**Lab 02 - Server Install and Linux File System**

# Introduction

This week we will be working with a headless server (versus a GUI) and will learn the basics behind Linux file structure, commands, processes, and redirection, and will cover some important files and their uses.

GUI vs Headless

A GUI, or graphical user interface, is a program that facilitates interaction between a user and underlying software. Last week’s Ubuntu Desktop had a GUI that allowed the user to use the mouse to click and drag, had colorful displays, and was generally tailored for ease of use.

A headless server does not contain a GUI and is typically interacted with via a textual program, such as a shell/terminal. These types of systems are designed to provide services to users in ways other than most typical desktop applications. Think email servers, website hosting, cloud storage, etc…

While interacting with a headless server can be daunting, a headless server is not bloated with unnecessary programs or burdened with rendering graphics, allowing resources to be used otherwise. In addition, from a security perspective, the fewer “moving parts” exposed to the world, the easier a system is to secure.

# Server Setup

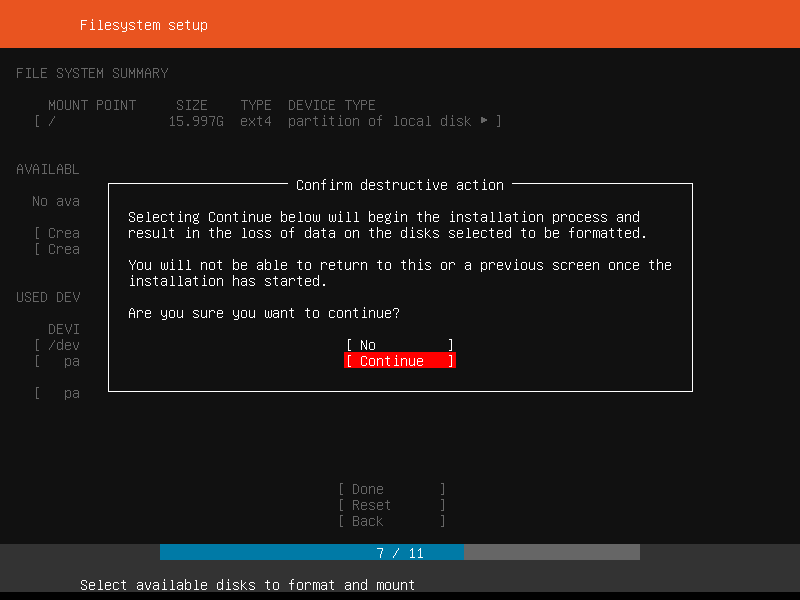
This week you will be installing the server version of Ubuntu. The main difference between this version and the desktop version is that the number of pre-installed programs is significantly less, and there is no GUI - all interaction must be done via the terminal.

1. Refer back to “[Lab 1 - Creating a New Virtual Machine](https://docs.google.com/document/d/1qArbl4VniXLPgfs719zW4fH0Kzjp7ZsQEdegwstF-wA/edit#heading=h.4d34og8)” to provision the server in vSphere.
   1. You should name your machine in vSphere similarly to in lab 1. We will use this machine as a DNS nameserver in a later lab, so it should be named as follows substituting your netid:

cpre230\_*netid*\_ns1

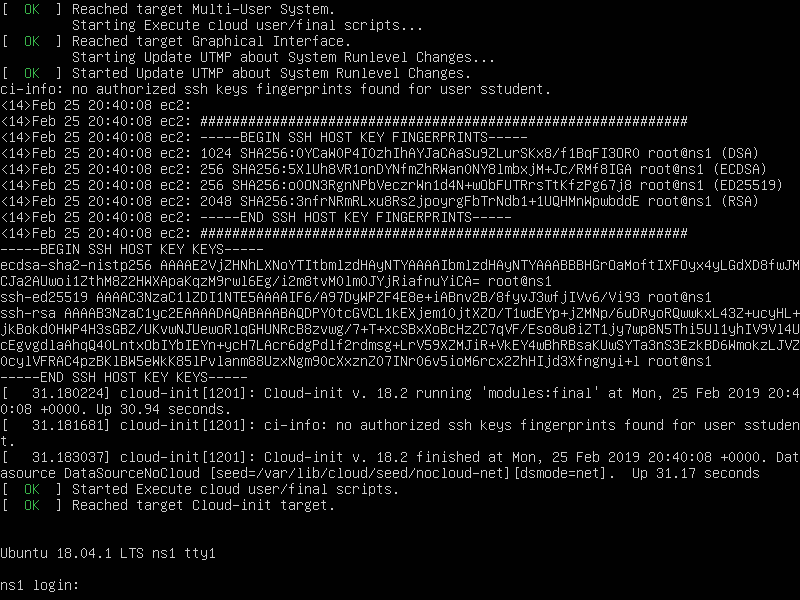
* 1. Hardware requirements:
     1. 1 CPU
     2. 1GB memory
     3. 16GB hard disk
  2. Be sure to select “Thin Provisioning”
  3. Add the same Configuration Parameters as in the previous lab.

1. Refer back to “[Lab01 - Installing the Operating System](https://docs.google.com/document/d/1qArbl4VniXLPgfs719zW4fH0Kzjp7ZsQEdegwstF-wA/edit#heading=h.2s8eyo1)” to start the installation process
   1. Use the “ubuntu-**22**.04-live-**server**-amd64.iso” disk image.
   2. Use the keyboard (arrow keys, tab, space, and enter) to follow the onscreen prompts. Defaults should be used, **except**:
      1. This time, instead of setting up the network connection during installation, we will select “Continue without network”. Additionally, you will continue without setting up the proxy, and continue without updating.
      2. After you choose filesystem options, you will be prompted to confirm the destructive action. This warning is saying that the hard drive for the virtual machine will be cleared ("formatted"), which is exactly what you want to do.



* + 1. On the "Profile setup" step, enter your name, "ns1" as the server's name, your NetID for your username, and the password as “cpre230”.
    2. When you finish the setup wizard, wait for the installation to finish. Reboot the system when prompted. When it asks you to remove the installation medium, right click on the VM in vSphere, click Edit Settings, and set the CD/DVD drive to “Client device”. Then return to the VM and press enter.

1. After your machine has rebooted, you will see some initial setup log messages hiding the login prompt. Simply press enter to show the login prompt after it seems that no more log messages are printing.



1. Log in with the credentials created during the setup. Type your username, hit enter, and then enter your password. Your password will not be shown to the screen, but it is still being entered.



1. To set up your network, open /etc/netplan/00-installer-config.yaml in your favorite text editor (using sudo). The file will look similar to the file from the previous lab. Refer to the [network information document](https://docs.google.com/spreadsheets/d/1R5J2njtK2FHUmtCnFyUdIA1zNvSklBBygKaYGKDfSrw/edit?usp=sharing) that you used in lab 1 for your network range.

* Address is XX.XX.XX.200 of your IP range.
* Default route is XX.XX.XX.254 of your IP range.
* Name servers will be 199.100.16.100.

1. Refer to [Lab 1 - Configuring Network Settings](https://docs.google.com/document/d/1qArbl4VniXLPgfs719zW4fH0Kzjp7ZsQEdegwstF-wA/edit#heading=h.17dp8vu) steps 9 through 11 to set up the proxy and update the system. open-vm-tools is installed by default with Ubuntu Server, so you don't need to do step 12.

# Navigating the Linux Filesystem

## Overview

When using Linux, it is important to understand the filesystem structure that is being used. In most cases, it is useful to think of the filesystem as a “tree”, with one base directory (folder) called “root” from which all other directories branch out. For example, when you log into your account on a Linux system, you are placed in your home directory which is located at “/home/username”. Here, the first “/” represents the root directory which contains the “home” directory, which then contains the home directories of all users.

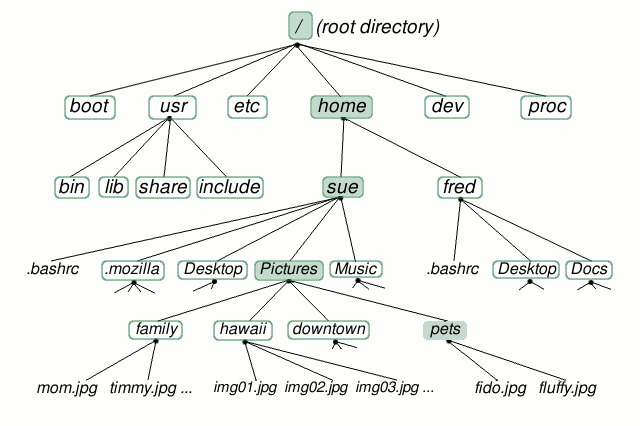


Image courtesy of: [linuxplanet.com](http://www.linuxplanet.com/imagesvr_ce/linuxplanet/1234407730filesystem.gif)

## 

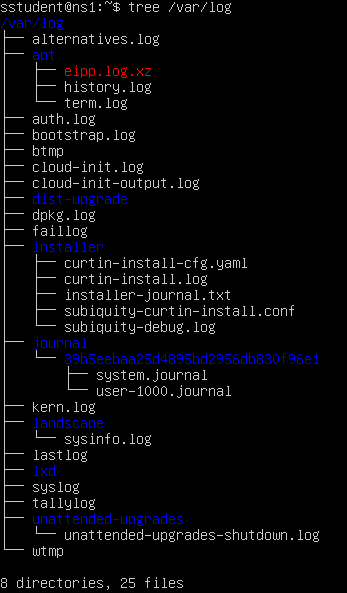
## 

## Basic Commands

### Tree

tree is a useful command for viewing the hierarchy of a specified directory. Install it with sudo apt install tree and run it with tree /var/log to see the directory structure of the /var/log directory.

Note: In general, if you see any errors after trying to install software, try running, “sudo apt update”. It is very common that if software is not installing, it boils down to either a network error, a simple update, or a time setting mismatch.



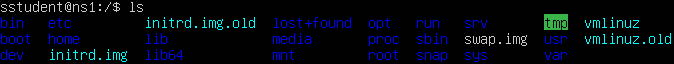
### Print Working Directory

To print exactly what directory you are currently “in”, you can always run the pwd command.



### List Directory Contents

To show the contents of a directory, you can use the ls command. ls can run with and without an argument. Without an argument, ls will show the contents of the current directory. By specifying the path to a directory, you can list the contents of other directories.



### Change Directory

To navigate the file system you can use the **cd** command. cd can be used to **c**hange **d**irectories. It can be run with or without an argument. Running cd without an argument will return you to your home directory. When provided an argument, cd will take you to the specified directory. Arguments can either be a full path (“/home/sstudent/Documents/”) or relative (if you are already in “/home/sstudent” and want to move to “/home/sstudent/Documents/pets”, you only need to run “cd Documents/pets”). To move “up” one directory, run “cd ..” This can be repeated to move “up” through the filesystem, and then specifying directory names to move “down”.

## Practice - Navigation

Combine what you have now learned to

1. print the working path after logging in,
   1. You can logout by typing exit and login again.
2. navigate to the "root" directory, and
3. print the contents of the "var" directory while remaining within the "root" directory.

You should submit **one screenshot** showing the output of these three tasks. If you want to clear the screen, use the "clear" command.

## Directory Management

### Making a Directory

An important ability is the creation of directories in order to keep documents and configuration files organized. To create a directory (folder), use the mkdir command. Unlike the commands we’ve learned about so far, this command requires an argument - the name of the directory to be created.

### Removing a Directory/File

To remove a directory, you can either use the rmdir <name> command to remove an empty directory, or the rm -r <name> command to remove a directory and all of its contents. rm can also be used to remove individual files (without the -r flag).

### Copying a Directory/File

The cp <source> <destination> command can be used to copy a directory or file to a new location. <source> is the name of the object to be copied, and <destination> is the location to where it should be copied.

### Moving/Renaming a Directory/File

The mv <source> <destination> command functions exactly like cp, except after the copy is completed, the <source> object is deleted, thus performing a “move”. mv can be used to rename an object too by “moving” to a non-existent <destination>. <destination> must either be

1. an existing directory, into which the <source> item will be moved or
2. a non-existent directory/name which the <source> item will be renamed to be.

## Practice - File Management

1. Ensure that you are in your home directory ("/home/<username>").
2. Create the following directories: "Documents", "Pictures", "Music", "bin".
3. Within "Pictures" create the directories: "pets", "report", "catz".
4. Move the "report" directory to "Documents".
5. Rename the "catz" directory to "kittens".

Submit one screenshot of the final hierarchy of "/home/<username>" as displayed with tree.

## Manipulating Files

### File Editors

There are multiple ways to edit files in a command line. Up until now, we’ve been using vim and nano to do most of the file editing (vi is generally an alias of vim). vim is a very popular and powerful editor (and does some text coloring that can help you identify typos!), so it is recommended that you at least become familiar with it. Throughout the rest of this course, editing configuration files in the terminal will be very important, so choose an editor and learn how to efficiently use it (practice). Some beginner tips for vim have been compiled and are recommended reading if you want to become familiar with vim.

[vim/nano basics](https://docs.google.com/document/d/1cQw34n61z04T3WEv_oWglbYFKP6zt0-JOKHwxiXIgp8/edit?usp=sharing)

To take a more in-depth look at vim, run the command from the terminal: vimtutor

Take a moment to create a few files, practice opening, saving, editing, and closing files. You are encouraged to search online (read: Google) to help your exploration; however, feel free to ask your TAs questions when you get stuck.

### File “Pagers”

If you want to read a file without accidentally modifying its contents, programs called “pagers” can be used. Two commonly used pagers are less and more (more is the original program, with less being its successor - “less is more”). Open one of your previously created files with less and navigate up and down with the “j” and “k” keys (same key bindings as vim). “q” (short for quit) will close the file, without editing it. "G" and "g" navigate to the bottom and top of the file, respectively. Pagers are useful for reading documentation shipped with many server applications.

less my\_file.txt

### Cat and Redirection

Sometimes you just need to take a quick peek into a file, or pipe the contents of a file into another file/program (more on that in a moment). cat (short for "concatenate") is perfect for this.

Print the contents of some of your previously created files to screen. Notice that once the program ends, you are back in the terminal. All cat has done is to dump the text through "standard out" (a "pipe") that programs dump to when printing output. Typically, "standard out" is directed to the terminal, thus why the text ended up being printed the way it was. However, pipes can be used to redirect data to other files and even other programs.

We can direct the output of a program to a file via the ">" and ">>" operators. As an example, run cat on one of your previously created files along with the ">" operator, as shown below.

cat my\_file.txt > new\_file.txt

Now open the new\_file.txt. You should see the contents of the original file in the newly created file. Also, note that nothing was printed to the screen. Instead of dumping the file contents to the display, standard out was redirected to a file.

Experiment with ">" and ">>" and **explain the difference** between them, with screenshots.

Here, it should be noted that when a program runs, it has three pipes automatically assigned to it: standard in, standard out, and standard error. To pipe data from one program's "standard out" to another program's "standard in" (instead of storing output to file as with ">" and ">>"), use the "|" operator (above the “Enter” key on the keyboard). The "|" character is called "vertical bar" or "pipe". To demonstrate this, execute the following command.:

cat my\_file.txt | less

Although it appears as though the file was just opened with less, it was instead opened by cat, its output piped through standard out, then redirected into the standard in of less. less then displays the content to the screen. Although useless in function, this demonstrates the [power of connecting basic Linux utilities together to create “pipelines”](http://www.linfo.org/pipes.html).

## Man Pages

An important feature of Linux is the inbuilt manual. Using the man command, you can read documentation about almost any command/service/utility available in Linux. Simply execute man with the utility you want to learn about as its arguments. For example, to read more about the less utility, execute

man less

Manual pages are opened in less (ironically enough), so navigation should be similar to that used in vim.

Use man to report on how to search for keywords in less. This should be included in your lab report.

Using the man command, figure out what which, head, tail and grep commands do. Include this in the lab report.

## Common Files

Now we are going to take a look at some common files. First print the contents of **/etc/passwd** to the screen using cat. This file stores users on the system, and in the past it stored password hashes. Notice you see that your account in this file along with multiple others. Your home directory is specified here (/home/<username>) and so is your default shell (/bin/bash).

Next print the contents of **/etc/shadow**. It won’t work as any user except root. Use sudo and cat to print the contents of /etc/shadow (combination of man and Google may help here). Notice that your user has a long string of random characters assigned to it - this is a salted hash of your password. We will cover this in greater detail in future lectures, but it's good to know that user password hashes are stored in the shadow file.

Another important directory to know is **/tmp**. This directory is used to store temporary files and is cleared periodically by the operating system. This space is used as “elbow room” for many applications. It would be unwise to place anything here that you wouldn’t want to lose.

Another useful directory is **/var/log**. This is where logs are generally stored, and this directory will be beneficial to remember throughout this semester when you run into problems. Run tree /var/log again and examine the output. You can see that many files exist with the .log extension. These can be used to examine the status of a system. For example, **/var/log/auth.log** lists all authentication attempts and the results. **/var/log/kern.log** lists kernel messages since system startup. As applications are installed and configured, more log files will appear. When experiencing unexpected behavior of a service or program, check in /var/log for corresponding log files. tail, grep, and less are particularly useful for reading log files.

Take a screenshot of the last 10 lines of the **auth.log** file using one of the commands learned in this lab and include it in your lab report.

# Processes

## Overview

Now we are going to quickly touch on processes. For this class, it is only important to understand how to view which processes are running and how to stop them if need be. The first command is ps(process status). This will list processes by PID (process ID).

Using **man**, figure out what option needs to be used with **ps** to view every process on the system. Include this command and a short description of what it does in the lab report.

## Killing Processes

When a program is running in the background or as a daemon (not attached to a terminal), it may be necessary to kill a process when the traditional Ctrl-C is not available. Via the kill command, processes can be killed by their PID. Usage is as follows.

kill <PID>

This will stop the command that is running with the specified PID. Note: PIDs are always unique to a process. You can also use the -KILL flag to force kill something if necessary (more on the man 7 signal page).

kill -KILL <PID>

Finally, use the top command to view running processes and their CPU usage. From here, it is possible to see the status of various processes and even kill them. Press "q" to exit top.

Submit a screenshot of the output displayed by top. What is the name of the process running at PID 1? Include these two things in your lab report.

## Foreground and Background

As you might have noticed, in the current terminal you are only able to execute one command at a time, or so it may seem. When you start a process in the terminal, it is run in what is called the foreground. This is the process that is immediately running and writing to stdout (standard out) and reading from stdin (standard in). Only one process is able to be run in a terminal’s foreground at a time. To circumvent this, it is possible to run commands in the background. Processes running in the background still write to stdout (unless told to redirect somewhere else at execution), but allow control to fall back to the user to continue executing programs in the foreground.

To execute a command in the background, start the program as usual, with a "&" at the very end. The program will begin to run, printing to the screen as usual, except you will still be able to interact with the shell as usual.

To demonstrate this, run the following command.

ping XXX.XXX.XXX.254 -i 5 &

This will start ping in the background, which will ping your gateway every 5 seconds. Now try to execute the following command and see what happens

ls

Notice that ping will still continue to print to the screen, however you are still able to run commands as usual. In some imaginative world, this might be useful in helping to keep you on top of your server’s uptime (note: this method is not recommended). However, this exact arrangement is probably not ideal. Try to kill the ping process via Ctrl-C. Does this work?

Processes in the background are not able to be directly killed. Rather, we need to either directly kill the process (via kill as mentioned above), or switch the process into the foreground.

Each terminal keeps a list of processes running in the foreground and background. Execute the command jobs and notice that ping is labeled as [1]+ This allows us to directly interact with this "job". Run fg %1 to switch job 1 (ping) to the foreground. Now try to kill via Ctrl-C. ping should now be killed (run jobs again to see this).

This is very useful for running multiple processes at the same time, especially if they do not directly output to the terminal. As a note, jobs that are already running can be switched to the background by first suspending by hitting Ctrl-Z, then sending to background via the bg command (format similar to fg):

ping XXX.XXX.XXX.254 -i 5

*ctrl-Z*

jobs

bg %<job\_id>

Think of a way that would allow you to run ping in the background, yet not print the output to the screen. Include your solution in the lab write-up. (Hint: There is a flag in the ping man page which can help accomplish this)

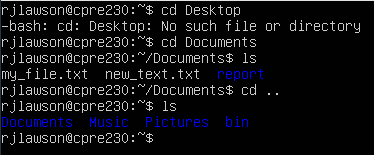
## Changing the Ubuntu Server Resolution/Font Size:

You may have noticed that the terminal for the ns1 machine is small, and sometimes hard to read. If you wish to change the resolution to make your server’s screen larger, follow the instructions listed [here](https://docs.google.com/document/d/1B-unFEVdGFBygXWcIKmA7A4JOzphJ9DERjnhytVvAa8/edit?usp=sharing).

# Lab 02 Template

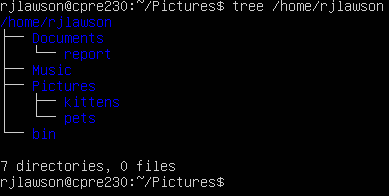
1. **Screenshot of cd/ls practice**

(10 points)



1. **Screenshot of directory structure**

(10 points)



1. **Difference between > and >>**

(10 points)

The difference between > and >> is that > replaces the file text with the current file. The >> just adds the file information to the other file

1. **How do you search in less?**

(10 points)

To search in less you use forward slash and typing what you’re looking for.

1. **What do which, tail, head, and grep do?**

(10 points)

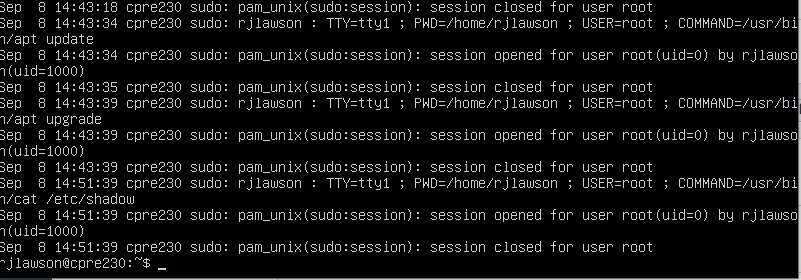
which returns the pathnames of the files that would be executed

head outputs the first part of the file

grep searches for patterns in each file

1. **Screenshot of /var/log/auth.log**

(10 points)



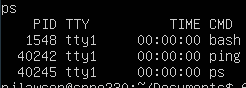
1. **How does one view all processes using ps?**

(10 points)

**ps -all**

1. **Screenshot of top output, name of process with PID 1**

(10 points)



1548

1. **Method of running ping in bg without output on screen**

(20 points)

