: wc -L .login .bashrc

Length of the longest line in **.login**: 36

Length of the longest line in **.bashrc**: 61

What do you think the line total stands for? Line with the greater value

1. After you logged back in, you have not moved from your login (home) directory. We will do so in the next few steps.
   1. Verify that you are still in the login (home) directory by looking at the prompt. Type the appropriate command to print the working directory and report.

Command to print the current working directory: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output of the command: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Type the following command to set access permissions for your current working directory appropriately: **chmod 711 .**

Note that **~** (tilde) stands for your login (home) directory. Likewise **.** ( a single period) stands for the current working directory, which may or may not be your login (home) directory. Further, **..** (two periods together) stands for the parent directory of the current working directory.

* 1. Type the command **ls –ld ~** to obtain the long listing of the login (home) directory entry alone, showing access permissions and other details. Write the entire output.

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

* 1. Type the command **ls –ld public\_html** to verify if a directory named **public\_html** exists under your home directory and, if it does exist, what access permissions are associated with it.

If the directory does not exist, an error message will be printed out. If an error message appears, type **mkdir public\_htm**l to create (make) this directory named **public\_html** under your login (home) directory.

Whether or not the directory is newly created, set its access permissions by typing the command: **chmod 711 public\_html**

* 1. Type the command **ls –ld public\_html** to obtain the long listing of the **public\_html** directory entry alone, showing access permissions and other details. Write the entire output.

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

* 1. Move to the **public\_html** directory by typing: **cd public\_html**

The command **cd** stands for change directory. Once you successfully execute this command, you will see that the prompt has changed.

Shell prompt: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Now, under **public\_html**, create two subdirectories named **csc209** and **cgi-bin** and set their permissions to **711**. Use appropriate commands. Note: Both directories can be created at the same time by giving two arguments to the **mkdir** command.

Command to create csc209 and cgi-bin: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Command to set access permissions for both: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Type the command **ls –l .** to obtain the long listing of the files and directories under **public\_html**, which is your current working directory.Note how this command is different from the one used in step d above, in terms of options. We are not using option d here. That makes the difference. The option d forces the system to look at the directory **public\_html** itself, rather than the files and directories underneath **public\_html**. Write out the two lines corresponding to **csc209** and **cgi-bin**.

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

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

* 1. Move to **csc209**.

Command to do so: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Shell prompt after move: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Copy the file **index.html** in my directory **~nyu/public\_html/csc209/index.html** into your **csc209** directory and set the permissions to **644** with the following two commands, typed one after another:

**cp ~nyu/public\_html/csc209/index.html ./index.html**

**chmod 644 index.html**

* 1. Use the **pico** editor, which has a fairly intuitive command structure (see the bottom of the screen), to make appropriate changes that reflect information about you. Start the editor using the command **pico index.html** After you are done with the editing, type Control-O (that is, hold down the control key and press O) to write out (save) the file. Confirm the name of file before you save (hit Enter key). Type control-X to exit the editor.

Systematically go through each line and make appropriate changes. (There are at least five lines in which changes are needed – name in two places and email address in three places. You may or may not change the phone number; remember that this web page can be seen by any one.)

* 1. The URL for the web page you created is

**http://www.itss.brockport.edu/~xxxxx/csc209/index.html**

with **xxxxx** replaced by your Net-ID. Note the **~** symbol in front of the Net-ID.

URL for your web page: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The format of a Web URL is as follows:

**protocol://web\_server\_address/document\_location\_on\_server**

Observe the absence of **public\_html** in the path specified above for document location. Normally **~jdoe1** stands for the login (home) directory of **jdoe1**. In the context of the Web alone **~jdoe1** stands for the **public\_html** directory under the login (home) directory of **jdoe1**, because we have configured our Web server software that way.

Open Internet Explorer, Mozilla Firefox, or Chrome, enter the URL**,** and check if the web page you created displays properly.

Does the page display properly: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If not, please check with me, and fix the problem.

1. It is important to keep the files you create for various assignments organized in proper subdirectories.

a. Move up to your home directory by typing the command: **cd ~**

Check the shell prompt to ensure that you are indeed in your login (home) directory.

b. Create a subdirectory named **csc209** under your login (home) directory. This is different from the **csc209** directory you created under **public\_html**. Ensure that the new directory has **700 access** permissions.

Command to create the **csc209** directory under your login (home) directory:

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

Command to set permission for **csc209** directory: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

c. Move down to **csc209** directory.

Command to do so: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Check the prompt to confirm after successful move.

d. Under **csc209**, create a subdirectory for the first assignment. That is, create **asgn01** and ensure that it has the same permissions as **csc209**.

Command to create the directory **asgn01** under **csc209**:

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

Command to set permission for **asgn01** directory: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

11. Move down to **asgn01** directory.

Command to do so: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Check the prompt to confirm after successful move.

Shell prompt at this stage:

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

12. From now on, be sure to place all new files relating to this assignment (except **index.html** discussed above) under **asgn01**

1. Below is an example of a Bash script program. Use the **pico** text editor and create a file named **hello.bash.** Start the editor using the command **pico hello.bash** and type in thefollowing six lines.

**#!/bin/bash**

**# print a one-line greeting**

**#**

**echo "Hello, World!"**

**exit 0**

**#**

1. Try to execute the shell script by typing the name of the file, i.e., type **hello.bash**
2. The script will not execute. Record the error message.

Error message: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. The shell script program did not execute because the file is not executable. In order to check the access permissions for this file, type the command **ls –l hello.bash** and write the output.

Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Note that the first character in the line is a **–** and not **d**. This means that **hello.bash** is an ordinary file and not a directory, which is fine. The next three characters indicate the access permissions you have for the file. They indicate **rw–** which means you can read and write the file, but cannot execute it. It should be changed to **rwx** to stand for read, write, and execute.
2. To ensure that the script program file has **execute** permission for the owner (you), besides read and write, type the command: **chmod 700 hello.bash** The Bash Shell script program created above can now be executed by simply typing the name of the executable file **hello.bash**
3. See that it prints one line saying **Hello, World!**

Output of the program: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13. Type the command **man cal** and see the manual pages for the **cal** command, which is meant to print a calendar. Read it. Determine what arguments you need to use to display the calendar for the month of July in 1776 from your reading. Once successfully displayed, determine which day of the week July 04, 1776 fell.

UNIX command with arguments needed to display calendar for July 1776: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Day of the week in which July 04, 1776 fell: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14. Determine the UNIX command with arguments needed to display calendar for the entire year 2014.

UNIX command with arguments needed to display calendar for the year 2014:

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

15. Determine the commands for the indicated tasks. Complete the table.

|  |  |  |
| --- | --- | --- |
|  | **Purpose** | **Command** |
| 1 | Print current working directory |  |
| 2 | Determine the number of lines, words, and bytes in a file named **test** |  |
| 3 | Display the contents of the file named **test** |  |
| 4 | Create a new directory named **junk** under your current working directory |  |
| 5 | Change the access permission of directory **junk** to 700 |  |
| 6 | Do a directory listing in long format of the directory **junk** as the only entry |  |
| 7 | Do a directory listing in long format of the files and directories under the directory **junk** |  |
| 8 | Display the man pages for the **cat** command |  |