Text

Description automatically generatedCSC 235, Introduction to Linux &

Linux Administration

Fall 2023

**Assignment #1: Introduction to Linux**

**Due: Sunday, September 3, 2023 (NLT 23:59)**

This lab introduces you to some of the basics of using the Linux operating system[[1]](#endnote-1). 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, first name.docx** (e.g., **A1 - Nonnweiler, Joel**)

# Section I. What is the shell?

Info

Before we jump into the ***man*** command, let’s talk about the shell. When someone talks about the command line, this is really the *shell*. The *shell* is a basic computer program that takes user input via a keyboard and passes it on to the operating system. The most common *shell* is referred to as ***bash*** (Bourne Again Shell). It is an evolution of the original shell, ***sh.***

Understanding what all the text indicates before the OS symbol ($, #, or %).

Example: jnonnweiler@ubuntu22:~ $

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, **jnonnweiler** 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 **ubuntu22**.
  + We then have our current working directory: **~** (home).
  + Finally, we have the **$** sign. This could be either a **#**, **%**, or **$** symbol, depending on the OS.

**Section I.a** 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.

Screenshot:

A screenshot of a computer

Description automatically generated

Q. Who is the user identified in your shell prompt?

Answer:

*vboxuser*

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

Answer:

Ubuntu-22

Q. Which symbol is at the end of your prompt?

Answer:

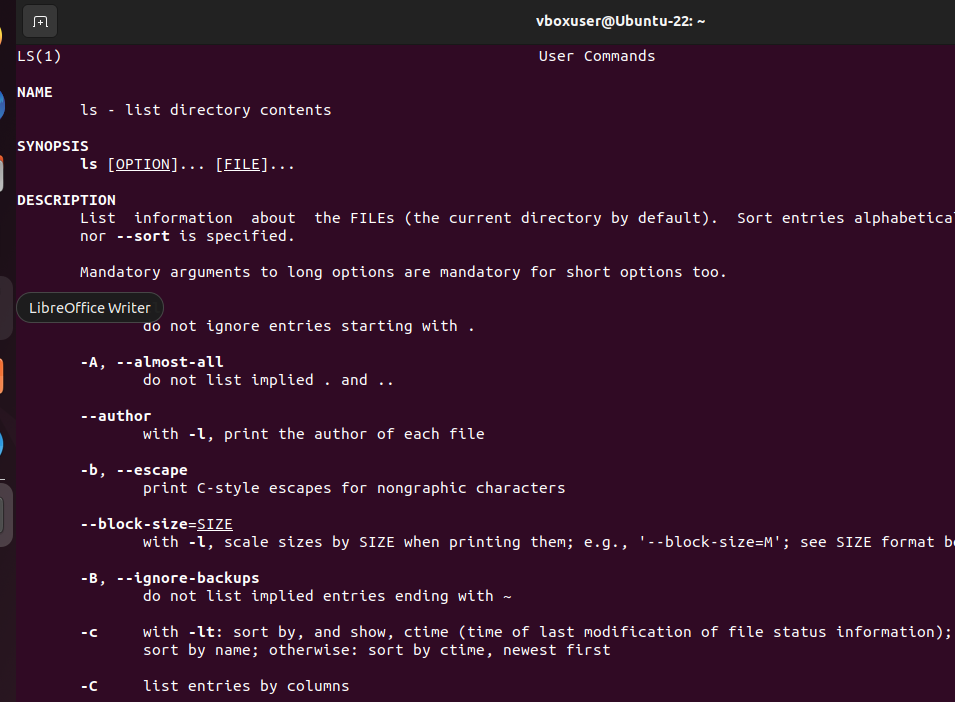
$

# Section II. Using man (manual) command

**Section II.a.** 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

Screenshot:



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

Answer:

The man command tells you the name, synopsis, description, author, bugs, copyright, and see also.

The ls command lists directory contents.

When you give a man command, and there is too much information to fit on one screen, it gives you one screenful and pauses until you hit the space bar; once you do so, it gives you the next page. If you want to quit (e.g., before reaching the end of the file), you can type 'q'. This behavior is actually implemented under the hood using the more command; you can find out more details by typing man more.

If you know the name of the command (say, tail), you simply type it following man (i.e., man tail).

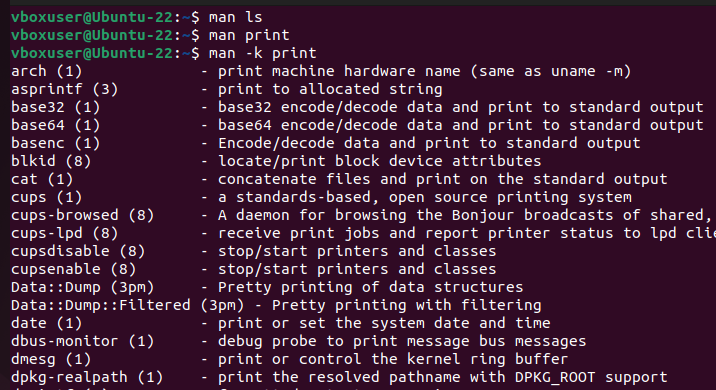
If you don’t know the name of a command but know a keyword that might be part of the command's description, use the -k switch. Thus, if you wanted to know about commands that are associated with printing, you might say man -k print or man -k print | more (if you don't want things scrolling off the screen)

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.

**Section II.b.** Name three command-line commands that have “print” in the description:

Screenshot:



1. arch

2. *asprintf*

3. base32

**Section II.c.** Locate and execute commands to perform each of the following. Use man -k (i.e., apropos) if necessary. You may have to install the calendar. If the commands give you an error message, you may need to install **calendar**. Here are the commands to install:

sudo apt update

sudo apt install gnome-calendar

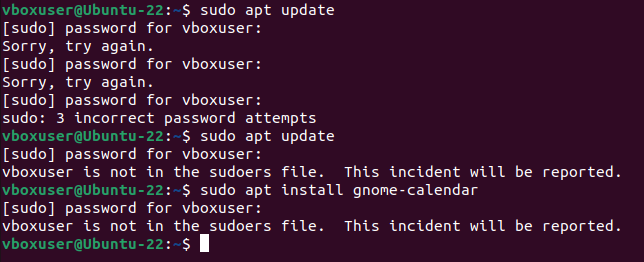
* In one window, print to the screen the calendar for September 1752.   
  **What is strange about this month?**

Answer:

{*insert answer here*} cannot figure out

* In a second window, print to the screen 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 and typing what is provided to you, 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

**Section III.a.** What is the difference between ls, ls -a, and ls -al?

Answer:

*ls – displays contents of a directory*

*ls -a – dispays all directory content, including hidden content*

*ls -al* displays a long listing of all the contents in the current working directory, including hidden content.

**pwd – Present Working Directory**

File System Tree

Similar to Windows and other operating systems, Unix and Linux systems are arranged in a *hierarchal directory structure.* It means that these files and directories are placed in a tree-like pattern. The first directory in a file system is called the *root directory*. The *root directory* contains files and subdirectories, which contains more files and subdirectories. Most GUIs today include a file system manager to view and manipulate contents on the file system. For learning purposes, this is what the file system looks like (the / is the *root directory):*

A screenshot of a computer

Description automatically generated

One important difference between the legacy Unix operating system and Unix-like operating systems such as Linux is that Linux does not employ the concept of drive letters. While drive letters split the file system into a series of different trees (one for each drive), **Linux always has a single tree**. Different storage devices may contain different branches of the tree, but there is always a single tree.

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).

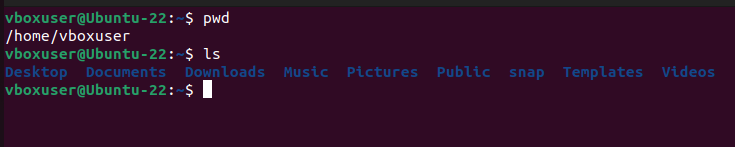
**Section III.b**. What prints to the screen when you enter pwd?

Answer:

*Displays current working directory*

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”)

Screenshot:



1. Desktop

2. Documents

3. 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.

First, with an absolute pathname:

cd /usr

pwd

Or, with a relative pathname:

cd /usr/bin ***(absolute path change to return to /usr/bin again)***

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

pwd

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 change to return to /usr/ again)***

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

pwd

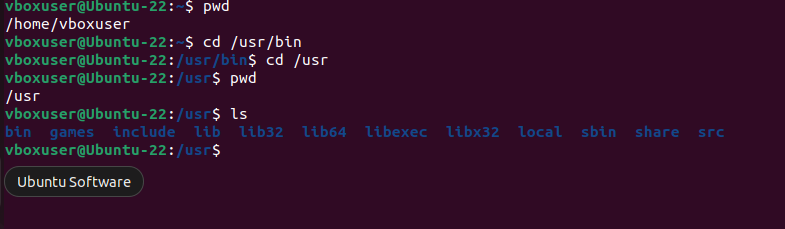
There are several other shortcuts with the cd command. You can issue cd itself on the command line to return to the user’s home directory:

cd

pwd

**Section III.c.** Where are you now?

Answer:



Name 3 things in this directory.

1. {bin}

2. {games}

3. {local}

**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.

**Section IV.a.** To get started with vi/vim, you’ll go through the tutor exercises that come pre-installed with most Unix systems. Fire it up with the following command:

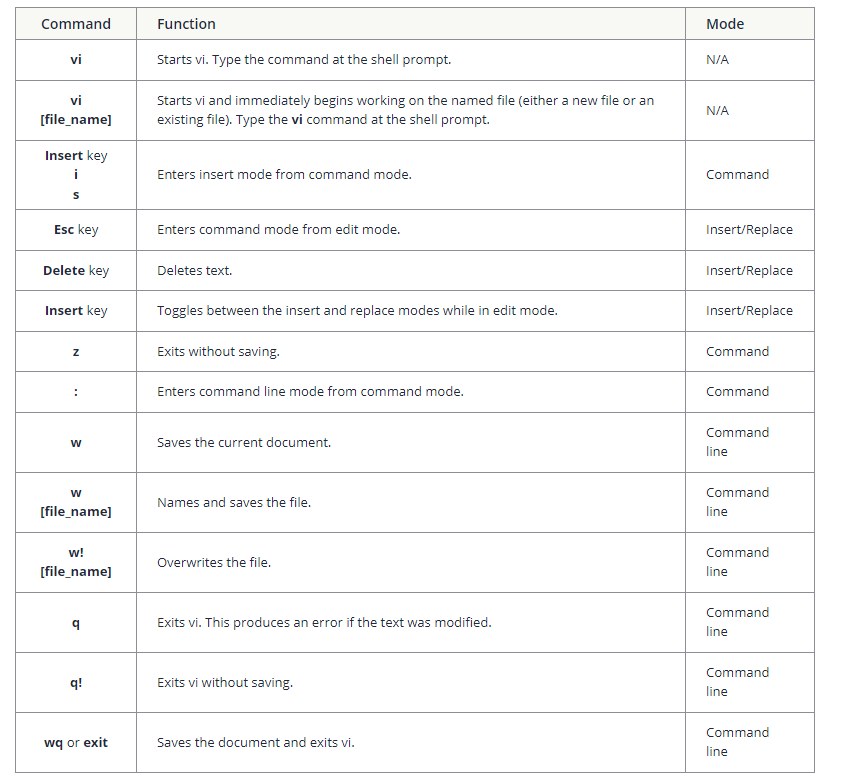
vimtutor

*\*or if you are comfortable with TestOut info…*

*\*or upload a cheat sheet \*\*\*(will upload a cheat sheet)\*\*\**

Read through the document and take a screenshot and paste it into this document

Screenshot:



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.

**Section V.a.** 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.

Screenshot a.):

A screenshot of a computer

Description automatically generated

Screenshot b.):

A screen shot of a computer program

Description automatically generated

1. Most of the commands are same in Unix [↑](#endnote-ref-1)