**L A B. 1: Shell Programming**

**CSC305**

**Instructor:Jonathan Parziale**

Shell programs

During this lab you will learn:

* How to research what other features commands have and how to use them?
* How to create text file?
* How to make file executable, changing file permissions?
* What is “shell program”?
* How to run programs in LINUX?

Upload to blackboard:

1)Every Script file

2)Answered Questions from [Csc\_305\_lab1\_questions.docx](https://bbhosted.cuny.edu/bbcswebdav/pid-34378288-dt-content-rid-148478921_1/courses/CSI01_CSC_305_E001_1182_1/Csc_305_lab1_questions.docx)

3) A screenshot of the output from the scripts

Additional Linux Terminal Tips:

* In recent versions of Linux, to recall your recent commands while in the terminal, you can simple press the up arrow, this will retrieve the last command.
* Tab Autocomplete: To save time and to reduce typing mistakes when using a command or typing a file path, start typing the beginning of a command, you can hit the tab key to have linux autocomplete the rest of the command or file path, if it is unique the terminal will finish the rest of the command for you. If the letters you type are common between more than one command it will display all the possibilities for you to try.

*Tasks to try:*

Type:cd directory

Instead of completing the word directory hit the tab button (Note:replace “directory” with an actual directory name, or create a new directory)

**How to Research commands:**

On linux systems there are a multitude of commands that can do many different things from changing your directory (cd) to creating files (touch).There are too many commands for someone to explain every single one of them in detail, therefore most of the time the user/admin has to research what commands are available and what those commands do.

To find out what a commands are available to the terminal, there are a few different ways from the command line to research them.

1)”help” Command lists available commands on the system

2) Individual command help parameter

Any command that you want to see additional functionality use the --help parameter.

Example:

Command:touch --help  
Command:cd --help

Command:cat --help

Command:mkdir –help

* “—help” after the command name is known as a parameter,

Parameters are options you can enable for a command that affects how the command performs its job.

3)A way to get additional information about a command and its parameters is using “man”

“man” is a command that brings up a support document for a command and describes a command in further detail usually with examples, on how to use a particular command

Note:man only works if the man pages are installed for that particular command notls -l all commands have man pages

Example:

Command:man ls

Command:man cd

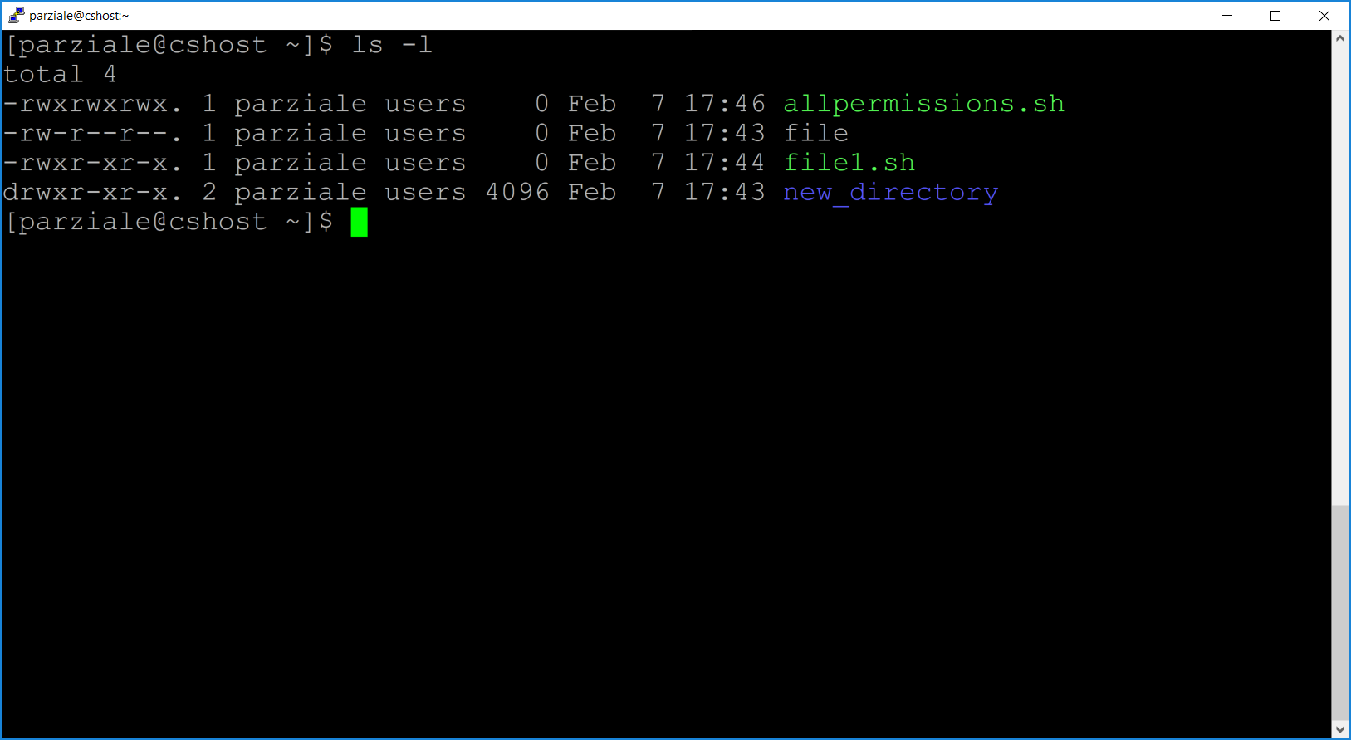
Command:man mkdir

**File and folder permissions:**

In the Linux file systems every file and folder has access permissions associated with it, these permission allow a user to Read write or Execute a file or folder in the file system.

In order to see the permissions associated with a file or folder you use the "ls -l" command

example output:



[parziale@cshost ~]$

The letters in front of the file represent the given permissions to each set of users,

The first 3(red) represent permissions for the User/owner of the file

The Next 3(blue) represent the group permissions of the file( i.e:the user group that file is a part of)

The last 3(green) represent permissions for any other user on the system.

example:

**-rwxrwxrwx** 1 parziale users 0 Feb 7 17:46 allpermissions.sh

The first dash represents if the object in question is a directory or a file, if it is a file the dash remains, otherwise a D is put in its place to represent that its a directory.

example:

drwxr-xr-x. 2 parziale users 4096 Feb 7 17:58 new\_directory

"r" Represents read persmision is active

"w" Represents write permisions is active

"x" Represents execute permission is active

if a dash is present in place of a letter it means the permission is not allowed for that group

**Changing file permissions**

To change file permissions we use the command "chmod"

example:

chmod a+w new\_directory

This command adds for all users and groups the write permission.

This can also be used to add read permissions "r" and execute permissions "x",

to remove a permissions, you would replace "a+w" with "a-w"

Interpreted Programing Language – A programing language that is run in, or sent to an interpreter which then reads the commands from the user and then executes system files and libraries to carry out the given functions. These rely on a Run time environment to execute.

Examples:

-bash

-shell

-Python

-Java Script

Compiled Programing Language – A programming language which needs to be converted into machine code first before it can be executed.

Examples:

-C

-C++

-C#

-COBOL

-BASIC

**Lab**

Download the file script\_file\_contents.docx from black board,

1) Experiment with scriptsby writing two series of three shell scripts called:

a) "a.sh", "b.sh", "c.sh",”d.sh”

run ./a.sh

b) "e.sh", "f.sh"

run ./e.sh

Write the scripts using the information in script\_file\_contents.docx

Note: You can use vim to write the scripts

2) In order to run the scripts add the execute permission to the following file only

a)a.sh

b)b.sh

c)c.sh

d)d.sh

e)f.sh

3) Run the scripts

a) To run the scripts simple treat them as a command

i) To run the first sequence:

**command:./**a.sh

ii) Run the second sequence run

**command:**./e.sh

iii) Run ./f.sh separate

**command:**./f.sh

b) Run the scripts using the “exec” command

i) **command:** exec ./a.sh

ii) **command:** exec ./d.sh

ii) **command:** exec ./f.sh

*note: use the command: “bash” before running “exec ./xxx.sh”*

*the bash command creates a new shell to work from,so when it finishes the new shell closes and you are left with the original shell prompt.*

*note: your connection to the server will close when script finishes,*

*you must log back in to the server afterwards*

4)Observe

a)Each shell from the first sequence will display its name, execute **ps** command, and then execute the next script in the sequence (ex: **exec b.sh** in a.sh),

Note:exec causes the ssh connection to terminate, this is by design

b)Each shell from the second sequence will display its name, execute **ps** command, and then execute the next script in the sequence by just calling its name (ex: **e.sh** in d.sh)

Observe the differences when you start:

 a.sh

 d.sh

 exec a.sh (note:Terminal may close)

Be ready to explain the differences in the execution of your shell sequences.

5) Answer the question in [Csc\_305\_lab1\_questions.docx](https://bbhosted.cuny.edu/bbcswebdav/pid-34378288-dt-content-rid-148478921_1/courses/CSI01_CSC_305_E001_1182_1/Csc_305_lab1_questions.docx) and upload it to blackboard

**Tentative Deadline:9/19/2022** Possible Points:10 points