Text

Description automatically generatedCSC 235

Introduction to Linux & Linux Administration

Fall 2025

**Assignment #1: Introduction to Linux**

**Assigned: Tuesday, August 26, 2025**

**Due: Sunday, August 31, 2025 (NLT 23:59)**

This lab introduces you to some of the basics of using the Linux operating system. Specifically, the (**man**)ual and (**info**)rmation commands.

**Instructions:**

1. Insert your answers into this document (*text and screenshots*)
2. Highlight your answers in green (*text only*)
3. When finished, rename the document (*please use the naming convention below*)
4. Upload the renamed document to the Moodle assignment

**A1 - Last Name** (e.g., **A1 - Nonnweiler**)

# Section I. Getting Started

**Shell Prompt**

**jnonn@ubuntu:~ $**

When you type, the characters appear after the dollar sign ($ e.g., ls, cd, etc.). All the stuff prior to what you type is called the ***shell prompt****.* This may differ across environments, distributions, or operating systems (FreeBSD or Debian Linux, for example), but can be quickly identified by the following characteristics:

* + Name before @ symbol is the user currently logged in. In this case, **jnonn** is logged in.
  + The machine name is indicated by the string after the @ symbol, but before the colon. In this example, we are logged in to **ubuntu**.
  + We then have our current working directory: **~** (home).
  + Finally, we have the **$** sign. This could be either a **#**, **%**, or **$** symbol, depending on the OS.

Given all that, please fill in the answers to the following questions. Take a screenshot and copy/paste it below (please size your image to be readable!), then answer each question.

1.1. Screenshot of your terminal window:

A screenshot of a computer

AI-generated content may be incorrect.

1.2. What is the user id shown in the shell prompt?

Answer: xnguyen

1.3. What is the name of the machine you are currently on?

Answer: Ubuntu

1.4. What is the symbol located at the end of the shell prompt?

Answer: $

Man

# Section II. Using the man (manual) command

The **man** command is used to view the Linux online documentation about a specific command, program, or application. It is often referred to as the “**man pages**” or manual pages. Type the following information:

**man ls**

2.1. **man ls** Screenshot:

A screenshot of a computer

AI-generated content may be incorrect.

2.2. What does the **man** command tell you about the **ls** command (*i.e., what does the ls command do*)?

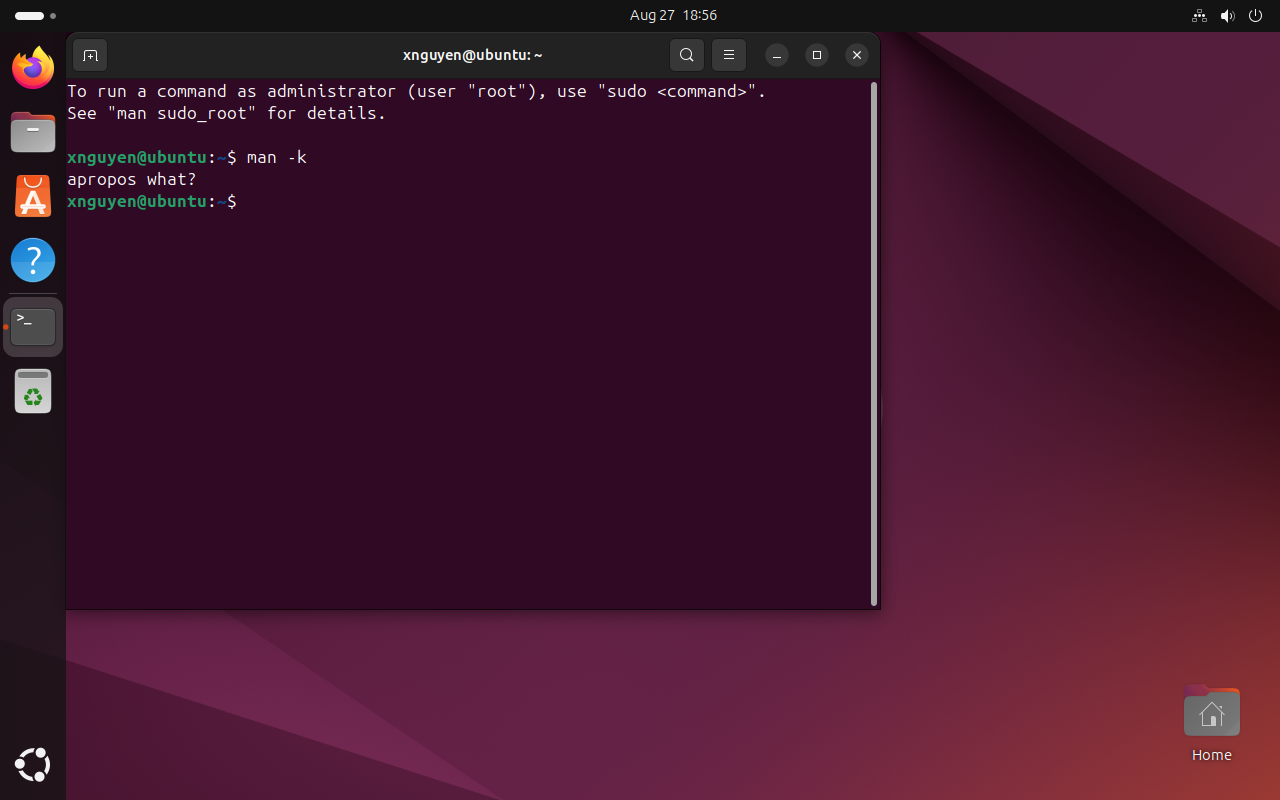
Answer: List directory contents

**But, what if you don’t know the name of the command you which to explore?**

If you know the name of the command (say, **tail**), you simply type it following **man** (i.e., **man tail**). If you don’t know the name of a command but know a keyword that might be part of the command's description, use the **-k** switch. Thus, if you wanted to know about commands that are associated with printing, you might say **man -k print** or **man -k print | more** (*if you don't want things scrolling off the screen*). The **man -k** command is also known as **apropos** and can be called using the **apropos** command (e.g., **apropos print**); the two are identical. Use whichever version you find easier to remember.

**man** is also used to tell programmers about system functions and data formats that can only be accessed in a meaningful way by writing a program. If you only want to find out about command-line commands, you can look in Section 1 of the man pages. For now, using the **-k** switch will be useful in identifying “words.” The returned entries will contain a number in parentheses “()” following each command. Therefore, commands that end in (1) are command-line commands.

2.3. Screenshot after the **man -k** command



2.4. Name three command-line commands that have “print” in the description:

Answer: zstdgrep, xvinfo, xkbprint

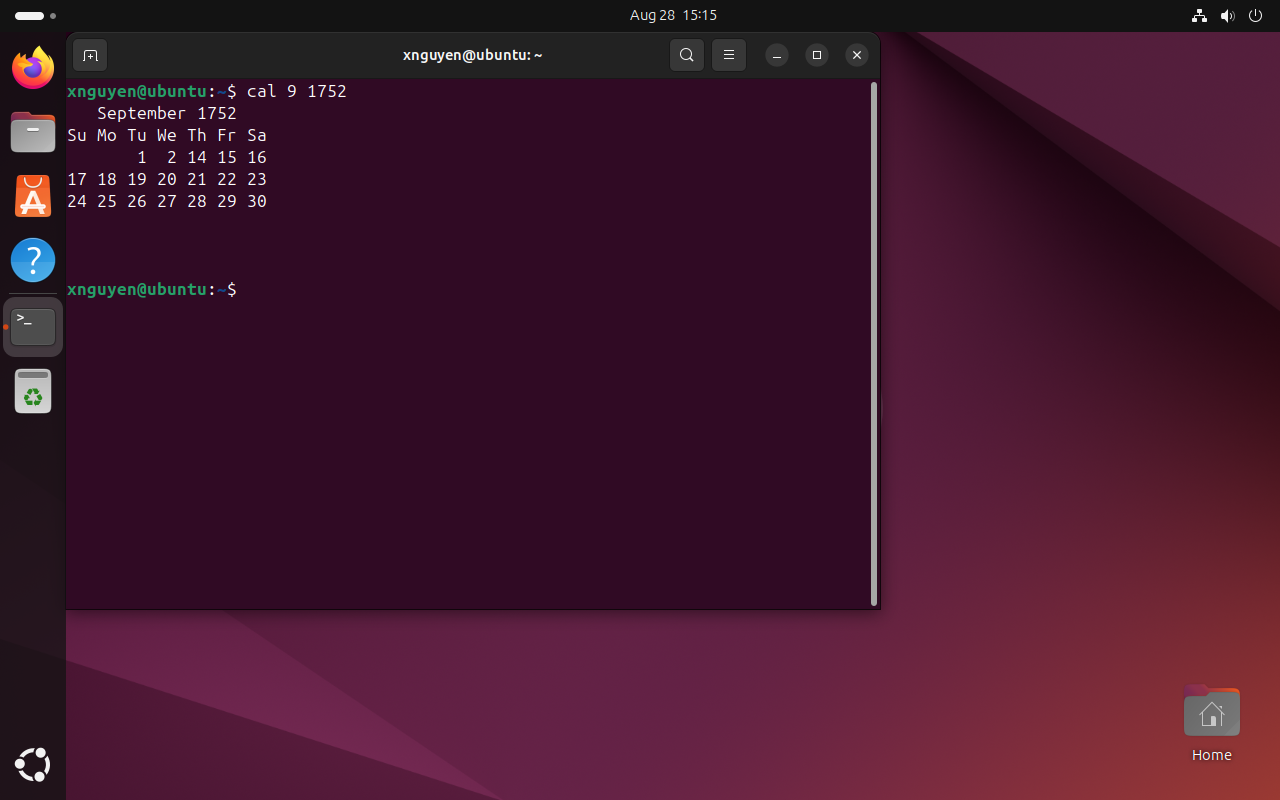
Locate and execute commands to perform each of the following. Use **man -k** (i.e., **su**) if necessary. You may have to install the calendar. If the commands give you an error message, you may need to install **calendar** (note, I ended up installing/using ncal. Remember, there will be multiple ways of accomplishing tasks within Linux!). Here are the commands to install:

**sudo apt update**

**sudo apt install gnome-calendar**

2.5. In one window, print to the screen the calendar for September 1752.

Screenshot:

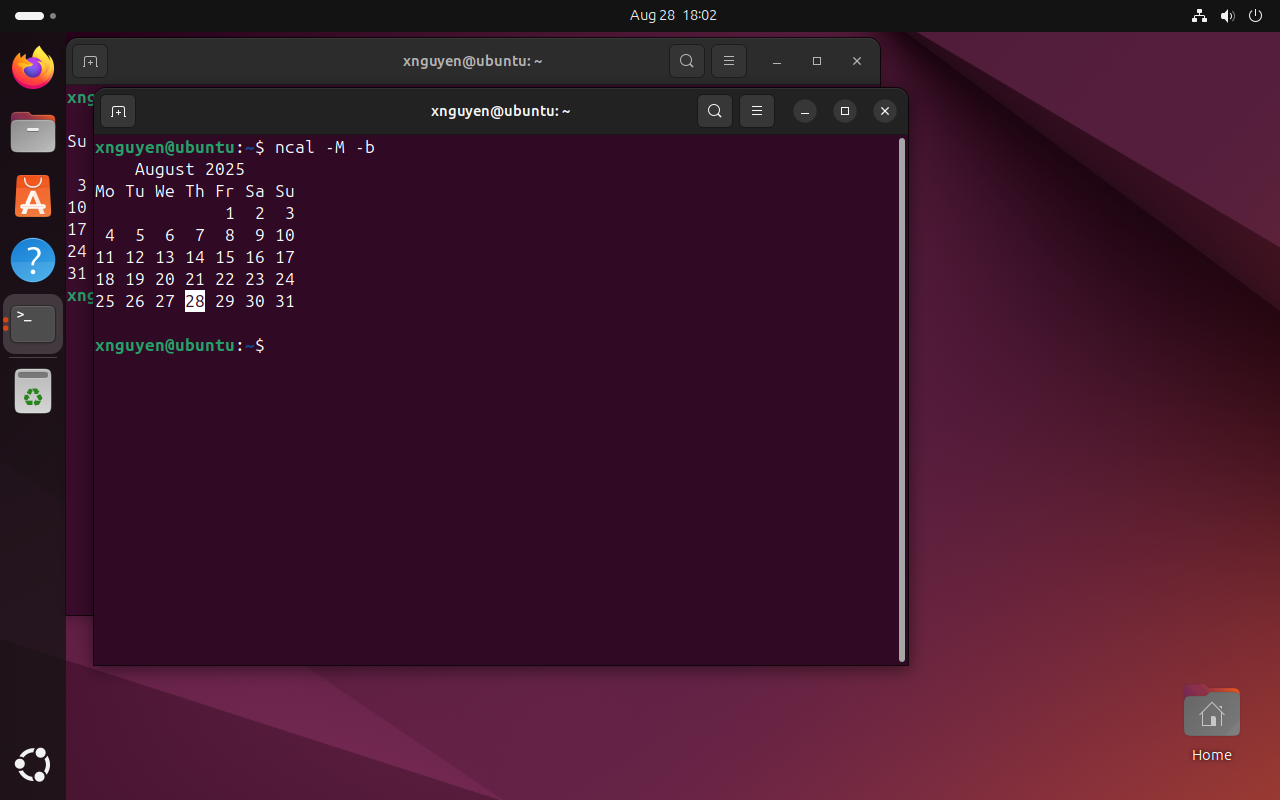


2.6. What is strange about this month?

Answer: The 3rd through 13th days are missing.

2.7. In a second window, display the current month, starting with Monday as the first day of the week. (hint: **ncal**)

Screenshot:



**Section III. Navigation**

Besides knowing how to research commands, you should know how to navigate the file system on the Unix system. We will cover the following commands: **ls**, **cd**, and **pwd*.***

* **pwd** present working directory
* **cd**  change directory
* **ls** list directory contents

3.1 What is the difference between **ls**, **ls -a**, and **ls -al**?

Answer: ls only shows non-hidden files and directories, ls –a shows both non-hidden and hidden files and directories, and ls –al shows both non-hidden and hidden files and directories as well as extended information about them.

**pwd – Present Working Directory**

The directory you are standing in is called the *working directory* (or *present working directory*). To find the name of the working directory, use the **pwd**command. When you first log on to a Unix/Linux system, the working directory is set to your *home directory*. On most systems, your home directory will be called /**home/your username**, but it can be anything according to the whims of the system administrator. Or it is the *root* user’s directory **(/root**).

3.2. What prints to the screen when you enter **pwd**?

Answer: /home/xnguyen

3.3. To list the files in the working directory, use the **ls** command. Name three things that print to the screen when you enter **ls**. (if you want, you may change directories to select something “more interesting”)

Answer: Desktop, Documents, Downloads

**cd – Change Directory**

To change your working directory (where you currently are in the tree structure) you use the **cd** (change directory) command. To do this, type **cd** followed by the **pathname**of the desired working directory. A pathname is the route you take along the branches of the tree to get to the directory you want. Pathnames can be specified in one of two different ways; *absolute pathnames* or *relative pathnames*. Let's look at absolute pathnames first.

* An **absolute pathname** begins with the root directory (**/**) and follows the tree branch by branch until the path to the desired directory or file is completed. For example, there is a directory on your system in which most programs are installed. The pathname of the directory is **/usr/bin**. This means from the root directory (represented by the leading slash in the pathname) there is a directory called "**usr**" which contains a directory called "**bin**".

Let's try this out. In a window, print to the screen the contents of the **/usr/bin** directory

Now we can see that we have changed the current working directory to **/usr/bin** and that it is full of files. Notice how your prompt has changed? As a convenience, it is usually set up to display the name of the working directory.

* Where an absolute pathname starts from the root directory and leads to its destination, a **relative pathname** starts from the working directory. To do this, it uses a couple of special notations to represent relative positions in the file system tree. These special notations are "**.**" (dot) and "**..**" (dot dot).

The "**.**" notation refers to the working directory itself and the "**..**" notation refers to the working directory's parent directory. Here is how it works. Let's say that we wanted to change the working directory to the parent of /usr/bin which is /usr. We could do that two different ways.

**cd /usr/bin** ***(absolute path)***

**cd ..** ***(relative path)***

Two different methods with identical results. Which one should you use? The one that requires the least typing! Likewise, we can change the working directory from /usr to /usr/bin in two different ways. First, using an absolute pathname:

**cd /usr/bin**

**pwd**

Or, with a relative pathname:

**cd /usr/** ***(absolute path)***

**cd bin** ***(relative path)***

**pwd**

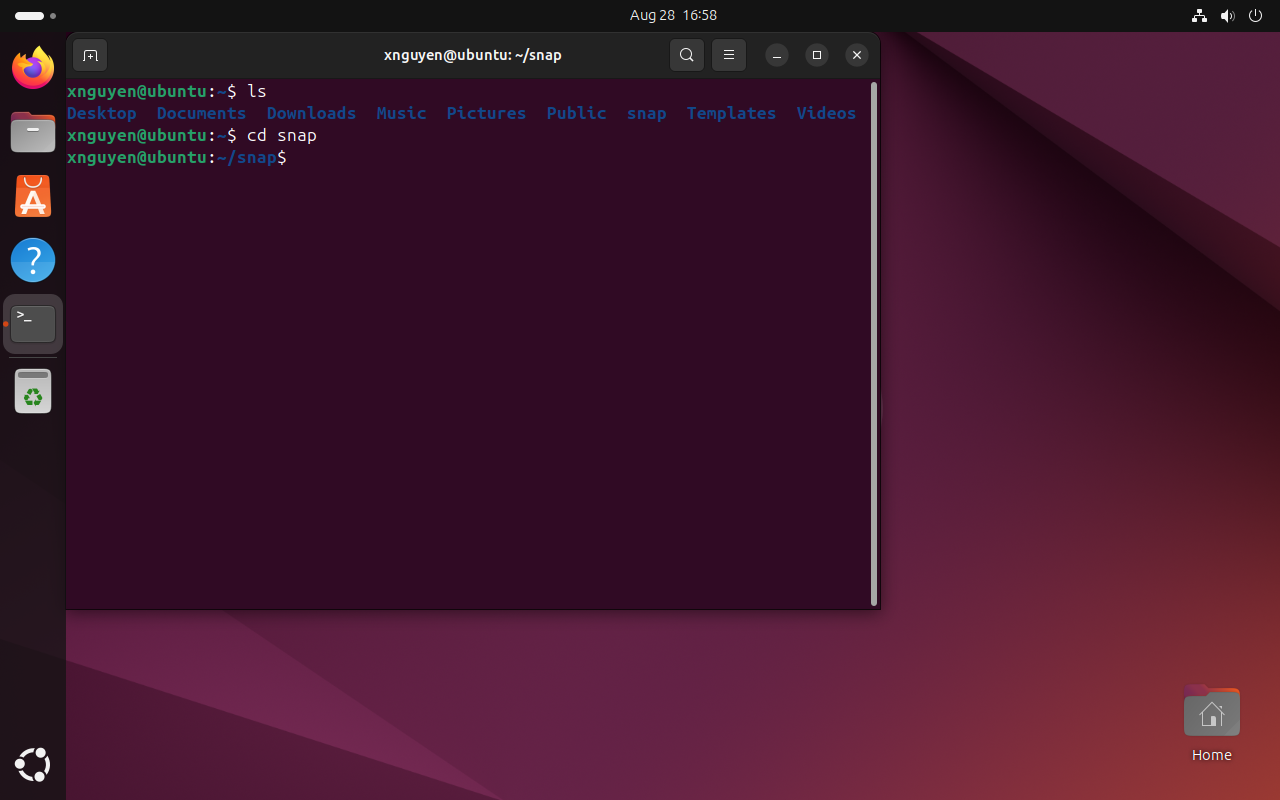
There are several other shortcuts with the cd command. You can issue cd itself on the command line to return to the user’s home directory:

**cd**

**pwd**

3.4. Change your present working directory to any directory other than your home (your choice!) Take a screenshot.

Screenshot:



3.5. List three objects in that directory

Answer: firefox, firmware-updater, snapd-dektop-integration

**Section IV. VI** (“vee-eye”) **or VIM** (“vim” meaning vi improved)

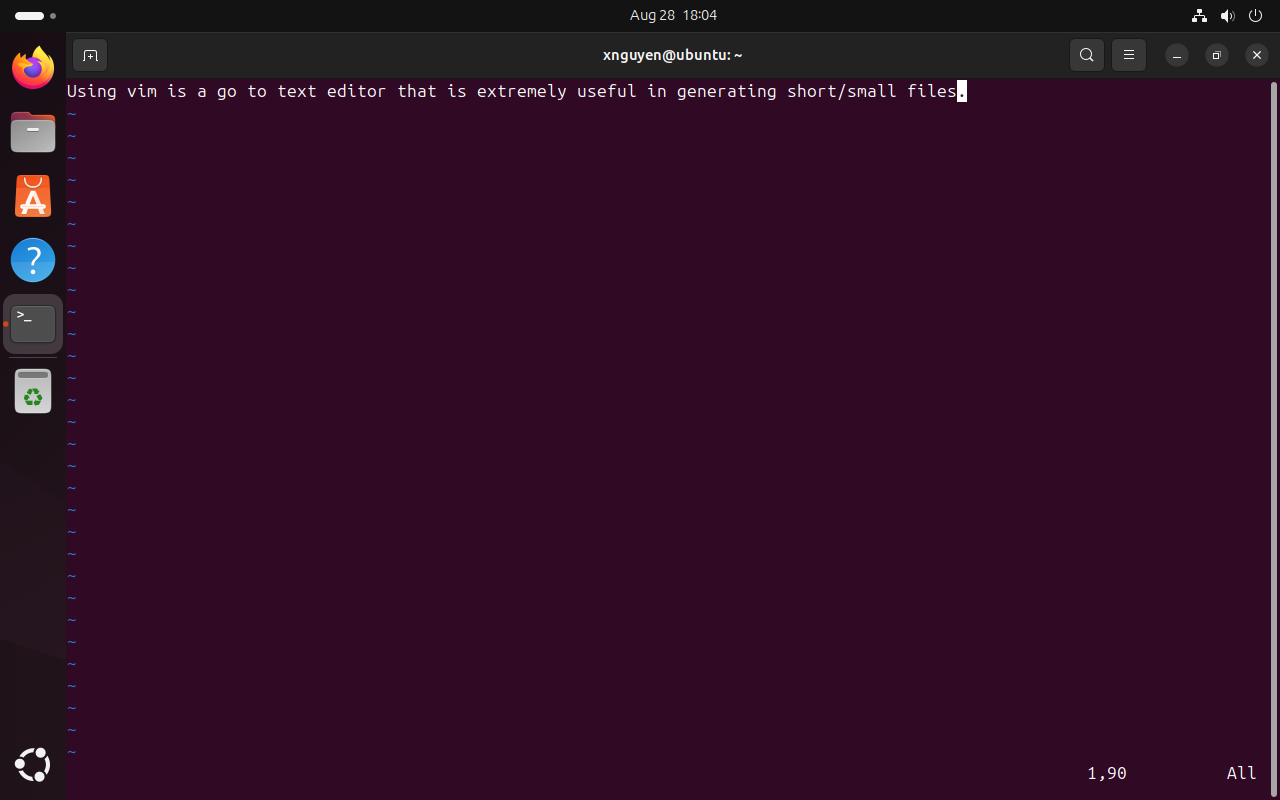
vi is a visual text editor for Unix, and it’s the default editor for Unix systems. Since most telnet interfaces start off in a Unix shell, you can open and edit files without having to send them to your desktop and then back to the machine where they reside.

There are several things vi has going for it. Since it’s the default on most Unix systems, it’s ubiquitous. It’s also small, fast, extremely powerful, and keyboard-based. If you’re working with a slow connection or on a portable machine with no mouse and very little system memory, you’ll still be able to create and edit files with ease.

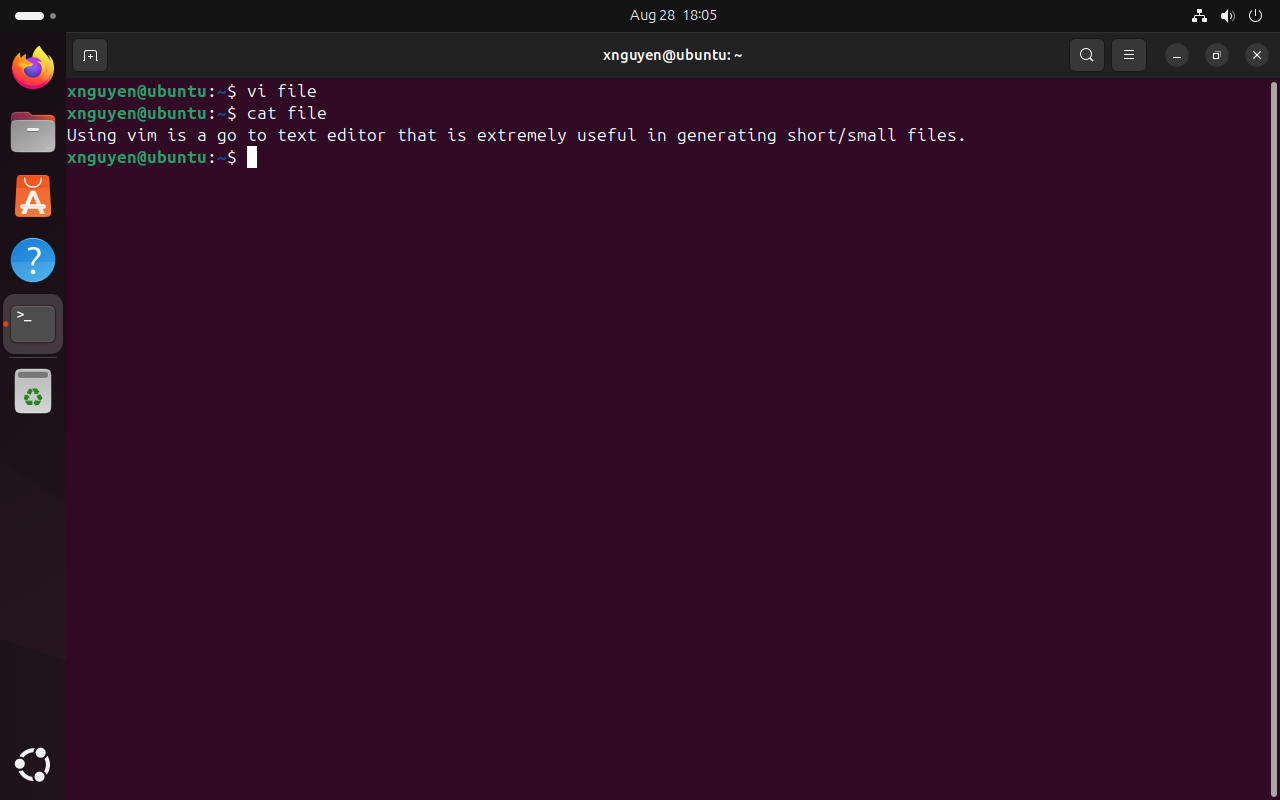
Visit w3schools, review TestOut Section 2.3 Text Editors, and/or any vi/vim cheat sheets.

Open vi/vim creating a new file. Write the following message (or any other appropriate message). “Using vim is a go to text editor that is extremely useful in generating short/small files.” Before exiting the file, take a screenshot. Then use the **cat** command to display the contents of your newly created file. Take a second screenshot of the output of the **cat** command.

4.1. Screenshot (from within the vim editor):



4.2. Screenshot (after the **cat** command):



You should be able to…

* append, delete, move around a document, and insert text
* perform additional deletion commands and undo actions
* perform copy and paste operations

**Section V. Linux File Commands**

Linux has a number of commands for manipulating files and directories (folders).

The **~** character by itself refers to your own home directory, so if you are deep in your directory structure, you may access a file named **myFile.txt**, for example, in your home directory using **~/myFile.txt**.

While in your home directory, accomplish the following tasks:

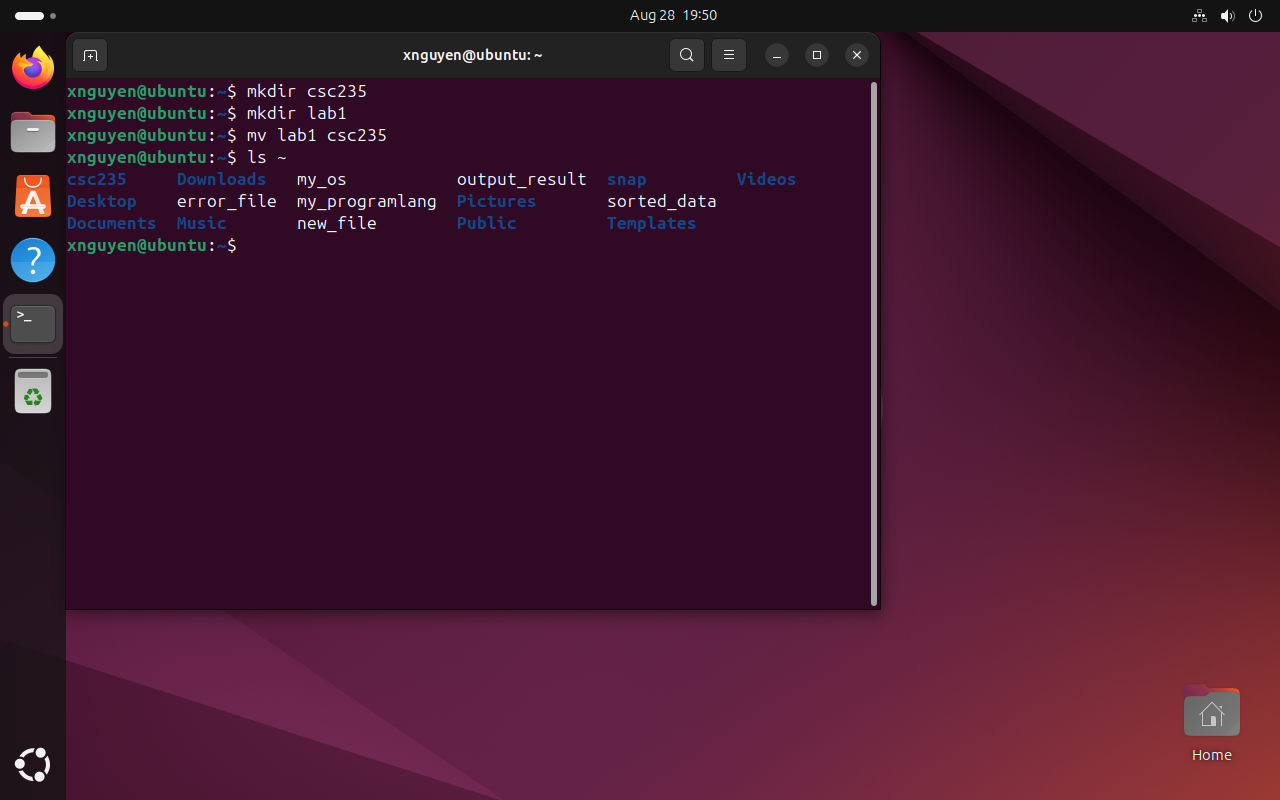
**mkdir csc235** (makes a new directory)

**mkdir lab1** (makes a new directory)

**mv lab1 csc235**  (moves files and directories)

Using the ls command, capture a screenshot that shows the contents of a.) **~** and b.) **~/csc235**.

5.1 Screenshot a.):



5.2 Screenshot b.):

