COSC 4740 – Lab01
Accessing COSC Linux Machines
Compiling Hello World

Part 1 – Access Linux machine remotely using putty

1.1 Putty

Follow the steps to access putty and connect to a COSC Linux machine remotely. You will be using these machines for each of your labs.

1. If you are on campus launch putty (Click Start => All Programs => Putty => Putty)
    a. If you are on a personal machine and don't have it installed go to https://www.puttygen.com/download-putty and install it. (Windows 10 works)
2. In the Host_Name box, type "hive.cs.uwyo.edu"
3. Click Open, then Yes when asked to add the RSA key
4. Type your Linux username and press enter
5. Type your Linux password and press enter
6. Take a screenshot of your Linux terminal for submission (named Lab1_1 respectively)
    a. You can use the snippet tool on Windows

Important: If you haven't talked to Jim about your password yet, you will have to finish this lab after you can log in!

1.2 Commands

Type these commands into the Linux terminal: pwd, ls

pwd will print the current working directory the terminal is in.
ls will list the contents of that directory.

1. Create a directory by typing "mkdir cosc4740" (typing ls will show the new directory)
2. Enter the directory you made by typing "cd cosc4740"
    a. Hint: using the tab key can help with auto completion

mkdir creates a new directory at your current working directory (type pwd to see what it is)
cd is used to enter a directory. Typing "cd .." will bring you back out one directory. Typing just "cd" will bring you back to your home directory.

If you want to learn more about some of the basic linux commands go to
https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners

Important: never use **sudo** on COSC machines or Jim will yell and throw forks at you

3. Take a screenshot of your terminal with the commands entered in it for submission. (named Lab1_2)

1.3 Text Editors

You can edit or create files from the terminal using one of several command line editors. Lets use vim.

1. Type "vim test.txt" to open a file called "test.txt"
    a. Note: If the file already exists it will open it, otherwise it will be blank. This does **not** create the actual file until you save it.
    b. To type and code you have to enter insert mode by pressing (i)
    c. To save type ":w"
        i. (:) enters command mode
        ii. (w) writes the file
    d. To exit type ":q" (will **not** save file)
        i. If vim is being stubborn add a (!) to the end for "I really mean it!"
    e. You can combine commands, for example to save and quit type ":wq!"
    f. Pressing (esc) will exit both command mode and insert mode.
2. Another line editor you can use is nano that has the same behavior as vim but is a bit easier to use
    a. Type "nano test2.txt" to open a file "test2.txt"
    b. You can type and code right away.
    c. Commands are listed at the bottom.
    d. Note: (^) means using the CTRL key.
3. Take a screenshot of the terminal with vim or nano for submission. (named Lab1_3)

If you want a more in-depth tutorial on vim see http://www.openvim.com/

1.4 Xming

With putty alone we are only able to use the terminal. If we use Xming along with it we can run GUI applications. If you type "kate" into the terminal and press enter you will find that the terminal will spew out some errors, lets change this.

1. Type "exit" into the terminal to close it and go back to putty.
    a. If it doesn't close just exit the terminal to close the session
2. If you are on campus launch Xming (Click Start => All Programs => Xming => Xming)
    a. If you are on a personal machine and don't have it installed go to https://sourceforge.net/projects/xming/ and install it.
        i. If it failed to install due to permissions, right click it, and select "run as administrator" and do it again.
    b. It will appear that nothing happened, that is fine. We just started a server process in the background.
    c. In your tray at the bottom right you should see the Xming symbol.
3. Type "hive.cs.uwyo.edu" again into the Host_Name box.
4. On putty on the left under the Connection tab, expand the SSH tab.
5. Click on X11 and check Enable X11 forwarding.

If you want to save the current putty configuration you can go back to the Session tab, type a name for the configuration in the box below Saved Sessions, and then click the Save button. To use it again later when you open putty, just select the name from the list and click Load.

6. Start the connection again by clicking Open and login to your Linux account.
7. Type "kate" and press enter. The kate text editor should open.

You can use this to start other GUI applications such as chrome and other text editors. If you want to stop Xming, go to your tray and right click it and then select exit.

Other text editors include: kate, gedit, and atom.

Note: Keep in mind that running GUI applications can be really slow.

8. Take a screenshot of kate running on Xming. (named Lab1_4)

1.5 Git

Now that you know how to connect remotely via putty and have been introduced to the terminal and possible text editors, we will get into how you will work on and submit your labs using Git.

Git is a version control tool that helps contain all code in a central location and allow multiple people to work on it in separate places.

Each of the labs will have a link to allow you to create your repo. All your work will be put here. If you are on your personal machine you can install git here: https://git-scm.com/downloads

1. Use this link to create your repo for lab01: https://classroom.github.com/a/2KCFlPvz
   Note: If you were not prompted to accept the assignment, paste the link into a browser.
   a. The repo (repository) on GitHub is the origin repo where your work resides.
   b. You can edit it over the browser if you wish but it is not recommended.
   c. Instead you can clone your repository to your local machine to edit it there.
      i. In essence you are making a copy of all the files from your origin repo on GitHub and putting them on your machine
      ii. local is relative to where you are cloning the repo to
2. In the browser where your repo is, enter the repo by clicking its name. (osLab01)
3. In the top right you will find a button named Code. Click it to find a GitHub link along the lines of: "https://github.com/YourUsername/YourClassName/YourLabRepoName"
   a. Copy this link. This is the link that will be used to clone your repository from your origin repo to your local machine.
4. In your Linux terminal, go to your "cosc4740" directory you created previously.
5. Type "git clone <repo link here> lab1" (do not include the < >)
   a. This will clone the repository from your GitHub to your local machine.
   b. The repo will be a normal file with the name lab1 (if you type ls you will see it)

Note: If you are entering this link over the putty terminal you will have to copy the link then right click the terminal to paste it.

6. Take a screenshot of your cloned repo for submission. (named Lab1_5)

1.6 Hello World

With your repo now on your machine lets create a hello world program inside it.

1. Enter your repo and create a new file named "lab1.cpp" with your preferred editor.
2. Type the following program

```
#include<stdio.h>
int main(int argc, char** argv)
{
   printf("Hello World!\n");
}
```

3. Save the file and exit the editor.
4. In the terminal, compile your program by typing "g++ lab1.ccp -o hello"
    a. This will compile your C++ code into an executable called hello.exe
5. To run the program type "./hello". Your terminal should print "Hello World!"

If you wish you can redirect the output from your program into another file by using (>). For example, "./hello > output.txt". You can use the cat command to quickly view the contents of a file. For example, "cat output.txt"

1.7 Git Continued

If you look on your browser with your repo you will find that your hello world program isn't there. That is because that file is on your local machine and not in your origin GitHub repo. In order to update your origin repo to include your new file you have to push it from your local machine up to it.

1. In your Linux terminal enter your lab1 repo if you aren't there already.
2. Type "git status"
    a. You will find your new files under "Untracked Files" in red.

Files in Git go through a process to keep track of changes for version control.

Untracked Files are files found by git that are not being monitored, often as such with new files. You can think of them as unofficial files for your repo. In order for Git to begin monitoring the files for changes you need to add them. Adding files is known as staging.

3. To add your lab1.cpp file type "git add lab1.cpp"
    a. Note: You can add all your files using "git add -A" (Don't do it yet though)
4. Type "git status"
    a. You will find that your file is now being monitored and is ready to be committed.

Changes to be committed: If a file monitored by Git has been changed and staged using add, these files will appear here. This is another step in version control as a type of "check" for whether you REALLY want the changes you made to become final.

If you found that you didn't want to commit your changes you can use "git checkout -- <file>" to reset the file to before you made changes to it. This will only reset your **local** changes. More information on undoing changes can be found here https://docs.gitlab.com/ee/topics/git/numerous_undo_possibilities_in_git/

After you are confident you are satisfied with your changes, we can commit the files.

5. To commit your changes type "git commit -m <message>"
    a. The message is used to describe what the changes are. Just typing "git commit" will start a text editor (such as vim) to type a more detailed message.
6. Type "git status"
    a. You will find that your file has been committed and is waiting to be pushed to your origin repo on GitHub.
    b. This is noted in your terminal with "Your branch is ahead of origin/master by 1 commit"

Now that the changes are committed, you can push the changes back to the origin repo.

7. Type "git push" to push your changes.
    a. You may need to enter your GitHub username and password to authorize the changes.
    b. If you look at your browser now you should see the new file (may need to refresh)
8. Edit the README.md file that came along with the cloned repo with your preferred editor.
    a. Enter your name, class section, how to compile and run your program, and anything that isn't working. This will also be where you can find your lab grade after it is done being graded.
9. Type "git status"
    a. You will find git recognized that the readme has been modified.

Changes not staged for commit: If you change a file that was committed (regardless of push) the file will be under this status. From here you will need to stage the file using add, and when satisfied with the changes use commit to finalize as before.

10. Add, and commit the readme file.

GitIgnore: If you find that you have many files that you don't want to push to the main repo but keep on your local machine, such as executable, configuration, or data files, you can create a .gitignore file.

11. Create a new file called ".gitignore" in your repo. (Don't miss the dot)
    a. Here you can ignore specific files by typing in their names, whole directories, or those matching a specific type using REG expressions.
    b. Note: this will not ignore files you have already added to your git repo.
12. Type "hello" so that it will ignore your hello executable.
    a. Typing "*.exe" will ignore all files with the extension .exe
13. Add, commit, and push the file. (This will also push your committed readme file)

14. Type “Git status”
    a. you will find that your hello executable can’t be found by Git (typing ls still shows it)

More on Git

If there have been changes in your origin repo and your local files need updated, you can type “pull” to update.

If you come across a merge conflict there are two possibilities

1. Your local files are out of sync with your origin repo, that is, your origin repo has been updated and your local machine needs to pull in these changes.
    a. Type “pull” after committing your local changes to merge with your origin repo.
    b. Type “push” afterwards to update your origin repo with your merged changes.
    c. This is the majority of merge conflicts
2. The code you modified was modified by another person in a conflicting manner
    a. This can occur when more than one person modifies the same code in the same place but in a different way.
    b. An example is as follow

Bob’s Code	Alice’s Code

print(1);
print(2);
print(3):
	print(1);
	print(1.5);
	print(3);

If either Bob or Alice tried to merge their code together Git doesn’t know how to do this as the second line of each of them conflicts with each other. When this happens Git will merge them together and the person merging the code can choose which changes to keep

Git generated lines

```
print(1);
=> Git ================== FIX IT!!! =============
=> print(2);
=> Git ================== Choose one ===========
=> print(1.5);
=> Git ================== FIX IT!!! =============
print(3);
```

After removing the generated lines made by git and choosing one or both of the changes you can push it as normal

Pushed Code

```
print(1);
print(1.5);
print(2);
print(3);
```

1.8 Submission

1. A total of 5 screenshots pushed to git. (you can add them through your browser)
2. Your hello world program and .gitignore file pushed to git
3. Your edited README.md file pushed to git
4. No other files should be in your origin repo

Part 2 (Extra - FYI) – Access Linux machine via on-campus windows machine using VM

1. Launch Oracle VM VirtualBox Manager (Start => Oracle VM VirtualBox Manager => Oracle VM VirtualBox Manager).
2. Select the Machine => Add… menu option
    1. Navigate to C:\linuxvm\Rocky
    2. Select the "Rocky" file (VirtualBox Machine Definition)
    3. Click open
3. You should now see the Linux 4740 virtual machine available in the left panel. With this machine selected, click the large green "Start" button on the top menu bar.
    1. The machine will now boot a Linux operating system just like the others used in the COSC department.
4. Once booted, you will be asked to login. Provide your Linux username and password
5. To open up the terminal click on the Applications menu in the top left corner of the screen, go to System Tools, and launch the Terminal application.
    1. Commands work the same as mentioned above in part 1.
6. Here you have some options for text editors, you can try one or both of these methods, and see which you prefer. From the Applications menu, under Accessories
    1. Launch the application called Kate
        A. Click Settings->Configure Kate
        B. Select Editing on the left
        C. Select the Indentation Tab
        D. Select default indentation mode: C style
        E. Change Indent Using to Spaces, 4 characters
        F. Select Plugins on the left
        G. Check Terminal Tool View
        H. OK
        I. Click File => Save As to save the current file
        J. Do some code-fu
        K. Save the file.
    2. Launch the gedit application
        A. Click File->Save As to save the current file.
        B. Do some code-fu
        C. Save the file.

Regardless of which text editor you use you can always compile and run it in the terminal.

Note:

There is a linux VM on each of the Lab machines in 4059. There are also a set of Linux machines in the Undergraduate Computer Lab (4072). The undergraduate lab also has the same VM on each of the windows machines and you can access it the same way you did here. This will allow you to write, compile, and test your code for labs and projects!

You can access and edit files that exist on you Linux home directory by navigating to \\alameda\<username> from windows explorer (on campus computers). You may want to use

this technique if the X session is slowing down your work! You can edit your files from the Windows machine, and switch to the terminal session to compile and run.

Be aware, however, that you may run into issues with end-of-line markers, since Windows and Linux use different representations!

Part 3 (Extra - FYI) – Setting up a personal Linux environment using VirtualBox

If you find that working on the COSC machines is too slow or inconvenient you can setup your own Linux environment on your personal machine.

1. If you don't have Oracle VM VirutalBox installed you can install it here https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html
2. Click this link https://www.cs.uwyo.edu/~seker/
3. At the bottom click "UWcosc ova file"
    a. This will be the file your VirtualBox will use as a VM
    b. Keep in mind this file is quite large (~5.4 GB)
4. In VirtualBox click Import
    a. For the file, select the .ova file you downloaded previously
    b. Select next
5. Select import on appliance settings
    a. After it is finished being imported you can start your Linux environment.
6. Select your VM on the right and click Start
    a. Wait for it too boot up
7. Select the cosc account and enter the password found where you downloaded it (cosc)
    a. **Note**: this account has sudo (admin) privileges.

You now have your Linux environment setup. This environment should contain all the necessary tools you need to do your labs such as Git, and even tools for compilers if you are taking that class. At any time you can close and shutdown the VM and start it back up. You can also pause your machine at any time and come back to it.

If you want to share your clipboard with your host machine you can install the tools to do so

8. Click Devices at the top
9. Select Guest Additions CD Image
10. Click Run and enter your password
    a. This can take a moment, and you will be asked to restart at the end.
11. In your VM manager select your VM and click Settings
12. Click the Advanced tab
13. For Shared Clipboard select Bidirectional to have your clipboard share both ways.
    a. You can do the same for Drag-N-Drop for files if you wish
    b. Keep in mind that this may not work depending on the host machine.
14. Click OK to finish.

Note: Make sure that when you are using this VM you remember to push your code onto Git and ensure that it compiles and runs on the COSC Linux machines.

Summary of mentioned links and commands

Links

Putty Download: https://www.puttygen.com/download-putty
Xming Download: https://sourceforge.net/projects/xming/
Git Download: https://git-scm.com/downloads
VirtualBox Download:
https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html
Linux Commands: https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
Vim Tutorial: https://www.openvim.com/
Git Undo: https://docs.gitlab.com/ee/topics/git/numerous_undo_possibilities_in_git/
UW OVA File: https://www.cs.uwyo.edu/~seker/

Terminal Commands

- pwd: List current directory
- ls: list items in current directory
- mkdir: create a directory
- cd <directory>: enter directory
    - use .. to go out once, add \.. to go out more than once
    - cd alone will put back in your home directory
- cat <file>: list file contents to terminal
- g++ <file> -o <name>: compiles the C++ file into an executable with the given name
- **Note**: Remember to not use **sudo** on COSC machines.

Git Commands

- clone <repo-link> <name>: Clone a repo to your local machine with the given name.
    - repo-link can be found online at your repository
- add <file>: Stages the given file to be ready to be committed
- commit -m <message>: Commits the staged files
- push: Push your committed local files from your machine to the master repository
- pull: Pull your files from the master repository to your local machine

Vim

- (i) enter “insert mode” to type
- (:) enter “command mode”
- (esc) exit “insert/command mode”
- (w) command to write the file (save)
- (q) command to exit vim (won’t save)
- (!) “I really mean it!”