PCC Linux and Vim Manual

Table of Contents

Chapter 1: Getting Started with Linux

What is Linux?

Linux is an operating system in the same way that Windows, Mac OS X, iOS, and Android are all operating systems. Essentially, what an operating system does is provide a platform for everything else on your computer to run on top of. This platform is made up of lots of different parts. Some parts are responsible for making the hardware work, others for displaying the user interface, and still other parts for ensuring that applications can work with the hardware and each other. Just like Windows, Linux performs all of these functions. You may be asking, "If Linux does all the same things that Windows or Mac OS X does, why bother switching?". The answer is that Linux has its way of doing things, and for some people, the Linux way suits them better.

Did you know?

Whether you know it or not, you are already using Linux every day. Every time you use Google or Facebook or any other major Internet site, you communicate with servers running Linux. Most DVRs, airplane and automobile entertainment systems, and recent TVs run on Linux. Most ubiquitously, if you are using an Android phone, you are using a flavor of Linux.

At its core, Linux is software used to control hardware like desktop and laptop computers, supercomputers, mobile devices, networking equipment, airplanes, and automobiles; the list is endless. Linux is everywhere.

Linux development is community-based; as people improve and create clever ways to make Linux work on an almost infinite variety of devices and platforms, they contribute their changes back so other people can continue to build on or be inspired by them. In short, it is this very community that helps drive the massive growth and versatility of Linux.

Because so many people are using and contributing to Linux, the software is better and more versatile than what any one company or individual could possibly create on their own.

The good news is that you don’t have to know very much about Linux to use it. You can take it for granted and use it invisibly every time you search the Internet, use your smartphone, or use a device with Linux powering it under the hood. That is what most people do, and there is nothing wrong with it. We will learn some basic commands to work in the Linux environment, but don’t worry, we don’t need to become an expert in Linux or take a course just in Linux. It will be good as you progress to keep learning more as it will be part of most of our computer science future!

Linux is as much a phenomenon as it is an operating system. To understand why Linux has become so popular, it is helpful to know a little bit about its history. The first version of UNIX was originally developed several decades ago and was used primarily as a research operating system in universities. High-powered desktop workstations from companies like Sun increased in the 1980s, and they were all based on UNIX. Several companies entered the workstation field to compete against Sun: HP, IBM, Silicon Graphics, Apollo, etc. Unfortunately, each one had its own version of UNIX, and this made the sale of software difficult. Windows NT was Microsoft's answer to this marketplace. NT provides the same sort of features as UNIX operating systems -- security, support for multiple CPUs, large-scale memory and disk management, etc. -- but it does it in a way that is compatible with most Windows applications.

The **kernel** is the core of the Linux operating system. When we use the term shell, it means that it is the **command line** interpreter. The command line is where you type in commands or requests. The **shell** then processes those commands and tells the kernel what to do. A **script** is then a collection of commands stored in a file. A **terminal** is where we work to enter in commands.

Linux is not only great for large servers like what we have supporting our curriculum, but also for laptops! **It provides high security, high stability, ease of maintenance, runs on any hardware, and maybe best of all, it is free!**

How to Login

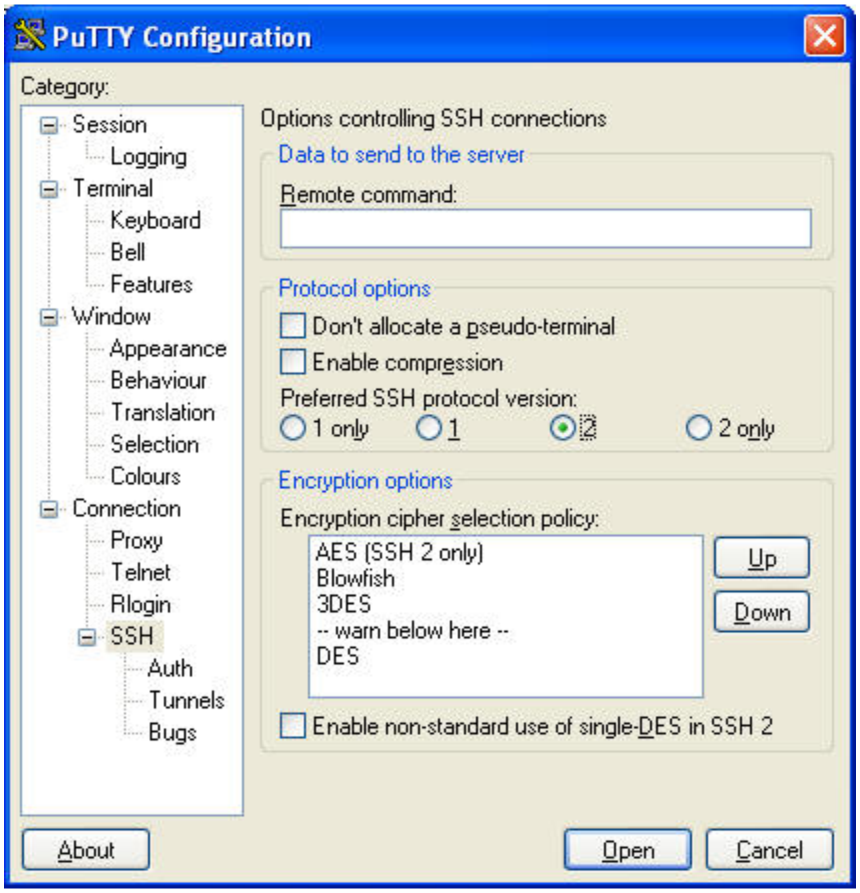
As a PCC student, you have access to the Linux server within the school and can log into it from your home. Having a basic understanding of how your files are stored on our systems will help you be successful in our computing environment.

Work created this term, and all accounts are only active for this term; files do not persist after the course has concluded. The host address is **`cslinux.pcc.edu`**

Windows OS

If you are using a computer with Windows operating system, you will need to download and install a piece of software to login to the PCC Linux server called Putty. Follow the instructions below to install:

1. Download Putty
2. Run putty by clicking on putty.exe
3. Under **Category,** click once on **SSH** and make sure to check the radio box 2, under Preferred SSH protocol version:



Options controlling SSH connections

Data to send to the server

Remote command:

Protocol options

☐ Don't allocate a pseudo-terminal

☐ Enable compression

Preferred SSH protocol version: 1 only 1 [✓ 2] 2 only

Encryption options

Encryption cipher selection policy:

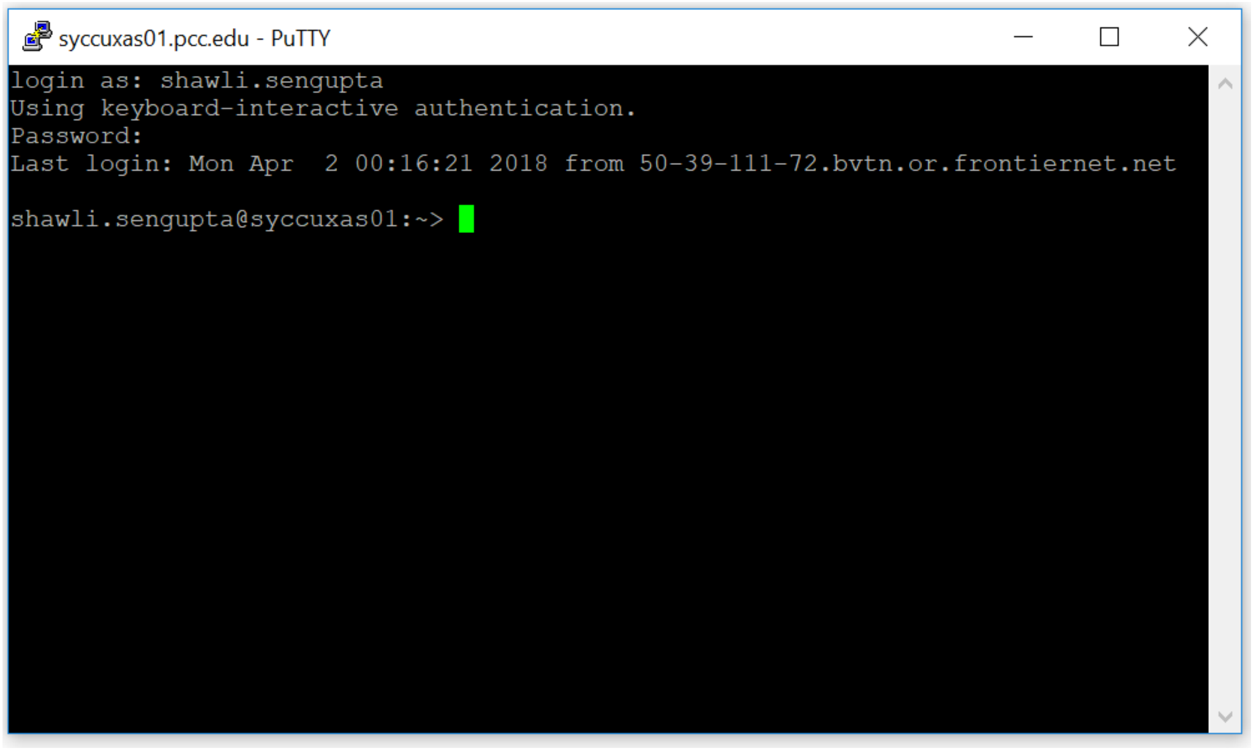
AES (SSH 2 only)
Blowfish
3DES
-- warn below here --
DES

☐ Enable non-standard use of single-DES in SSH 2

4. After changing the **Preferred SSH protocol** version to 2, under **Category,** click on **Session**.
5. In the Host Name (or IP address) box, type: **cslinux.pcc.edu**, also make sure to click on the radio button SSH.
6. You can type a name under Saved Sessions, like PCC, and click on the Save button to save the configuration.
7. Click on the **Open** button at the bottom of the putty window.

Login to the Linux Server

8. You should see the window that will ask you for your login as: (it should be the same name as your mypcc login name firstname.lastname), then your password should be GXXXXXXXX (your G Number, the password will not echo back as you type it).
9. Here is a Student Instructions document that tells you how to login from different OSs.
10. Make sure to change it after your first login with the command **`passwd`** . Once you change the password, keep it in a safe place, and DO NOT FORGET IT!
11. If your login is successful, you should see the shell prompt:

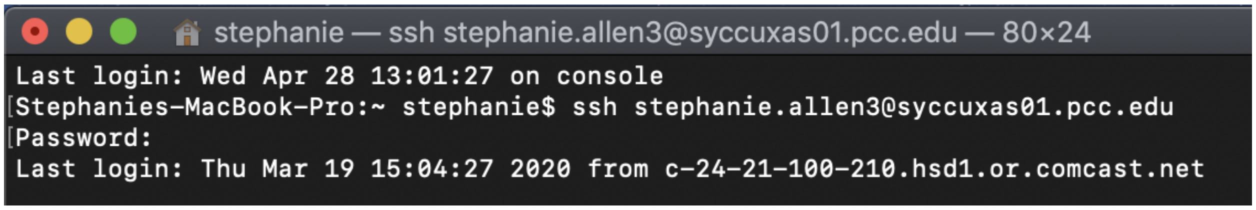

syccuxas01.pcc.edu - PuTTY

```
login as: shawli.sengupta
Using keyboard-interactive authentication.
Password:
Last login: Mon Apr  2 00:16:21 2018 from 50-39-111-72.bvtn.or.frontiernet.net

shawli.sengupta@syccuxas01:~>
```

Apple Macintosh OS X:

1. Terminal software is available as part of the Apple Utilities, allowing us to open a window for remote login.
2. Follow these steps:
    a. Open the Finder and navigate to Applications.
    b. In Applications, navigate to Utilities.
    c. Find a program called Terminal and double click on it.
3. To login, type: **`ssh <your mypcc login name>@cslinux.pcc.edu`**
4. Type in your password; note - it will not display as you type!


stephanie — ssh stephanie.allen3@syccuxas01.pcc.edu — 80×24

```
Last login: Wed Apr 28 13:01:27 on console
[Stephanies-MacBook-Pro:~ stephanie$ ssh stephanie.allen3@syccuxas01.pcc.edu
[Password:
Last login: Thu Mar 19 15:04:27 2020 from c-24-21-100-210.hsd1.or.comcast.net
```

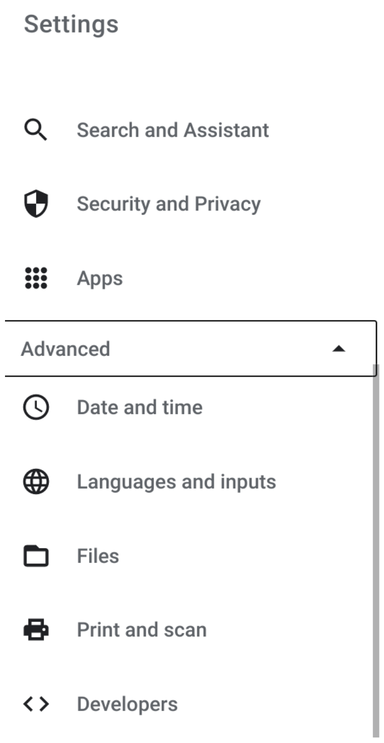
Set terminal type

Once you login, you will need to type: **`TERM=vt100`** at the shell prompt. This is just the first time. You won't need to do this again unless you change the terminal type.

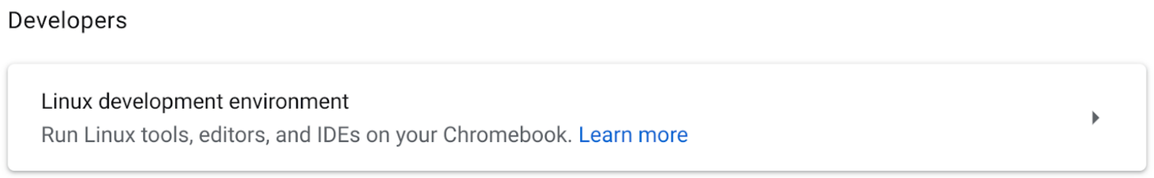
Chrome OS

How to turn the Linux environment on:

1. Go to Settings and click open Advanced and then click Developers

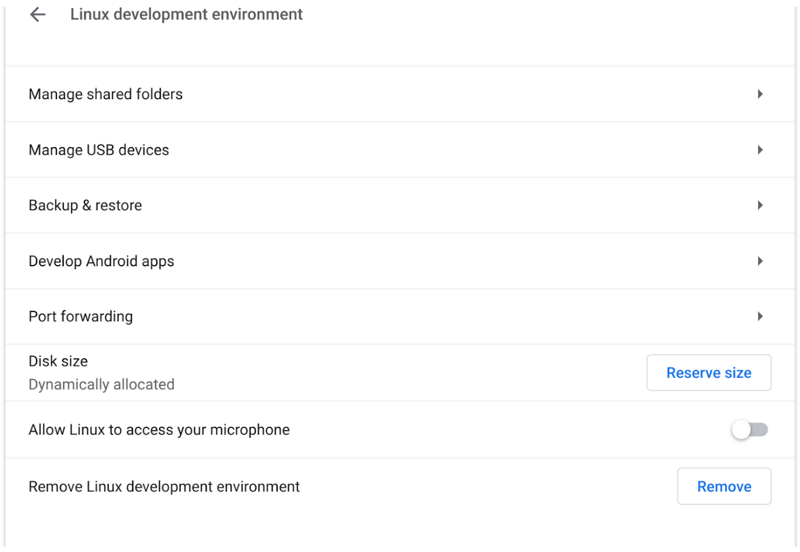


2. Click into the Linux development environment


Developers

Linux development environment
Run Linux tools, editors, and IDEs on your Chromebook. Learn more

3. At the bottom, you can add or remove the Linux development environment


Linux development environment

Manage shared folders

Manage USB devices

Backup & restore

Develop Android apps

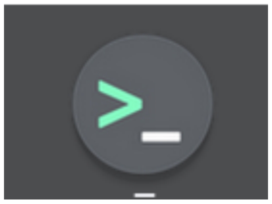
Port forwarding

Disk size
Dynamically allocated

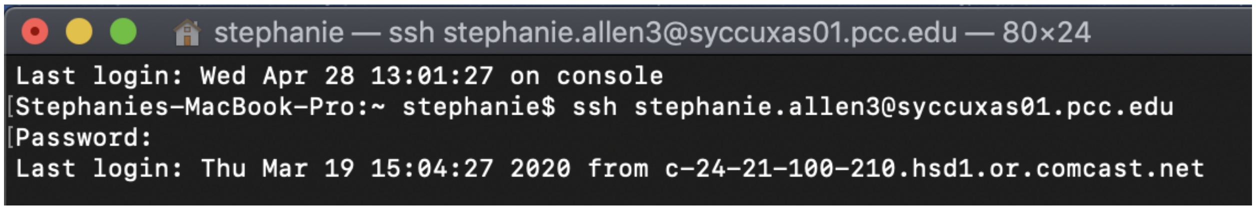
Allow Linux to access your microphone

Remove Linux development environment

4. After the Linux environment is set up, there will be a Terminal Icon in the shelf

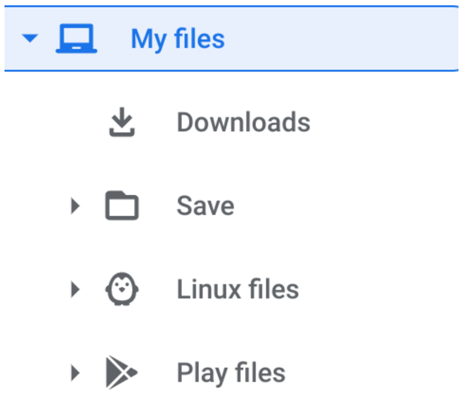


5. To login, type: **`ssh <your mypcc login name>@cslinux.pcc.edu`**
6. Type in your password; note - it will not display as you type!


stephanie — ssh stephanie.allen3@syccuxas01.pcc.edu — 80×24

```
Last login: Wed Apr 28 13:01:27 on console
[Stephanies-MacBook-Pro:~ stephanie$ ssh stephanie.allen3@syccuxas01.pcc.edu
[Password:
Last login: Thu Mar 19 15:04:27 2020 from c-24-21-100-210.hsd1.or.comcast.net
```

7. You can access Linux files via Files in the shelf



Changing Password

To change your password, type the command :

```
passwd
```

The system will then ask for your old password, to prevent someone else from sneaking up and changing your password. Then it will ask for your new password. You will be asked to confirm your new password, to make sure that you didn't mistype. It is very important that you choose a good password so that someone else cannot guess it.

Here are some rules for selecting a good password:

- Do not use any part of your name, your spouse's name, your child's name, your pet's name, or anybody's name. Do not use any backward spellings of any name, either.
- Do not use an easily guessable number, like your phone number, social security number, address, license plate number, etc.
- Do not use any word that can be found in an English or foreign-language dictionary.
- Do not use all the same letter or a simple sequence of keys on the keyboard, like qwerty.
- Do use a mix of upper-case and lower-case letters, numbers, and control characters.
- Do use at least six characters.

Working with Directories

When logging into the Linux server, we recommend creating a separate directory for each of your programming assignments this term. All C++ programs have a .cpp extension!

Linux Commands:

The table below lists some of the useful commands you will be using:

`mkdir CS161B`	Make a directory (folder)
`cd CS161B`	Change directory (go into the folder)
`pwd`	**P**rint **W**orking **D**irectory, prints the path of the working directory, starting from the root

`rm projx`
`rm -r projx`
	Remove the executable file (delete) projx.
	Removes projx and any directories/files in projx.

`ls -l`	List files in a directory with permissions, etc.

Programming Assignments

Programming assignments **from Module 6 (CS161B)** should ideally be written on the PCC Linux server.

1. The first time:
    a. The first time you log in to do your Computer Science programming assignments, create a directory to create programs for this course.
    b. To make a new directory use the `mkdir` Linux command. For example:

       ```
       mkdir CS161B
       ```

2. Once you login, navigate to the directory containing the files you intend to work on. For example, to change into that directory type:

   ```
   cd CS161B
   ```

3. To make a directory for working on Assignment 1, type:

   ```
   mkdir assign_01
   ```

4. Each time you login:
    a. From then on, you will need to change into that directory to get your work done. This will need to happen each time you login.
    b. To change into that directory to start to work type:

       ```
       cd CS161B/assign_01
       ```

Note:

- **You can add `cd CS161B` to your .bashrc file on the PCC server to go to that directory when you login.**
- **To do this, type nano .bashrc**
- **This will open the .bashrc file which is a text file.**
- **All the way at the bottom, add the line `cd CS161B`, and follow the prompts at the bottom to save the file.**
- **Exit the Linux server and login again - it should change to CS161B folder when you are logged in.**

Programming Assessments (CS162 students mainly)

Preparing for Your CS 162 Programming Assessments. Please check out this document before you start practicing for your assessments. **This is mainly for CS162 students. Others can definitely look into it and learn from it and be prepared for 162.**

Here is a quick video by one of our PCC instructors Li Liang on how to use some quick commands and shortcuts while editing and compiling files using multiple windows and a few other tricks.

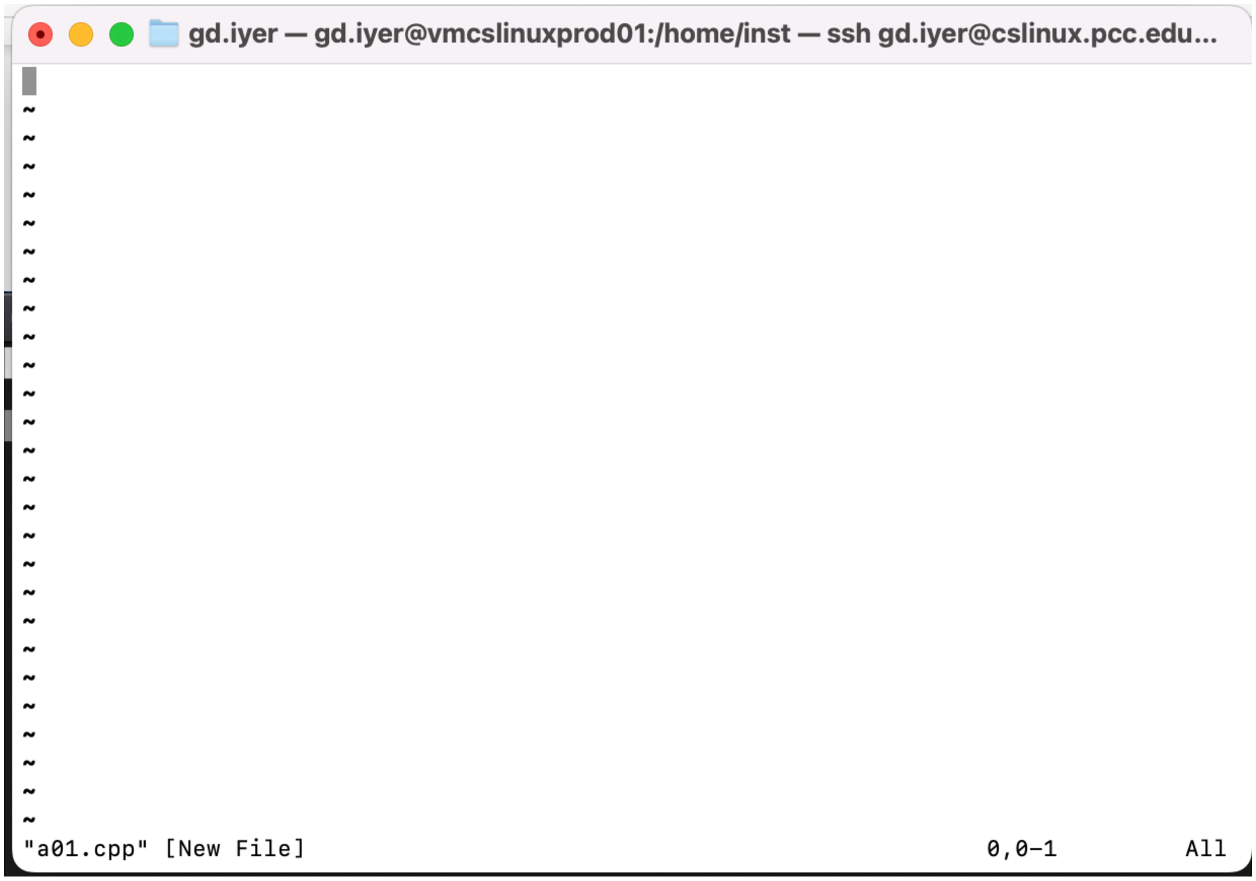
Chapter 2: C++ Programs

Creating a C++ Source File

1. You will be writing your source code using the **`vim`** editor. To create the file named a01.cpp, navigate to the assign_01 directory and type:

   `vi a01.cpp`

2. This opens a new editor window, and you can start editing the file:



gd.iyer — gd.iyer@vmcslinuxprod01:/home/inst — ssh gd.iyer@cslinux.pcc.edu...

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"a01.cpp" [New File]                                          0,0-1         All
```

3. Insert header comments to your file by typing **`i`** followed by comments
4. When you are done, hit the **`escape`** key
5. Now, let's say you wanted to add more at the end of the line you were working at. To get to the end of the line type **$**
6. Now append some additional text by typing **a** followed by text
7. When you are done, hit the **`escape`** key
8. Let's do this again, but this time we will append and get to the end of the line using the capital **A** followed by text
9. When you are done, hit the **`escape`** key
10. If you want to delete the line just added, type **dd**

11. If you made a mistake and didn't want to delete, you can undo it by typing **`u`**
12. Single characters can be deleted with the **`x`** key. Try it!

Saving our work

13. Now you know how to move the cursor, delete characters, insert text, and exit files without changes with **`:q!`** vim command. Exiting the vim editor without changes does not always produce the desired outcome. To save changes to the file you are editing, use **`:wq`** command in vim command mode. There is also a quicker way to do it, which is by keystrokes **`ZZ`** (two uppercase Z's).

14. Let's now save our work by typing **`:w`**

15. If you want to save and quit type **`:wq`**
16. If you want to quit without saving type **`:q!`**
17. **When you are on Linux, you cannot use your mouse! You will need to navigate using your keys on the keyboard.**
18. Type a basic Hello World program and type **`:wq`** to save your work and quit
19. Type **`ls`** at the command prompt to show the files in the directory. You should now see **`a01.cpp`**.

You will learn more about the **`vim`** editor in Chapter 3 Getting Started with vi/vim of this manual.

Compiling and Running Programs

1. All programming assignments should be completely implemented using the PCC Linux systems. Do not use PC editors or compilers. All C++ programs should have a **.cpp extension.**

2. An approved Linux editor must be used to type in C++ code and programs. Do not use an IDE unless instructed by your teacher to do so. You are allowed to use Replit for the pair programming discussion prompts.

3. Once a program has been created, save your work and exit the editor.

4. The next step is to **compile** the C++ source code file. To compile the a01.cpp code type:

```
g++ -Wall -g a01.cpp -o a01
```

Here are some of the common options for using g++ on the Linux server.

Common command line options:

-v = verbose mode. Causes g++ to display internal information that may be useful when tracking down bugs

-g = include debugger information. The -g option is needed if you want to use gdb at the source code (instead of assembly) level

-std=c++11 = Enforce a particular C++ language standard (C++ 11 in this case). The default for g++ is C++ 17, but C++ 11 is used in most CS classes

-Wall = Display all warnings. This can help find dead code and unused variables.

-o = Used to name the output executable file, if you don't like "a.out"

Here are some examples:

g++ -g -std=c++11 -o main source.cpp #Creates a 'main' executable file from the "source.cpp" source code file

g++ -g -Wall -o main source.cpp &>log.txt #Sends all error messages to "log.txt", which can be useful if they scroll off the screen

5. Linux is “no news is good news”. No message will be displayed if the program successfully compiles.

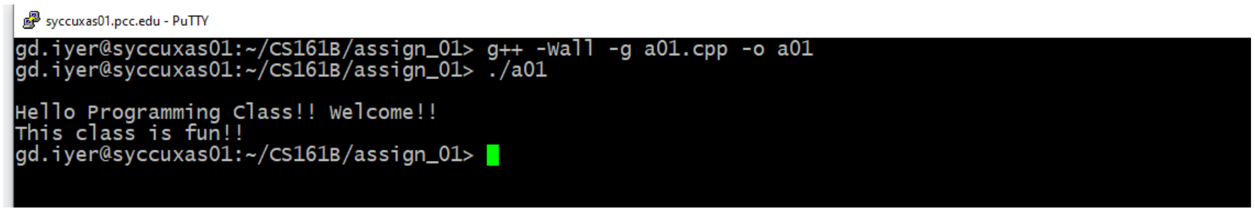
6. If there are syntax errors, examine the line number of the first 2 or 3 errors and fix those before continuing. To fix the syntax errors use the **`vi`** editor again:

   `vi a01.cpp`

7. After the program successfully compiles, a file named **`a01`** will appear in the current working directory by default. Type **`ls`** to view the files.

8. To run your program type **`./a01`,** and the output will appear directly below.

9. Watch this video on quick tips to create multiple files, using multiple windows and quick tips on using Linux.



```
gd.iyer@syccuxas01:~/CS161B/assign_01> g++ -Wall -g a01.cpp -o a01
gd.iyer@syccuxas01:~/CS161B/assign_01> ./a01

Hello Programming Class!! Welcome!!
This class is fun!!
gd.iyer@syccuxas01:~/CS161B/assign_01>
```

File Types and Permissions

The permissions are indicated by a series of rwx's on the left side of the listing. The first position indicates the type of file. For our purposes, it is always a "**-**" for a file, or a **d** for a directory. **x** means execute permission - except for directories where it means search permission.

The first three permission characters are for the **user** (owner), the middle three are for the

group, and the last three are for all **other** people on the system.

`chmod:`

To change access permissions, change mode.

chmod changes the permissions of each given file according to mode, where mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

Numeric mode

The format of a numeric mode is 'augo'

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Any omitted digits are assumed to be leading zeros. The first digit selects the set user ID (4) and set group ID (2) and sticky (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

`chmod 775 dirname` is the normal permission for a directory. drwxrwxr-x

Example: **`chmod 775 assign01.`**

You can read a little more about file permissions in Tutorials Point.

- ❏ Learn about ls by typing **man ls**
- ❏ Learn about the file types on Linux. Type **ls –l *** in your directory.
- ❏ Answer the following for your .cpp file:

  Find the permissions: ________________

  Owner: ________________

  Group: ________________

  Others: ________________

  Number of bytes for your .cpp file: ________________
- ❏ Change the permission of your file for others and group to be non-read, non-write and non-execute.

  ```
  chmod –v og-rwx     filename
  ```
- ❏ Verify that this change has taken place with a **ls –l**
- ❏ Try it again without the –v (verbose) mode to add the permissions back:

  ```
  chmod og+rxw filename
  ```
- ❏ Verify that this change has taken place with a **ls –l**

How to use the Script command on Linux

1. Type “`script output.txt`” on the command line and it will start recording your session in a file called “`output.txt.`”
2. Anything you type or any commands you run after this will get recorded in the file “`output.txt.`”
3. Run your C++ program as usual and test it with some sample input. Do this 3 times with different inputs.
4. Type “`exit`” to stop recording. Your file “`output.txt.`” will contain all the commands from step 2.
5. Watch this video to see how to use the script command.

Submitting Work

Programming assignments must be transferred from Linux and ultimately uploaded to D2L. This process requires some practice to prepare and should be followed carefully!

1. Go to the **assign_01** directory.
2. Remove all executable files by using:

   rm a01

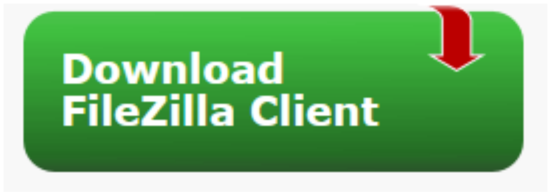
3. **Be careful to remove only the executable and not your .cpp files.**
4. Now you should have just the .cpp file in the directory.
5. Follow the process below to transfer the files to your Windows (or your Mac) and upload to D2L.

Transferring Files

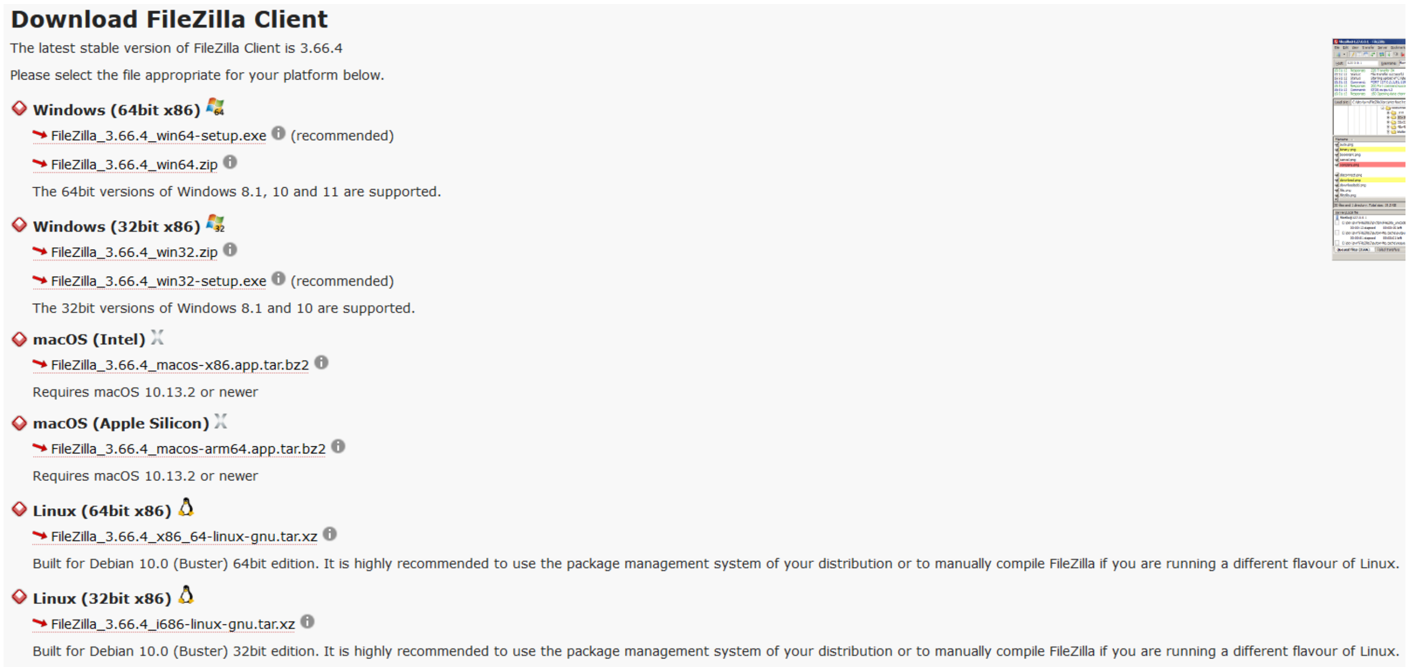
The following outlines the process for transferring files from our Linux system to other platforms. Only upload source files to D2L (for example, .cpp files) and *not* executable files *(a.out)*.

Using Windows or Mac platforms:

1. Download FileZilla client for Windows or Mac. FileZilla is a free, open-source, FTP solution. The File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. **Please read this section completely before you start installing FileZilla client.**
   a. **Note:** Don’t download from the main installer that states “Download FileZilla Client”, as seen below.



b. Instead, use the link above (or click "Show additional download options" on the download page) to access the page shown below: (numbers may change as updates are made)



Download FileZilla Client

The latest stable version of FileZilla Client is 3.66.4

Please select the file appropriate for your platform below.

Windows (64bit x86)
- FileZilla_3.66.4_win64-setup.exe (recommended)
- FileZilla_3.66.4_win64.zip

The 64bit versions of Windows 8.1, 10 and 11 are supported.

Windows (32bit x86)
- FileZilla_3.66.4_win32.zip
- FileZilla_3.66.4_win32-setup.exe (recommended)

The 32bit versions of Windows 8.1 and 10 are supported.

macOS (Intel)
- FileZilla_3.66.4_macos-x86.app.tar.bz2

Requires macOS 10.13.2 or newer

macOS (Apple Silicon)
- FileZilla_3.66.4_macos-arm64.app.tar.bz2

Requires macOS 10.13.2 or newer

Linux (64bit x86)
- FileZilla_3.66.4_x86_64-linux-gnu.tar.xz

Built for Debian 10.0 (Buster) 64bit edition. It is highly recommended to use the package management system of your distribution or to manually compile FileZilla if you are running a different flavour of Linux.

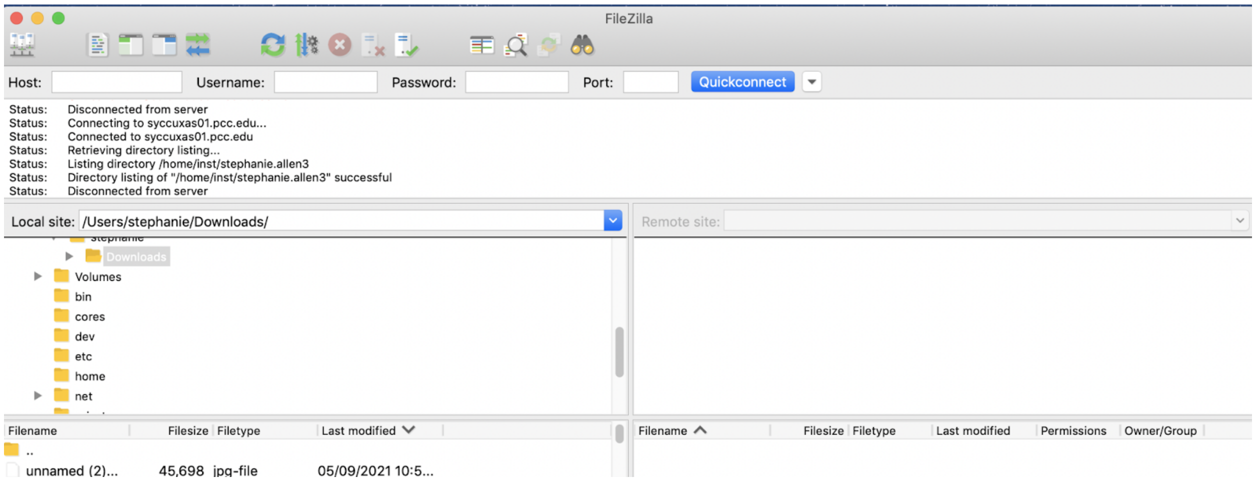
Linux (32bit x86)
- FileZilla_3.66.4_i686-linux-gnu.tar.xz

Built for Debian 10.0 (Buster) 32bit edition. It is highly recommended to use the package management system of your distribution or to manually compile FileZilla if you are running a different flavour of Linux.

c. From this page, download your respective Windows or Mac installer

d. **Why can't I use the main installer?:** The main installer bundles adware into the install files. To avoid downloading potentially unwanted programs, we recommend that you install from the clean versions in the "Show Additional Options" link/the link seen above. This way, you will install only the FileZilla Client, and no other programs.

2. Login to the Linux server, in the host box type **`cslinux.pcc.edu`**. Type your username and password and port 22. Click **Quickconnect**.



```
Status:    Disconnected from server
Status:    Connecting to syccuxas01.pcc.edu...
Status:    Connected to syccuxas01.pcc.edu
Status:    Retrieving directory listing...
Status:    Listing directory /home/inst/stephanie.allen3
Status:    Directory listing of "/home/inst/stephanie.allen3" successful
Status:    Disconnected from server
```

Local site: /Users/stephanie/Downloads/

Remote site:

Filename	Filesize	Filetype	Last modified
..			
unnamed (2)...	45,698	jpg-file	05/09/2021 10:5...

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group

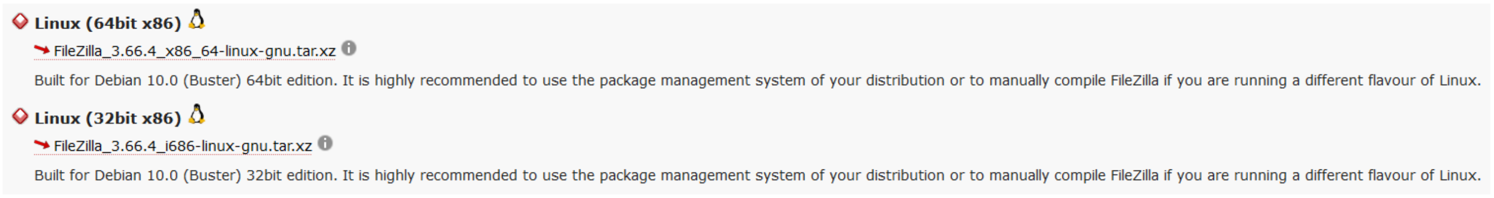
3. After you are logged in, you can drag and drop the file from Linux to your desktop (or desired location)

4. This file can then be uploaded to D2L using the dropbox tool. Make sure when using D2L to hit the submit button. Files uploaded but not “submitted” will not be saved by D2L!!!

5. Double check D2L that your file exists there. Never assume!!

Using Chrome OS:

1. Download Linux version from their website, as seen below:



Linux (64bit x86)

FileZilla_3.66.4_x86_64-linux-gnu.tar.xz

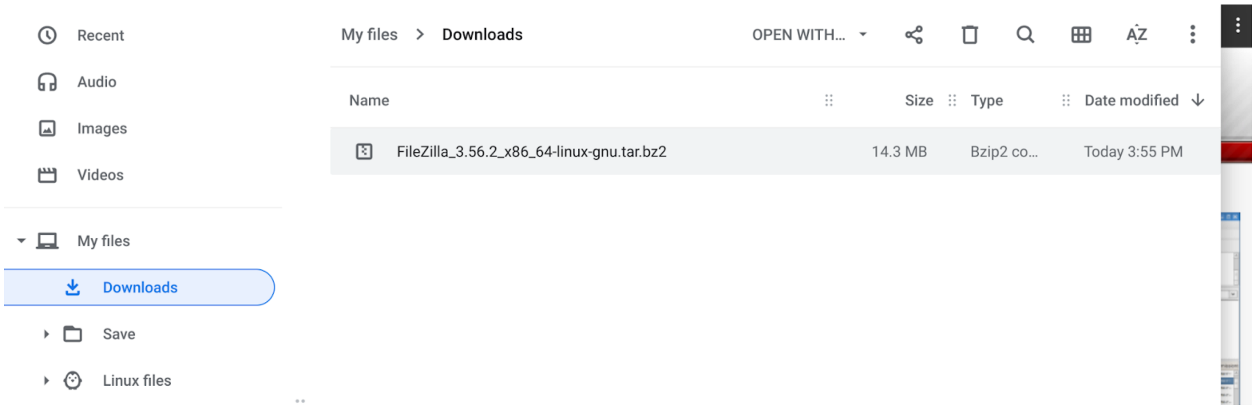
Built for Debian 10.0 (Buster) 64bit edition. It is highly recommended to use the package management system of your distribution or to manually compile FileZilla if you are running a different flavour of Linux.

Linux (32bit x86)

FileZilla_3.66.4_i686-linux-gnu.tar.xz

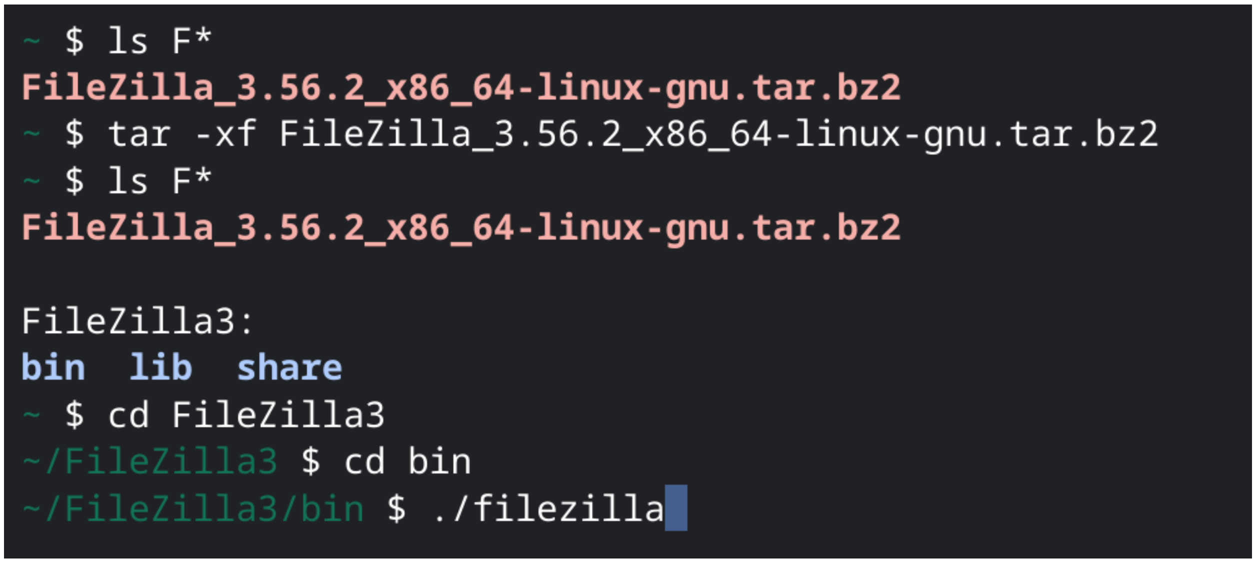
Built for Debian 10.0 (Buster) 32bit edition. It is highly recommended to use the package management system of your distribution or to manually compile FileZilla if you are running a different flavour of Linux.

2. Move from the Download folder to Linux Files



Name	Size	Type	Date modified
FileZilla_3.56.2_x86_64-linux-gnu.tar.bz2	14.3 MB	Bzip2 co...	Today 3:55 PM

3. In the Terminal, use tar to extract the files and then go to the FileZilla3/bin directory and run filezilla.



```
~ $ ls F*
FileZilla_3.56.2_x86_64-linux-gnu.tar.bz2
~ $ tar -xf FileZilla_3.56.2_x86_64-linux-gnu.tar.bz2
~ $ ls F*
FileZilla_3.56.2_x86_64-linux-gnu.tar.bz2

FileZilla3:
bin  lib  share
~ $ cd FileZilla3
~/FileZilla3 $ cd bin
~/FileZilla3/bin $ ./filezilla
```

Sending Mail on Linux - Alpine

On the Linux operating system, pine is a program for accessing email and newsgroups.

Alpine or pine is a screen-oriented message-handling tool. In its default configuration, pine offers an intentionally limited set of functions geared toward the novice user. Still, it also has a growing list of optional power-user and personal-preference features. Pine's basic feature set includes:

1. View, Save, Export, Delete, Print, Reply and Forward messages.
2. Compose messages in a simple editor (pico) with word-wrap and a spelling checker. Messages may be postponed for later completion.
3. Full-screen selection and management of message folders.
4. Address book to keep a list of long or frequently-used addresses. Personal distribution lists may be defined. Addresses may be taken into the address book from incoming mail without retyping them.
5. New mail checking and notification occur automatically.
6. Context-sensitive help screens.

Linux Quick Reference

1. **`mkdir`** – to organize files and directories. This will make a new directory:
   `mkdir CS161B`

2. **`cd`** - change into another directory:

a. To change into a specific directory:	**`cd CS161B`**
b. To change into a directory using a wild card:	**`cd CS*`**
c. To change back to the directory above:	**`cd ..`**
d. To change to the root directory (regardless):	**`cd`**

3. **`ls`** - list directory contents:

a. To list the contents of the current directory:	**`ls`**
b. To list the contents of specific files (e.g., that start with a capital P):	**`ls P*`**
c. To list the contents of a specific directory:	**`ls CS161B`**
d. To list the contents of directories using a wild card:	**`ls CS*`**
e. Add flags to get more detailed displayed:	**`ls -l /etc`**
f. Use the wildcard (*) to list all of the .cpp files in the current directory:	**`ls *.cpp`**

4. **`cat`** – to display the contents of a file: **`cat test1.cpp`**

5. **`pwd`** – to determine your current path and directory name : **`pwd`**

6. **`cp`** - copy a file or directory:
   a. For example, **`cp source dest`** if you want to copy a directory, use the -R option for recursive. The forward slash (in front of the path) means that we are working from the root directory:
   ```
   cp  -R  ./source  ./dest
   ```
   b. The source and destinations may be complete paths or a path from the current directory; the following will copy the entire CS161B directory and all of its files into the CS_Copy directory:
   ```
   mkdir CS_Copy
   cp -R CS161B CS_Copy
   ```

7. **`mv`** - move a file or directory. You could think of this as a rename, but it can also move a file to another directory! Make sure to pay close attention to the order of the arguments. The first argument (source) is the file or directory that you want to copy *from,* and the second argument (dest) is the file that you want to copy to (e.g., the file to be created). Keep in mind that a move does not leave the original (unlike copy) so use it with great caution!

a. Syntax:	**`mv source dest`**
b. Move a file:	**`mv program1.cpp new_name.cpp`**
c. Rename a directory:	**`mv CS161B new_directory_name`**

Exercise #2.1 – Files and Directories

Practice with the following Linux commands; most of these commands must be followed by an enter. Check off each step as they are done:

Working with Files and Directories

_____ 1. First, login to **PCC Linux Server (`cslinux.pcc.edu`)**

_____ 2. When already logged in, start off at the home directory by typing: **`cd`**

_____ 3. Verify that you are in your home directory **`pwd`**

Write down the path that it provides: ______________________________

_____ 4. List the files in your current directory **`ls`**

_____ 5. Move into the directory for your class **`cd CS*`**

_____ 6. Verify that you are in your home directory **`pwd`**

Write down the path that it provides: ______________________________

_____ 7. List the files in your current directory **`ls`**

Write down the path that it provides: ______________________________

_____ 8. What happens when you type ls with the -a flag **`ls -a`**

What additional files are in this directory? __________________________

_____ 9. Go back to the home directory **`cd ..`**

_____ 10. Make a new directory for practicing **`mkdir practice`**

_____ 11. Change into that directory **`cd practice`**

_____ 12. Use an approved editor to create a file called *test file* and put your name and email address in it as a comment. Save the changes and exit the editor.

_____ 13. Verify that the file *test file* exists **`ls`**

_____ 14. List the contents of the testfile to the screen **`cat testfile`**

Copying and Renaming Files

_____15. Clear the window **`clear`**

_____16. Make a copy of the testfile under the name copy; be careful how you use the copy command. The file that contains the information you want copied (the source) is listed first and the name of where you want the information copied to (the destination) is listed second. If these are listed in the wrong order, you may destroy the contents of your *test file!*

```
cp testfile  copy
   (source)     (destination)
```

_____ 17. Verify that both files exist in your directory **`ls`**

_____ 18. Display the contents of your testfile to ensure that it is still intact:

```
cat testfile
```

_____ 19. Clear the window **`clear`**

_____ 20. Use the bang (exclamation mark) to re-run the last used command. For example:

● Runs the last cat command	**`!cat`**
● Runs the last cp command	**`!cp`**
● Runs the last command that started with c	**`!c`**
● listing the files in a directory by typing	**`!l`**

_____ 21. Rename the new copy of your file to *contact_info*

To rename a file we move it to another. Again, it is important to pay close attention to the order of the arguments. The first file listed is the original file that has the contents and the second file listed is the new name. If these are swapped, you will destroy the contents of your file!

```
mv      copy        contact_info
        (source)    (destination)
```

_____22. Verify that both files exist in your directory ls Is contact_info now listed?___________ Is copy gone?___________

_____23. Display the contents of your contact_info, ensure it is intact: **`cat testfile`**

_____24. Clear the window **`clear`**

_____25. Move the file to your home directory **`mv contact_info ..`**

_____26. Verify that the file exists in the home directory **`ls ..`**

_____27. Move the file back into the practice directory **`mv ../contact_info .`**
(The period at the end is important!!)

_____28. Verify that the file does not exist in the home directory **`ls ..`**

Chapter 3: Getting Started with vi/vim

Background Information

There are several powerful text editors that are worthwhile learning. If you find yourself having difficulty with navigation or modifying a source file, spend time with vim. The power available through this editor can significantly speed up development time! All students in CS 162 and CS 201/260 will be expected to become proficient in programming with these commands.

1. The following are common modes used with vim:

   a. **Command mode**: This is the initial mode that you are in when opening up the editor. You can ask to insert, append, delete, and perform other navigation-related commands in this mode. Navigation allows you to move around within the file. In command mode, it is not possible to start typing in text. Everything that is typed is taken as a command. This is why it is called command mode!

   b. **Insert mode**: This is the mode that we enter when you ask to insert or append. Once in insert mode, anything typed is entered into the document (although not saved until that request is executed). To end insert mode, you must use the **escape** key!

   c. **Last Line mode**: From command mode, you can execute specific commands while still examining your document using the colon key. After selecting the colon, a line appears below the editor window (called the "last line" that allows you to perform operators to save, quit, search and replace. To save and quit, we would type *:wq*

      A useful last line mode command is to get help concerning the commands available.

      Follow the :help with the command you would like more information *:help command*

   d. **Visual mode**: Visual mode is available with vim to extend regular use and add the ability to highlight text.

A fun way to learn how to navigate in vim is this Vim Adventures online game!

Exercise #3.1 - vim Basic Commands

Before you start:

1. Make sure you are in the home folder for this class: **`/CS161B`**.
2. Then copy the LinuxLabs folder from my home directory to yours:
   `cp -r ~gd.iyer/LinuxLabs .`
   (The period at the end is important!!)
3. Do not forget the period at the end, which says copy to the current directory.
4. Now, check and make sure you have LinuxLabs in your CS161B folder: **`ll or ls -a`**
5. Now you are ready to do your exercises below.

_____ 1. **Basic Commands**

In vim you can move the cursor around with the following keys h, l, k, j, which are left, right, up, and down respectively. **DO NOT USE THE ARROW KEYS to move the cursor**! Get used to the vim commands! Vim has both command mode and editing mode. Command mode is where we can instruct the vim editor to exit to the command line (shell). To do that, we need to press ESC and type **:q!.** For edit mode, you can type **i** (for insert) or **a** (for append).

_____ a. Login into the PCC Linux server **(`cslinux.pcc.edu`)**
_____ b. Change to your Labs folder **`cd CS161B/LinuxLabs`**
_____ c. Start editing your file by typing **`vim lab1.cpp`**
_____ d. Insert header comments to your file by typing **`i`** followed by comments
_____ e. When you are done, hit the **`escape`** key
_____ f. Now, let's say you wanted to add more at the end of the line you were working at. To get to the end of the line type **`$`**
_____ g. Now append some additional text by typing **a** followed by text
_____ h. When you are done, hit the **`escape`** key
_____ i. Let's do this again, but this time we will append and get to the end of the line using the capital **`A`** followed by text
_____ j. When you are done, hit the **`escape`** key
_____ k. If you want to delete the line just added, type **`dd`**
_____ l. If you made a mistake and didn't want to delete, you can undo it by typing **`u`**
_____ m. Single characters can be deleted with the **`x`** key. Try it!

_____ 2. **Saving our work**

Now you know how to move the cursor, delete characters, insert text, and exit files without changes with **`:q!`** vim command. Exiting the vim editor without changes does not always produce the desired outcome. To save changes to the file you are editing, use **`:wq`** command in vim command mode. There is also a quicker way to do it, which is by keystrokes **`ZZ`** (two uppercase Z's)

_____ a. Let's now save our work by typing **`:w`**
_____ b. If you want to save and quit type **`:wq`**
_____ c. Reopen your file to edit, and insert another line. Now type upper case **`ZZ`**
Do your changes get saved?
_____ d. Reopen your file to edit one last time. Make some additional changes. **`:q!`**
Do your changes get saved? ____________________________

Exercise #3.2 – vim Navigation

_____ 1. **Navigation**

In many cases, navigating large files with thousands of lines requires more sophisticated navigation. We wouldn't want to move the cursor to line 3500 with the **`j`** key. And, don't even think about using the arrow keys to get around; vim has far more powerful commands that you should get to know to navigate your program.

Fortunately, vim developers equipped vim with the **`g`** operator. To get to the end of the file, hit **`G ( SHIFT g )`**. To get to the beginning of the file, use **`gg`** command. To navigate your cursor on **`n`** line, use **`nG`**. If you are unsure how many lines your file has, you can use a **`CTRL+g`** key combination.

_____ a. Login into the PCC Linux server **(cslinux.pcc.edu)**
_____ b. Change to your Labs folder: **`cd CS161B/LinuxLabs`**
_____ c. Start editing your file by typing **`vim list.cpp`**
_____ d. To go to the first line in a file, type **`1G`**
_____ e. Go to line # 5, type **`5G`**
_____ f. To advance to the end of the line you were working at, use **`$`**
_____ g. Let's try going to the end of your file, type **`G`**
_____ h. Now go back to the beginning of the file, type **`gg`**
_____ i. Now let's go forward 20 lines, type **`20 enter`**
_____ j. Now let's go another 20 lines forward, type **`20 enter`** again
_____ k. Go back to the beginning by typing **`1G`**
_____ l. To go physically to line 40, type **`40G`**
_____ m. Now you can quit without storing your changes, type **`:q!`**

(Notice when you get errors and need to get to a particular line number, this is a quick and easy way to do so!)

_____ 2. Advanced Navigation

_____ a. Page down by holding down **control** key and hit **d** **`CTRL+d`**
_____ b. Page up by holding down the **control** key and hit **b** **`CTRL+b`**
_____ c. Go to a specific line in the document (e.g., line 50) **`:50`**
_____ d. Go to the end of the line, type **`$`**
_____ e. Go to the end of the line and start appending in insert mode, type **`A`**

Exercise #3.3 – Making Modifications with vim

_____ 1. Login into the PCC Linux server **(cslinux.pcc.edu)**

_____ 2. Change to your Labs folder **`cd CS161B/LinuxLabs`**

_____ 3. **Deletion**

Now we can delete words instead of just deleting characters one by one! Vim has for deleting words very intuitive command **dw**

- _____ a. Start editing your file by typing **`vim list.cpp`**
- _____ b. Search for a word by typing **`/Java`**
  (to search for main we could type /main)
- _____ c. Delete the word you searched for by typing **`dw`**
- _____ d. If you want to undo your deletion, remember you can type **u**
- _____ e. You can also delete from the current position to the end of the word with **`de`**
- _____ f. If you want to undo your deletion, remember you can type **u**
- _____ g. Now you can quit without storing your changes **`:q!`**

_____ 4. **Replacing**

Replacing characters with the vim editor is a very easy task. Simply get vim to the editing mode, remove the character and insert a replacement character. Well, this approach is logical but there is a way to do it faster with *r* (replace) operator. Move the cursor to the character you would like to replace. First enter r command followed by the character which should be there instead. rt command will replace the current character with t.

- _____a. Start editing your file by typing **`vim list.cpp`**
- _____ b. Move your cursor to the character you would like to replace, then type **`r`** followed by the letter you would like to replace
- _____ c. If you want to undo your replacement, remember you can type **u**
- _____ d. To change a word instead of a character, you can use the **`ce`** command, followed by the word to replace. First position your cursor to the word you would like to replace
- _____ e. Now type **`ce`** followed by the word you would like
- _____ f. Notice this places you in insert mode. To end this mode hit the **`escape`** key
- _____ g. Now, replace until the end of the line you were working at. First position your cursor to the place where you want replace to take place. Then type **`C`** to change to the end of the line; start typing the replacement text
- _____ i. When you are done, hit the **`escape`** key
- _____ j. If you want to undo your change, remember you can type **u**

_____ 5. **Find and Replace**

- _____a. For example, to search for the first occurrence of the string ‘foo’ in the current line and replace it with ‘bar’, you would use:

```
:s/foo/bar/
```

_____b. To replace all occurrences of the search pattern in the current line, add the g flag:

```
:s/foo/bar/g
```

_____c. If you want to search and replace the pattern in the entire file, use the percentage character % as a range. This character indicates a range from the first to the last line of the file:

```
:%s/foo/bar/g
```

Exercise #3.4 – Copy and Paste with vim

Previously, we have seen how we can use the **d (delete)** operator to delete a line and paste it to another location within the file. This time we will do the same, but instead of deleting the line, we will copy it and paste it. The principle is the same as with the **d operator**, but we will use **y (yank)** operator.

Check off each step as it is performed.

_____.1. First, login to **lab system (cslinux.pcc.edu)**

_____ 2. Change to your Labs folder **`cd CS161B/LinuxLabs`**

_____ 3. Start editing your file by typing **`vim list.cpp`**

_____ 4. Position your cursor at the line desired **`$`**

_____ 5. Now try some new ways to navigate. We can move across a line by going to the 4th word with **`4w`**. Try it!

_____ 6. Copy the line with the yank **`yy.`** Now, position your cursor at the new desired place where you want the copied line(s) to be pasted. Now paste with **`p`**

_____ 7. If you want to copy more than one line, you can place the number of lines desired in front of the yank command **`3yy.`**

_____ 8. Now, position your cursor at the new desired place where you want the copied line(s) to be pasted. Now paste with **`p`**

_____ 9. When you are done, hit the **`escape`** key

_____ 10. This time let's experience moving to the end of a word rather than to the beginning. To go to the end of the 2nd word you can type **`2e`**

_____ 11. Now Paste with **`p`**

_____ 12. What happens if you type **`p`** again? ________________________

_____ 13. What happens if you place a number in front of the p? (e.g., **`2p`**) Explain what happened: ____________________________________

_____ 14. Now undo with the **`u`**
How many of the operations were reversed? ________________

_____ 15. Now let's copy multiple lines. Place the number of lines interested in copying in front of the yank command: **`10yy`**

_____ 16. Position your cursor at the newly desired location and paste, with **`p`**
How many new lines appeared? ________________________

Vim Quick Reference

1.	To begin editing a file	**vim file_name.extension**
a.	For example:	**vim prog1.cpp**

Command mode:

2. To navigate in the document to prepare to edit, here are a few of the choices available:

a.	Go to the first line	**1G**	
b.	Move to the top of the screen	**H**	
c.	Move to the bottom of the screen	**L**	
d.	Move the cursor down one line	**j**	
e.	Move the cursor up one line	**k**	
f.	Move the cursor right one character	**l**	(lower case L)
g.	Move the cursor left one character	**h**	
h.	Go to the beginning of a line	**0**	(a zero)
i.	Go to the first non-black character	**^**	
j.	Go to the end of the line	**$**	
k.	Go forward one full screen	**CTRL+f**	
l.	Go backwards one full screen	**CTRL+b**	
m.	Go forward half a screen	**CTRL+d**	
n.	Go backwards half a screen	**CTRL+u**	

3. Other command mode options are:

a.	Undo the last change	**u**
b.	redo the last change	**CTRL+r**
c.	To delete an entire line	**dd**
d.	To copy the current line	**yy or Y**
e.	To paste the copied line(s)	**p**

Insert mode:

4. Enter insert mode; once insert **mode** has been entered, you can type normally inserting text. When done, you will need to exit insert mode with the escape key or by holding down the control key and pressing c at the same time:

a.	Insert before the cursor	**i**
b.	Append after the cursor	**a**
c.	Open a new line below the current line	**o**
d.	Insert at the beginning of a line	**I**
e.	Append at the end of the line	**A**

5. To end insert mode, press **escape key or control c**

Last line mode commands:

6. Options for saving your work and quitting:
    a. To save your work **:w**
    b. To save changes and exit (quit) :**wq**
    c. To quit without saving :**q**! **all changes will be lost!
7. Navigation options at last line mode include:
    a. Go to a particular line **:42** go to line # 42
    b. Find some text in the file **`/text`**
    c. Find the next occurrence **`/`**
    d. Also to find next occurrence **`n`** or **`N`**
    e. Search and replace **`:%s/to_replace/replace_with/<flags>`**
       flags:
       `g` // replace all
       `i` // case insensitive
       `c` // confirm before doing each replacement

Visual mode:

8. To enter Visual mode:
    a. Character mode: **`v (lower-case)`**
    b. Line mode: **`V (upper-case)`**
    c. Block mode: **`CTRL+v`**
9. Once in Visual mode, we can cut, copy and paste. Use your arrow keys to highlight the text.
    a. To delete text and copy it to the clipboard: **`d`**
    b. To copy (yank) text: **`y`**
    c. To paste text that has been copied: **`p`**
10. Or we can alter indentation:
    a. a. To add indentation to the right: **`>`**
    b. b. To reduce indentation (move to the left): <
11. To end visual mode, press **`escape`** **`CTRL+c`**

Advanced vim Commands:

12. **Split Screen**: One of the really great features is working in split screen mode; this allows you to view multiple files simultaneously
    a. Split horizontally: **`vim –o file1.cpp file2.cpp`**
    b. Split vertically: **`vim –O file1.cpp file2.cpp`**
    c. Split all of your .cpp files: **`vim –O *.cpp`**
    d. Switch between screens: **`<CTRL> ww`**

Chapter 4: Creating Backups

Whenever working with computers, it is important to understand that our work is temporary. Unlike when you print or handwrite, the information we type into a computer (e.g., term papers or programming assignments) is not permanently stored. System failures can occur, or user errors can cause our files to be altered or destroyed. This is a relatively common occurrence. Although we hope this doesn't happen, it is not unusual for someone to write over the contents of one of their files without having a backup readily available. As such, this exercise will prepare us to always backup our work.

Always make a backup before

- Making any changes to a working program
- Transferring files
- Submitting assignments

Exercise #4.1 - Creating Backups

_____ 1. The first step is to learn about how to copy files with **`cp`**. To do this, we will use the manual pages on Linux with **`man`**

_____ a. Type **`man cp`**, to learn about the flags for making a copy.
_____ b. List three flags that seem useful when backing up files:

_____ 2. Login into the PCC Linux server **(cslinux.pcc.edu)**
_____ 3. Step through the process of copying an entire directory. First, make sure to be in the appropriate directory **`cd ~/CS161B/LinuxLabs`**
_____ 4. Type **`pwd`** to determine your current path to confirm
_____ 5. Create a new directory for your backup files

```
mkdir ../LinuxLabsBackup
```

_____ 6. Type **`ls ..`** to confirm that the new directory exists
_____ 7. Where is this directory being placed? ______________________________
_____ 8. Copy the entire contents of your LinuxLabs files. Remember, with copy, the first file name is our source (what we are copying from, and the second name is the directory where we want to place the file. Keep in mind that copying files will OVERWRITE files that already exist; but, by using the interactive flag, we can be prompted if this situation will occur and quickly abort if we have made a mistake (with a control c). This means if you place the arguments in the incorrect order, you may lose your original files!

The following flags help us to perform a copy of all the files easily:

- **`-i`** use the interactive option where you will be prompted before a file is overwritten. If you type y (or yes, Y, or YES), then the file will be overwritten if it already exists. Typing anything else will abort the copy operation
- **`-a`** preserving the attributes of directories, ownership
- **`-v`** using verbose mode, explaining what is being done
- **`-r`** copying recursively with any sub directories that might

The arguments begin with the flags, then the source file or directory followed by the destination:

- Make sure you are in the LinuxLabs directory.
- The dot (.) indicates that you want to copy all of the files from the current directory. The wildcard (*) can also be used for this.
- The second directory name (../Lab5_Backup) indicates the destination location. Two dots in a row (..) backs-up one to directory

cp -iar	**.**	**../LinuxLabsBackup**
flags	source	destination location

_____ 9. Type **ls ../LinuxLabsBackup** to confirm that the files are now copied

_____ 10. An alternative approach is to use mcopy. This will select all of the files from a directory and copy them over to another location. In this case, permissions for the newly copied files may not match the original permission. Again the -i can be used to ensure that files are not overwritten by mistake.

Copy the files by typing **mcopy -i * ../LinuxLabsBackup**

Chapter 5: Using gdb Linux Debugger

Exercise #5.1 - Beginning with gdb

As we write programs with more complicated data structures, using gdb is going to become more important. Research and learn how to do the following tasks using “man gdb”; write one sentence about each. Refer to zyBooks Chapter 21.CS 162 Debugging for more information on GDB.

Or you can check out tutorials point. Or here is another great GDB tutorial.

______1. Use **`man`** to look up how to use gdb. Type **`man gdb`**

______ 2. List 3 commands that you think would be useful and write down what you will need to know to use these while debugging a program:

1. ____

2. ____

3. ____

______ 3. Login into the PCC Linux server **`(cslinux.pcc.edu)`**

______ 4. Let’s work in the practice directory **`cd ~/CS161B/LinuxLabs`**

______ 5. You can copy my lab6.cpp. Do

```
cp ~gd.iyer/CS161B/LinuxLabs/lab6 .
```

Or create a program that has some logic errors (not syntax errors). Make sure to:

- Have an infinite loop
- Have a function that returns prematurely.

______ 6. Now compile a program using g++ on the command line:

```
g++ -g *.cpp *.o
```

After going through the gdb tutorials from one of the websites given above, try the following and see if you can find some answers.

______ 7. Locate a Segmentation Fault ______________________________

______ 8. Display the contents of a data member ______________________

______ 9. Backtrace ____

______ 10. Set a breakpoint

Explain how you did this: ________________________________

______ 11. Display the contents of your local variables.

What was the command to do this: __________________________

Chapter 6: Compressing Files and Directories

Exercise #6.1 – Archiving Files

In many courses, you will find that programming assignments must be submitted as tarballs. This is especially necessary when faculty use D2L. Since programs typically consist of multiple files, archiving files together provides an easier technique for managing the software; it is also interesting to note that D2L alters the file names when files are individually submitted. This is a problem when we expect a .h file to have a specific name! These exercises are an attempt to prepare for submitting assignments.

Note:

- Rsync is not installed on the pcc linux server. Please do not try rsync for now.
- Follow the below instructions from step 4.

_____ 1. Login into the PCC Linux server **(`cslinux.pcc.edu`)**

_____ 2. First, make sure to be in the appropriate Lab directory

```
cd  ~/CS161B/LinuxLabs/Lab7
```

_____ 3. We will start by **backing up the files** using **rsync**. *rsync* can be used to copy your files from one directory to another and it can also be used to sync files on two directories on the same remote location. As with all Linux commands, the first path is the source directory and the second path is the destination, where a copy is being made.

The advantage that *rsync* has over copying the files, is that if a newer file exists in the backup folder, it will not be copied over (depending on the flags).

The following are some interesting flags when synchronizing the files:

- -u this will skip any files that are newer in the destination directory so that all of the files are the latest
- -v using verbose mode, explaining what is being done
- -r copying recursively with any subdirectories that might

The dot (.) indicates that you want to copy all of the files from the current directory. The wildcard (*) can also be used for this.

The second directory name (../Lab7_Backup) indicates the destination location. Two dots in a row (..) backs-up one directory

_____ a. Type:

```
rsync   -uvr ../Lab7_Backup
```

flags source destinationlocation

Important!

- Follow these instructions carefully
- Pay close attention to the order of the arguments

_____ 4. Creating **tar files** is also a great way to compress our programs and enables us to submit assignments that contain multiple .cpp, .h, and data files

To create an archive, we first list the flags, then the archive file's name (this needs to be a file that does not exist), followed by a space separated list of all of the files you want to archive together.

The following are some interesting flags for creating archives:

- -c means that we are creating an archive
- -v places it in verbose mode which will display all the filenames that are added to the archive. Pay close attention to the list to make sure that it includes all of the files you wanted
- -f means THE NEXT ARGUMENT is the archive name and NOT one of your source files

For Assignment 7, use:

tar –cvf archive_name **.tar** ***.cpp** ***.h *.txt**

- Do not archive the a.out or other object modules that may exist.

_____ a. Now it is time to practice! Make sure you are in the right folder. When you are archiving a folder, it is important you are in the parent folder of the folder you want to archive. This is very important.

`cd ~CS161B/LinuxLabs`

Start by creating a tarball: **`tar –cvf Lab7_Archive.tar Lab7`**

Important!

- The first file name listed is your resulting tar archive file name
- The tar file name should not already exist - name it something new

_____ 5. Next, make sure the archive is correct. **NEVER SKIP THIS STEP!**

_____a. Now list the contents of your directory to see that a new file should be called *Lab7_Archive.tar*. Type: **ls**

_____ 6. Next, we will unpack the archive to make sure it is correct.

***NEVER unpack in a directory that already has files of the same name that you want to keep. I recommend always working in a new directory!

_____ a. Make a new directory so we can unpack the archive
mkdir Unpack

_____ b. Change into this new directory, type: **cd Unpack**
_____ c. Unpack the archive, type: **tar –xvf ../Lab7_Archive.tar**

_____ d. Examine the files in your Unpack directory.
Are there .cpp and .h files? (type: **ls**)
Are their contents intact? (type: **cat *.cpp**)

_____ 7. Summarize what you have learned:
_____ a. Why is creating a backup important before creating a tarball?

____

_____b. What happens if we forget to place the name of the archive first?

____

_____ c. Explain the order of the arguments for tar:

____