The objective of this first lab assignment is to familiarize yourself with one of the available editors and several Linux commands. Be sure to read this lab carefully and completely, especially if you are unfamiliar with this environment. The commands listed here are generally available on most Linux implementations.

1. **Navigating Linux**

Now let’s type some nonsense characters and press the enter key:

mat0299@cse01:~$ **ksjdksjf**

ksjdksjf: command not found

If all went well, you should have gotten an error message similar to the above complaining that it could not understand you. Now, if you press the up-arrow key, you should see your previous command "ksjdksjf" return. This is called *command history*. Press the down-arrow and we get the blank line again. Recall the "ksjdksjf" using the up-arrow key if needed. Now try the left- and right-arrow keys. You can position the text cursor anywhere in the command line. This allows you to easily correct mistakes.

The files on a Linux system are arranged in what is called a *hierarchical directory structure*. This means that they are organized in a tree-like pattern of directories called folders, which may contain files and other directories. The top-level directory in the file system is called the *root* directory. The root directory contains files and subdirectories, which contain more files and subdirectories and so on.

Since a command line interface cannot provide graphic pictures of the file system structure, it must have a different way of representing it. Think of the file system tree as a maze, and that you are standing in it. At any given moment, you can stand in a single directory. And inside that directory, you can see its files and the pathway to its parent directory and the pathways to subdirectories of the directory in which you are standing.

The directory you are standing in is called the *working directory*. To find the name of the working directory, use the pwd (i.e., print working directory) command.

mat0299@cse01:~$ **pwd**

/home/mat0299

When you first log on to a Linux system, the working directory is set to your home directory. This is where you put your files.

To list the files in the working directory, use the ls (i.e., list files and directories) command:

mat0299@cse01:~$ **ls**

test1 test4.py test.out a.out

test2.c test6.c test2.cpp csce1030

and many more…

To change your working directory, you use the cd (i.e., 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*.

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. Let’s try this out:

mat0299@cse01:~$ **cd /usr/bin/**

mat0299@cse01:/usr/bin$ **pwd**

/usr/bin

mat0299@cse01:/usr/bin$ **ls**

2to3 gr\_plot\_fft\_c pnmconvol

2to3-2.7 gr\_plot\_fft\_f pnmcrop

and many more…

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 should already be 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 current working directory. To do this, it uses a couple of special symbols to represent relative positions in the file system tree. These special symbols are "." (dot) and ".." (dot dot).

The "." symbol refers to the current working directory itself and the ".." symbol refers to the working directory’s *parent* directory.

Now let’s say that we wanted to change the working directory to the parent of /usr/bin, which is just /usr. We could do this in two different ways. First, we can utilize an absolute pathname using the command cd /usr. Or we can simply use a relative pathname as we show now:

mat0299@cse01:/usr/bin$ **cd ..**

mat0299@cse01:/usr$ **pwd**

/usr

Likewise, we can change it back to /usr/bin in two different ways. We can use an absolute pathname as before, typing cd /usr/bin. Or we can use a relative pathname as we show now:

mat0299@cse01:/usr$ **cd ./bin**

mat0299@cse01:/usr/bin$ **pwd**

/usr/bin

In general, if you do not specify a pathname to something, the working directory will be assumed. Therefore, in almost all cases, you can omit the "./" as it is implied. So we could have just typed cd bin above instead. An important exception to this, however, is when executing compiled programs as you might do in a C or C++ course. To execute an executable compiled program called a.out in your current directory, for example, you would type ./a.out. For this class, however, you will either type python3 to work interactively with Python or you will type python3 followed by the name of your Python program file, so no ./ prefix is needed.

Two directory shortcuts consist of the following:

* If you type cd followed by nothing, cd will change the working directory to your home directory.
* If you type cd ~user\_name (where user\_name is a specific user ID), cd will change the working directory to the home directory of the specified user.

1. **Executing Common Linux Commands**

Some common Linux commands that you should be familiar with are:

* ls list the contents of a directory
* man show the manual page for a command
* mkdir make a new directory
* rmdir remove a directory
* cp copy a file
* cd change directories
* rm remove a file (be careful using this command)
* sort sort lines of a text file
* grep print lines matching to a pattern
* pwd display the location of the current directory

If you are still not familiar with any of these commands after this lab exercise is completed, please review the manual pages for these commands to ensure that you understand and know how to use them (more details about how to use the manual pages will be provided shortly).

As previously discussed, when you first login, your current working directory is your home directory. To find out what is in your home directory, first type **cd** (to get you back to your home directory) and then use the ls command by typing

mat0299@cse01:/usr/bin$ **cd**

mat0299@cse01:~$ **ls**

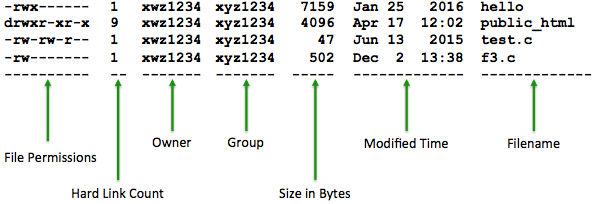
test1 test4.py test.out a.out

test2.c test6.c test2.cpp csce1030

and many more…

The **ls** command can take options. For example, "**-al**" is an option that can be used as in "**ls –al**". Check it out and see what it does. The options change the behavior of the command and can be very helpful when working with files.

The following diagram illustrates the various parts of this listing.



In Linux, "**man pages**" are useful when you do not know how to use a command. A man page describes commands and system calls from the reference manuals, often providing a lot of detail how to use a command or understand what is output, especially regarding supported command options.

For example, type "**man mkdir**", without the quotes and see what is output to the screen. At the bottom of your screen, you should see directions to "press h for help or q to quit". In addition, if the manual page is longer than what can fit on the screen, you can hit your spacebar to go on to the next page of the manual entry.

We now want to create a new directory in your home directory called "**Lab01**", again, without the quotes. To do this, type

**mkdir Lab01**

This will create a new directory called **Lab01** in your current directory (which should be your home directory, assuming you have not changed directories). To see the directory you have just created, type

**ls**

Verify that the **Lab01** directory has been created in your home directory. Now, to copy some files that we will work with, we need the "**cp**" command. Type

**cp -r ~mat0299/public/csce1035/lab01/\* Lab01**

The "**-r**" option, as you can see for yourself by doing a "**man cp**", says to copy recursively. This means that all files and subdirectories in the ~mat0299/public/csce1035/lab01 directory will be copied. The "**\***" is a wildcard character that matches any string (i.e., filename or directory name). You should now have a copy of all of the files inside your **Lab01** directory.

As previously discussed, the command "**cd**" changes the current working directory to the specified directory. The current working directory may be thought of as the directory you are in. To change to the directory you recently made, type

**cd Lab01**

Now, type "**ls**" to see the contents that should include a file called integers and a directory called programs with a number of .py files contained inside the directory that you copied over.

Pathnames enable you to work out where you are in relation to the file-system. You can type "**pwd**" to see where in the file-system you are. If you get lost, you may always type "**cd**" by itself to return to your home directory.

You may refer to the **Linux Quick Reference** PDF file on Canvas for more assistance in Linux commands.

1. **Creating and Editing a Text File**

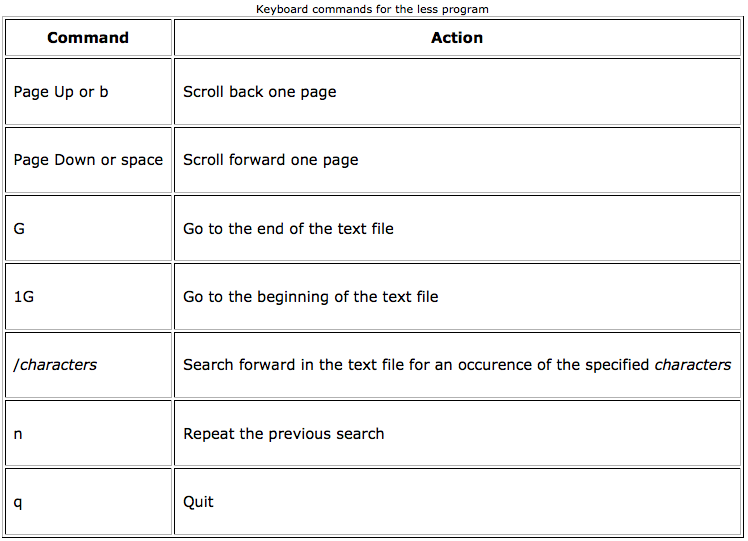
There are several editors available on the CSE machines. The two most widely used editors on these machines are **nano** and **vim**. The **nano** editor is an improved open source version of **pico** available on Linux systems, while **vim** is an improved version of **vi** available on Linux systems that includes enhancements such as syntax highlighting and command line editing. See the **Nano Cheat Sheet** and **VI-Help-Sheet-011** PDF files on Canvas for a quick summary of **nano** and **vim** commands, respectively.

Now, using one of these editors of your choice, create a file called "**musings**", without the quotes and write one sentence about what you hope to get out of this class in that file. Then save the file and quit the editor.

One ***important*** concept that you should become very familiar with is how to quit your editor by saving *and* not saving the file. For example, if you make changes that you want to keep, you should save and then quit the file. However, if you have written a program, for example, and you just want to look at it (but not make changes), you should know how to simply quit without saving the file.

To display the contents of your file on the screen, you can use the "**cat**", "**more**", or "**less**" commands, as in "**cat musings**". Regarding the **less** command, once started, **less** will display the text file one page at a time. You may use the Page Up and Page Down keys to move through the text file. To exit **less**, type "**q**", without the quotes.

Here are some commands that **less** will accept.



Note that you will submit this file to Canvas.

1. **Redirecting and Sorting**

We use the "**>**" symbol to redirect the output of a command. For example, to create a file called **list1** containing a list of animals, type

**cat > list1**

Then, type in the names of some animals. Press **[Return]** after each one.

**giraffe**

**dog**

**cat**

**horse**

**bird**

**frog**

**alligator**

**elephant**

**^D** (Control-D to stop)

What happens is that the **cat** command reads the standard input (i.e., the keyboard) and the "**>**" redirects the output into a file called **list1**. To see the contents of this file, now type

**cat list1**

We use the "**<**" symbol to redirect the input of a command. The **sort** command alphabetically or numerically sorts a list. Now, suppose we want to sort our list of animals alphabetically. To do this, we type

**sort < list1**

You should be able to see that the animals are output in sorted order on the screen. However, if you perform the "**cat list1**" command again, you will see that **list1** still contains the animals in its original unsorted order. If we wanted to keep that list of animals in sorted order, we can type the **sort** command again, redirecting the input, but also redirecting the output to a file, called **sortedList1**. Thus, type

**sort < list1 > sortedList1**

This command will sort the animals in the **list1** file and redirect the output (i.e., the sorted list of animals) to the **sortedList1** file. You can type "**ls**" to see the **sortedList1** file. Then, type

**more sortedList1**

You can see that the list of animals is now sorted alphabetically in the **sortedList1** file.

Now, let’s look at one particular file that you copied into your **Lab01** directory called "**integers**", without the quotes. We would like to sort this file containing integers in numerical order (i.e., a numerical sort) and store the newly sorted integers in a file called **sortedIntegers**. Note carefully that this will require you to use one of the **sort** command options, not just the "default" **sort**. Feel free to use the **man pages** for **sort** if you are having any difficulty. Note that you will submit this file to Canvas.

1. **Executing Python Code**

Next, let's move into some of the sub-directories of your **Lab01** directory. We use the "**cd**" command to move from one directory to another.

**cd programs**

Now, let's take one of the existing Python programs in this directory and run it. To run this Python code, you will need to invoke the **python3** shell command, followed by the name of the file. To see the list of Python file (i.e., those files with the **.py** file extension), let’s use a slightly more complicated version of the “**ls**” command, namely

**ls –lt**

The "**-lt**" means print out the long (i.e., more information) form of the command and to list the files in decreasing order of most recent access. You should see a Python file called **variable\_demo3.py**, so let’s actually run (i.e., execute) the program. To do that type

**python3 variable\_demo3.py**

You should see some output regarding top speed and distance traveled.

You may want to try your hand at running some of the other Python programs in this directory as well to see their differences and how they work.

1. **Finding Files Containing a Matching Pattern**

Finally, let's try finding those .py program files that contain the word "**name**", without the quotes. You should already be in your **Lab01** directory. In this part of the lab, we will capture a screenshot of your work and results using the **script** command that simply captures everything output to your terminal session. To start, type

**script**

You should see the message, "Script started, file is typescript" output on your terminal. This means that everything in your terminal session is now being captured to a file called **typescript**. To search for .py program files that contain the word "**name**", without the quotes, we use the **grep** command. This command takes two arguments: (1) the pattern you want to find, and (2) the file(s) that you want to look through for this pattern. So to find the files containing our desired word (note that it is case-sensitive), we can type

**grep name \*.py**

You should hopefully see 4 lines from 2 different files that contain the word "**name**", without the quotes.

The pattern given (in this case, the word "**name**", without the quotes) is case-sensitive. Now, enter the command to find the files that contain the word "**Name**", without the quotes. In this case, you should have only found 2 lines from 2 different file containing this word.

When looking for certain coding examples, you can use the **grep** command to search through files for that specific pattern, which can be really helpful in this class!

Now, let’s save the typescript of your terminal session by typing **exit**. You should see the message, "Script done, file is typescript" output to your terminal, indicating that you are no longer capturing the output of your terminal session. Note that this **typescript** file will be submitted to Canvas.

Now that you have completed this lab, it’s time to turn in your results. Each of the computers should have a "**WinSCP**" program. WinSCP allows you to transfer files between your computer and the server (i.e., CSE machine) using drag-and-drop. Your TA will likely demonstrate the use of this during the lecture period of the lab. It will have a click and drag feature so that you can copy files, from the CSE machine that you're logged on to, to the Windows machine that you're sitting at. Once you've moved the files to your windows machine, you may use the browser to submit them to Canvas for the **Lab01** dropbox.

You should submit the following files:

* **musings**
* **sortedIntegers**
* **typescript**

Ask your TA to check your results before submission.

Now that you've finished the lab, use any additional time to experiment with your editor of choice and other Linux commands included in this exercise.