**02 – Linux Command Line Interface (CLI)**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

Spring 2023

**Name:**

While Linux/Unix usually comes with a nice graphical user interface (GUI) that looks similar to the Mac or Windows desktops, it is very common to interact with Linux through a *command line interface* (CLI). This may be because you are accessing the Linux/Unix system remotely and only have a text-based interface, or it may be that using the Terminal is the most efficient or convenient way to accomplish a task you are working on. You gained some experience with the Linux CLI using the Terminal in the first set of activities. The activities for today will familiarize you more fully with the CLI, including how to specify file paths, how to use a number of commonly used Linux/Unix commands and how to learn about others.

**Before We Start:**

1. Start the comp190 docker container that you created in the first homework. Recall that to do this:

* Run the Docker Desktop application.
* Go to the “Containers” Tab
* Start the comp190 container (click its “play button”)
	+ If you do not see a comp190 container, revisit HW01 #9-12 to create and run it.
* Run the Tiger VNC Viewer application.
* Connect to localhost:5901
* Open a Terminal window.

Nothing is required here. However, if you have not been able to get the comp190 container running and connect to it via VNC you must see your instructor as soon as possible. All of the activities today (and essentially the rest of the semester) require that you have this working.

**Path Preliminaries:**

2. Identify each of the following paths as a relative path or an absolute path. Don’t worry about what they refer to for now. The point here is to be able to recognize paths as relative or absolute when you see them.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Path** | **Path Type** |  |
|  | /usr/bin/sort |  |  |
|  | home/comp190 |  |  |
|  | Notes.txt |  |  |
|  | /home/comp190/Notes |  |  |
|  | .. |  |  |
|  | ../Notes |  |  |
|  | ./MyFile.docx |  |  |
|  | ~ |  |  |
|  | / |  |  |
|  | /home/comp190/../knuthd |  |  |
|  |  |  |  |

**Who Are You?**

Like other operating systems you have used, you have a username on a Linux/Unix system. Normally, each user on a system will have a different username. On the system we are using for this class, the username of the user that is logged in when you connect to it is student. There may however be circumstances when it is useful to be able to ask the system who you are (i.e. the username that is logged in).

3. The whoami command displays the name of the logged in user. Run this command and paste a screenshot of the command and its output below.

**Where is Your Home?**

Every user on a Linux/Unix machine has a *home directory*. You worked in your home directory in HW 01, making files and directories and moving them around. Your home directory is just part of the larger collection of directories and files that exists on the machine. In this section, you’ll learn where your home directory is located.

4. The pwd command displays the *absolute path* of the working directory. Run this command and paste a screenshot of the command and its output below.

5. Based on the output you see in your answer to question #4

a. What is the absolute path to your home directory? Note: Be sure to distinguish between “your home directory” and “the /home directory”. This question and the next are asking about “your home directory.”

b. As mentioned, your username on this system is student. If your username on this machine were the same as your school username, what do you think the absolute path to your home directory would be?

**Getting to the Root of Things:**

Your home directory is one of the significant places in a Linux/Unix system. The *root directory* is another. The *Root Directory* of a Linux/Unix machine is like the file cabinet in an office. It contains all of the other files and directories that exist on the machine.

6. The command cd / changes the working directory to be the root directory. Run this command, use the pwd command to confirm you are in the root directory, and then display the files and directories that are in the root directory. Paste a screenshot showing the commands you used and their output.

**Getting Back Home:**

In question #5 you saw that the absolute path to your home directory is /home/student. That path tells you that the root directory /, contains a directory named home (you should see it in your answer to #6). That home directory then in turn contains a directory named student. That student directory is your home directory.

7. Use the cd / command to make the root directory the working directory, if it is not already. Then use the cd <dir> command to get back to your home directory. Finally, use the pwd command to confirm that your home directory is now the working directory. Paste a screenshot showing the commands you used and the output that shows you are now in your home directory.

As you just saw, you can always get back to you home directory from the root directory by changing to home and then to student. However, getting back to your home directory is something that you’ll do frequently so there are some shortcuts that will let you get home from anywhere. The two most common ways are:

|  |  |
| --- | --- |
| **Command** | **Description** |
| cd | When no <dir> is specified, the working directory is changed to your home directory. |
| cd ~ | Using the shortcut ~ as the <dir> changes the working directory to your home directory. |

These commands can be particularly useful when you are first learning to use the CLI. If you are moving around and lose track of where you are, these commands will magically transport you back home (like Dorothy’s ruby slippers[[1]](#footnote-1)).

8. Change to the root directory, use one of the above commands to get back to your home directory and then confirm that your home directory is the working directory. Paste a screen shot of the commands you used and their output below.

**Getting Setup:**

To complete the rest of today’s exercises you will need the directories and files that you created in HW 01. If you successfully completed HW 01, tour home directory should contain the following directories and files:



9. If your directories and files do not appear as shown here, please create them. You may use the CLI or GUI interface to create them. Recall that each of the .txt files should contain a list of at least 5 of your favorites (foods, movies, places), with one per line.

There is nothing required for this question. Just ensure that your directory and file structure looks as shown above.

10. Use the cd and ls commands to confirm that you have these directories exist and also that each one contains the appropriate file. Paste a screenshot of the commands that you used and their output.

**Using Relative and Absolute Paths:**

As you have seen, when using the cd <dir> and mkdir <dir> and mv <file> <dir> commands you can replace the <dir> and <file> arguments with the name of the directory or the file on which to operate. These commands, and most other Linux/Unix commands will also allow you to use relative or absolute paths in place of the <dir> and <file> parameters. This flexibility helps to simplify multi-step or more complex operations.

For example, imagine you wanted to make a directory Genres inside of the Movies directory. One approach would be to change into the Movies directory (cd Movies) and then make the Genres directory (mkdir Genres). But you can accomplish the same thing in one step using an absolute path:

 mkdir /home/student/Movies/Genres

or, assuming your home directory (/home/student) is the working directory, with a relative path:

 mkdir Movies/Genres

11. Use one of the above commands to create the Genres directory within your Movies directory. Use other commands to confirm that the Genres directory was created (hint change directories and use ls). Give a screenshot showing the commands you used and the output.

12. Make your home directory the working directory. Then complete the following tasks:

a. Use a single mkdir <dir> command with an absolute path to create a Comedy directory inside of your Genres directory. Use other commands to confirm that the Comedy directory was created. Give a screenshot showing the commands you used and the output.

b. Use a single mkdir <dir> command with a relative path to create a Drama directory inside of your Genres directory. Use other commands to confirm that the Drama directory was created. Give a screenshot showing the commands you used and the output.

13. Now make your Foods directory the working directory. Then complete the following tasks:

a. Use a single mkdir <dir> command with an absolute path to create a Thriller directory inside of your Movies/Genres directory. Use other commands to confirm that the Thriller directory was created. Give a screenshot showing the commands you used and the output.

b. Use a single mkdir <dir> command with a relative path to create a Horror directory inside of your Movies/Genres directory. Hint: You will need to use .. in your path. Use other commands to confirm that the Horror directory was created. Give a screenshot showing the commands you used and the output.

14. The cd <dir> command can also be used with both absolute and relative paths (including ..). Using these types of paths will allow you to use cd to change the working directory more flexibly.

Give a cd <dir> command that accomplishes each of the following tasks using the type of path specified. Use the cd and pwd commands in the terminal to test your answers. Screenshots are not required for this question.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Command** | **Task** |  |
|  |  | With your home directory as the working directory change to the Horror directory using a relative path. |  |
|  |  | with Horror as the working directory change to the Movies directory using a relative path (hint: ..). |  |
|  |  | With Movies as the working directory to the Foods directory using an absolute path. |  |
|  |  | With Foods as the working directory change to the Places directory using a relative path (hint: ..). |  |
|  |  | With Places as the working directory change to the Genres directory using a path beginning with ~. |  |
|  |  |  |  |

**Displaying a File:**

Sometimes (like in the exercises coming below ;) you will you want to see the contents of a file but do not want or need to edit it. So rather than opening it in an editor, there are several Linux/Unix commands that can be useful.

15. One such command is cat <file>. One of the things cat can do is simply display the contents of a file on the screen. Use the cat command to display the contents of your my-foods.txt file. Paste a screenshot of the command that you use and its output here.

**Copying Files:**

The cp <file> <dest> command makes a copy of the <file> file into the <dest>. Where <dest> may be a file or a directory.

For example, if the file my-notes.txt is in the working directory, then the command:

 cp my-notes.txt my-notes-backup.txt

would make a copy of the file my-notes.txt with the name my-notes-backup.txt.

16. Use the cp command to make a copy of your my-movies.txt file as

my-drama-movies.txt. Check that the new file contains a copy of the original by using the cat command. Paste a screenshot of the commands that you use and the output here.

As you probably expect, the cp command also allows the use of relative and absolute paths for the <src> and the <dest>. For example, if you want to copy the my-notes.txt file into a directory named Backups in the working directory you would use the command:

 cp my-notes.txt Backups

If you want to give the copy in Backups a different name, then you can include that name in the path, for example:

 cp my-notes.txt Backups/my-notes-backup.txt

17. Make the Movies directory your working directory. Then use the cp command to copy the my-movies.txt to the locations specified in the table below. Use the cat command to verify that your commands worked. Screenshots are not required for this question. However, you will need the copied files for later exercises, so be sure they have worked correctly.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Command** | **Location** |  |
|  |  | Into the Horror directory. |  |
|  |  | Into the Drama directory. |  |
|  |  | Into the Drama directory with the name my-dramas.txt.  |  |
|  |  | Into the Foods directory. Hint: use a .. in the path. |  |
|  |  |  |  |

**Deleting Files and Directories:**

The rm <file> and rmdir <dir> commands are used in Linux/Unix to delete files and directories.

18. Use the rm <file> command to delete the my-movies.txt file from the Horror directory. Use the ls command to confirm that the file has been deleted. Paste a screenshot of the commands that you use and the output here.

19. I don’t like Horror movies, so use the rmdir <dir> command to delete the Horror directory. Use commands to confirm that the directory has been deleted. Paste a screenshot of the commands that you use and the output here.

20. If you completed the prior exercises, your Drama directory should contain two files: my-movies.txt and my-drama-movies.txt. Ensure that it does and create them if not. Then, try to delete the Drama directory using rm. You should receive an error message. Paste a screenshot of the command that you used and the error message here.

**Operating on Multiple Files (Wildcards):**

In question #20 you saw that rmdir will not delete a directory unless it is empty. Thus, before that directory can be deleted the files inside it will need to be deleted. There are two files in the Drama directory. You could delete them one by one, but this is a good chance to use the *wildcard character (\*)*.

The wildcard character (\*) is used to write paths that refer to multiple files or directories. When a command is used with a path containing a wildcard it operates on all files or directories represented (or we often say *matched*) by that path. For example, the path:

 my-\*.txt

matches the following filenames (and lots more):

 my-movies.txt

my-dramas.txt

In fact, it will match any filename that starts with my- and ends with .txt. Another way of saying this, is that the wildcard can replace any number of characters at the location it appears.

As another example, the path:

 Movies/\*.txt

will match any file with a name ending in .txt that appears in the Movies directory.

One final example, the path:

 Movies/\*/my-\*

will match any file in any sub-directory of Movies (e.g. Drama, Comedy, etc.) that begins with my-.

21. Make the Movies directory your working directory, and then use a single rm command with a wildcard to delete both of the files in this directory. Then also delete the Drama directory. Use other commands to confirm that everything as worked. Then paste a screenshot of the commands that you used and their output here.

22. Wildcards can be used in paths with most Linux/Unix commands (e.g. cp, rm, rmdir, mv, etc…) to operate on multiple files or directories at a time. For each set of files shown in the table below, write an rm command using a path with a wildcard that will delete the “Files to Delete” but not the “Files not to Delete.” Screenshots are not required for this question.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Files to Delete** | **Files not to Delete** | **rm with Wildcard(s)** |  |
|  | Quan-food.txtQuan-places.txtQuan-movies.txt | Jones-food.txt | Quan-\*.txt |  |
|  | Joy-Spanish-food.txtXia-Spanish-movies.txtJuan-Spanish-places.txt | Emil-French-music.txt |  |  |
|  | Movies.datFood.datPlaces.dat | Movies.txtFood.txtPlaces.txt |  |  |
|  | Buddy-music.txtBB-music.txtMuddy-music.txt | Buddy-Food.txt |  |  |
|  | Movies/Drama/Mov1.txtMovies/Drama/Mov2.txtMovies/Comedy/Mov3.txt | Movies/Comedy/Mov4.datFoods/Spanish/Paella.txt |  |  |
|  |  |  |  |  |

**Linux/Unix Manual Pages:**

You have now seen a number of Linux/Unix commands, but there are many many more. Instead of reviewing them one-by-one, this section introduces you to the *Linux/Unix Manual Pages*, often called “the man pages” for short and because the command (man) is used to access them. The man pages are documentation that is available on most Linux/Unix systems and provide a reference to the Linux/Unix commands on that system.

A man page for a command is accessed using man <cmd>, where <cmd> is the command for which you would like the reference. When the man page for a command is displayed, you can use the “Space Bar” to show the next page of information, or the “q” key to quit before you get to the end.

23. Use man to display the manual for the head command. Use the documentation that is shown to answer these questions about the head command. What does the head command do? How many lines does head display by default?

Each man page gives a “SYNOPSIS” for the command. This synopsis shows the general format for the command. They use a syntax similar to what we have been using, but they use all CAPITAL letters to indicate values to be replaced instead of the < > as we have been doing. In addition, they use a “…” to indicate that a value may be repeated multiple time.

24. What is the general format (i.e. synopsis) for the head command?

**Command Options (i.e. Flags):**

Most Linux/Unix commands will have *Options* (also often called *Flags*) that change their behavior. The SYNOPSIS on the man page shows that any flags to be used are listed following the command itself. The “DESCRIPTION” section of the man page lists all of the flags that can be used and explains what each one does.

Most flags will have two forms. A *short form* with a single dash and a single letter (e.g. -n) or a *long form* two dashes and a longer name (e.g. --number). The short and long forms are equivalent and either may be used.

25. Consult the man page for the head command.

 a. What does the -q flag do? What are its long forms (it has two)?

 b. What does the --lines flag do? What is its short form?

**Hidden Files:**

In Unix/Linux filenames beginning with . are called hidden files. For example, a directory named .cache for a file name .profile will be hidden files. Hidden files are often used to save configuration information or other data that users do not need to see or interact with regularly. For this reason, the ls command does not display hidden files by default. However, with the right flag we can ask it to display them for us.

26. Use the man command to find the flag that causes ls to include hidden files in its listing. With your home directory as the working directory, use the ls command first without and then with this flag. Give a screenshot of the commands that you used and their output.

**Command Flag Arguments:**

Some flags, like the --lines flag you just saw for the head command will require an argument. When this is the case, the description of the flag will include an “=”. For example,

--lines=[-]NUM

indicates that when the --lines flag is used, it must be followed by an equal sign and then number, possibly preceded by a minus sign.

The short form for flags can also be used to provide arguments. When using the short form of a flag, the argument is placed immediately following the flag, without an equal sign. For example:

-nNUM

27. Use a head command to display the specified number of lines from your my-foods.txt file.

a. Use the long form of the flag to display the first 4 lines. Paste a screenshot of the command that you use and its output here.

b. Use the short form of the flag to display the first 3 lines. Paste a screenshot of the command that you use and its output here.

It is important to know about the man pages and how to read them because they provide the definitive reference for how each commands behaves. That said, when you just want to remember how to use a command, it can often be more effective to just do a web search. For example, a search for “Unix display first 3 lines” gives a page describing the head command with an easy-to-follow example.

**Linux/Unix Terminal Text Editors**

Linux/Unix systems are often accessed remotely and using only a text based CLI (i.e. no GUI desktop). Because of this, most Linux/Unix systems provide some basic text editor programs that can be run within the Terminal. Many people also find these editors useful for quick tasks even when the GUI is available. One of the most popular Terminal-based editors is nano.

28. Use the command nano to open the Nano text editor in the Terminal. Type a sentence or two about something into the editor. Paste a screenshot of the editor with your text here.

The commands for Nano are listed at the bottom of the screen. The “^” before the commands indicate that you use the Ctrl (or Control) key. For example, to exit Nano use press “Ctrl+X”.

29. Save what you have typed into a file using the “Write out” command and then exit Nano. Use cat to display your file. Paste a screenshot of your cat command and its output here.

Optional: Another terminal-based Linux/Unix editor is vi. It is difficult to learn but allows you to complete every possible action using keyboard commands. This makes using vi very fast and efficient once you know how to use it well. For this reason, vi is a favorite of a lot of hard-core Linux/Unix folks. It’s also worth knowing about because vi is the only editor you’ll find on some Linux/Unix systems. If you are interested, you can read more about vi and how to use it on the following site:

* <https://www.redhat.com/sysadmin/introduction-vi-editor>

**Summary:**

30. Complete the table below by filling in the Linux command that corresponds to each task.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Command** | **Task** |  |
|  |  | Change to your home directory  |  |
|  |  | Change to the root directory |  |
|  |  | Change to the parent of the current directory |  |
|  |  | Display the username of the current user |  |
|  |  | Display the path of the working directory |  |
|  |  | Display the contents of a file |  |
|  |  | Copy a file |  |
|  |  | Display the first lines of a file |  |
|  |  | Display directory contents including hidden files |  |
|  |  | Delete a file |  |
|  |  | Delete an empty directory |  |
|  |  | Start the nano editor in the terminal |  |
|  |  |  |  |

**Optional:** To help us improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.

**Acknowledgements:**

Some materials, questions and resources have been adapted from activities posted on foss2serve.org.

* [http://foss2serve.org/index.php/Intro\_to\_Bash\_(Activity)](http://foss2serve.org/index.php/Intro_to_Bash_%28Activity%29)
* <http://foss2serve.org/index.php/Linux_Beginner_Activity>
1. Cultural reference to the Wizard of Oz: <https://www.youtube.com/watch?v=n5PQIXpS0R8> [↑](#footnote-ref-1)