**Linux Ubuntu Notes**

**Linux Overview**

What is a distribution?

Debian one of first major Linux distributions.

RedHat and SUSE are also major distributions.

They are called Linux distros

A distribution is a process of delivering software from a developer to the end user.

A Linux distribution is an operating system made from a software collection that includes the Linux kernel.

A kernel is a system software that is part of the operating system.  It is the core that provides basic services for all other parts of the Operating System.

* The kernel is the lowest level of the operating system.
* It is responsible for translating user commands into machine language.
* Provides interface between applications and hardware
* Responsible for memory management, disk management, process management, and task management.
* First program to load when operating system loads.
* Translates between software and hardware

An Operating System manages all the software and hardware on the computer.

* Provides interface between user and hardware.
* First program to load when machine boots up.

Linux commands are written in terminal which is the same as the “Shell” or “Command-Line”

The default shell for Linux is bash

**sudo**

Anything that changes system (i.e. installing and removing applications) requires administrative privileges.

* You have to use ‘sudo’ in the command line.

Files with .run files extensions can be run directly in the terminal by typing in the file name

* You must use ./ to indicate that the file/program you want to run is in the current directory.

**GTK**

GTK is the theming engine that Unity uses (the Desktop environment).

We can get different themes from gnome-look.org

NOTE: Always take snapshots of virtual machine when doing something risky.

**Apt-get**

apt-get is the program that installs, uninstalls, updates, and manages software packages

* Must run as administrator (use sudo)
* It connext to the Advanced Package Tool (APT) library

Unity-tweak-tool is used to change desktop and icon theme.

* sudo apt-get install gnome-tweaks

Any file or directory that begins with a ‘.’ Is a hidden file/directory.

* To access press control+H

To install new theme, we must create hidden directory

* Common practice names it ‘.themes’

Linux File System is like a tree of directories

/ is the root directory where all other directories branch out from

~ is the home directory

* You can use ‘cd ~’ to go to home directory as well as ‘cd /home/handle’ and ‘cd’

The shell prompt allows you to explore file system using the ‘cd’ command

* The default prompt for Linux is a ‘$’
* Directly preceding the prompt is the location you are currently located in a file system

The shell is a computer program that allows you to use operating system’s services.

* The default shell for Linux is bash.

/usr contains users

* /usr/bin contains binary files that you can run
* /usr/share is used to store configuration files for binary applications
* /usr/share/applications contains all installed applications
* Use .desktop file extension, but you cannot see file extension.
* If you cannot find an installed application in your applications dashboard, you can go into this directory and find it there.

**pwd**

pwd (print working directory) command allows you to print the current working directory you are in in terminal.

**cd**

cd (change directory) allows you to navigate what working directory you’re in

* If you specify a directory starting with ‘/’ then this means that the path is an absolute path from the beginning of the hard disk/root directory.
* If you omit the ‘/’, this means you are using a relative path and you are going another level directory from the current directory you’re in.
* The ‘./’ character combination means the current working directory.
* Cd next level And cd ./next level may have same result if they both start form the same directory.
* The ‘~’ character indicates the home directory
* cd ~ will take you to the home directory.
* The ‘..’ character combination means the parent directory
* cd .. will take you to parent directory

**ls**

The ‘ls’ or ‘l’ command will list the files in the working directory in alphabetical order.

* ‘-l’ option means ‘long’ and allows you to list files in directory in a ‘long’ way. Lets you see
* Permissions of a file or directory
* The owners user account
* The user group (default group is owners user name)
* This will let you see the owners of files.
* Shows date modified
* Shows file name
* ‘-r’ option lists files and directory in working directory in reverse alphabetical order.
* ‘-a’ means list hidden files as well.
* ‘-p’ option lists file types for files and directories.
* ‘/ ‘ as a suffix denotes a directory
* Blue color by default
* ‘.file\_extension’ denotes a file.
* White color usually
* Binary files (I.e. a.out) are green
* ‘-s’ option shows size of files and directories in current directory
* Sorts in descending order by block size
* ‘-lh’ option lists the size of the items in directory, but explicitly states sizes in kilobytes, megabytes, etc.
* If you want to list the contents of a directory that is not the working directory you can provide the path to ls to print those contents. For example:
* ‘ls /usr/share/applications’

You can use multiple options in the same ls command.

-- help option will give you more info on any command you are using such as what it does and available options

sudo command means ‘superuser do’.

* Allows you to perform administrative tasks
* Make changes to files

nano is text editor in terminal that allows you to edit files.

* To save changes you can press ‘ctrl + O’
* To exit file you can press ‘ctrl + X’
* Default for Debian distributions of Linux

!! means to run the previous command.

su command means ‘switch user’

* You would write ‘su accountName’

sudo su will switch you to the root account

* Terminal will look like root@ComputerName, where root is the current user.
* This allows you to do administrative tasks without writing sudo
* Command ‘su username’ will switch you to other user.

You only have permission to edit files in ‘Home’ directory, any other directory requires administrative privileges – have to use sudo.

The package manager for Debian/Ubuntu is ‘Aptitude’ and we use it through program apt-get.

Program ‘apt-get’ is used to install/uninstall applications/packages from APT (Advanced Package Tool) library.

* The command ‘install’ means you want apt-get to install program.
* Command ‘remove’ means you want program removed.
* Command ‘upgrade’ will update any existing packages that are in system that have been updated in APT library

Must use sudo for anything that changes administrative directory content.

Program ‘apt-cache’ is used to display information about packages inside APT’s internal database.

* Command ‘search’ of apt –get will search packages in APT db
* I.e. Comannd ‘apt-cache search name\*’ will display all pacakages beginning with ‘name’.
* Command ‘policy’ will determine if certain package is installed or not.
* ‘apt-cache policy’ will get version of package installed on machine.

**dpkg**

If you want to install debian package that is not in APT library, you can download the file and then run ‘dpkg’ program with the ‘i’ or install option.

* ‘-I' means ‘install’
* For example, ‘sudo dpkg -I ./google-chrome.deb’ command will install google chrome debian package in current directory
* Must be debian package, e.g. ‘.rpm’ files not allowed
* This is for Fedora and RedHat systems.

To create text file with administrative permissions, you must use sudo command and nano, I.e.:

‘sudo nano filename.txt’ command

If we type ls –l and list files, we will see ‘filename.txt’ is owned by ‘root’ user and a part of ‘root’ group.

If you want to allow this file to be edited by other users, you can change the user group to the user group you want to allow to write to the file, you can do this with the ‘chown’, I.e:

‘sudo chown user:group filename.txt’

* ‘chown’ means change owner and changes ownership of file/directory
* ‘group’ is the user group.
* Note: /etc/group directory contains all groups defined in system
* Note: you can use ‘groups’ command to find all groups you are a member of.
* And ‘user’ is the user who owns the file
* If you only want to change the user who owns the file you can use:
* ‘sudo chown user filename.txt’

Then you must change the permissions of the file for the specific group you want to grant access to

* i.e. ‘sudo chmod 764 file.txt’
* ‘chmod’ means change made and grants access permissions to specific users
* This give rwx permissions to the user user
* ‘rw’ permissions to the user group
* And ‘r’ permissions to others
* The permissions of the file are indicated by the first 7 dashes after executing ls -l
* i.e. -rw-r-- r--
* The first '-’ indiciates the object is a file
* If it were a directory it would be a ‘d’
* ‘r’ means read
* ‘w’ means write
* ‘x’ means execute
* The three dashes after first bit are the permissions for the user
* The following three dashes are the permissions of the group
* The last three dashes are the permissions for the world.
* You can change file permissions numerically, you must use: ‘sudo chmod XXX file.txt’
* XXX is a 3 digit octal number
* The first digit is the permissions for the user.
* The second digit is the permissions for the group.
* The third digit is the permissions for the world.
* 0 –  no permissions
* 1 – execute
* 2 –  write
* 3 – execute + write
* 4 – read
* 5 – read + execute
* 6 – read + write
* 7 – read+ write + execute
* For example ‘sudo chmod 777 file.txt’ gives read, write, and execute permissions to all users of the file.
* You can change file permissions symbolically as well for individual owners
* Letters represent the specific user(s)
* ‘u’ is the user/owner
* ‘g’ is the user group
* ‘o’ is other/the world
* ‘a’ is all owners
* Mathematical symbols are used for a specific operation made to a file or directory
* ‘+’ adds permission(s)
* ‘-’ removes permission(s)
* ‘=’ sets permission(s) and overrides previously set permissions
* The permissions you can set are
* ‘r’ - read
* ‘w’ - write
* ‘x’ - execute
* For example:
* ‘sudo chmod g+wx sudotemp.txt’
* This adds write and execute permissions to ‘sudotemp.txt’.

Note:

* You cannot have nested groups
* One file/directory cannot belong to more than one group
* Execute permission on a directory means being able to enter it and its subdirectories

If you want to remove object such as file, directory, or symbolic link from directory, you can use the ‘rm’ command

* ‘rm’ means remove

**‘rm’**

Used to remove file

* r (recursive) lets you remove files, directories and its contents (including subdirectories)
* rf lets you remove files, directories and its contents (including subdirectories)
  + ‘f’ option means force, so command will not prompt before removing object

**mkdir**

‘mkdir’ command makes a directory, I.e.:

* ‘sudo mkdir mydir’
* Will make a directory with administrative permissions.
* All permissions are given to root user
* User group only get ‘rx’
* Other only gets ‘x’
* ‘x’ or execute means they can enter the directory and its subdirectories

‘sudo nano ./mydir/file.txt’ will create file in mydir directory with administrative priveleges.

* As you can see, you can specify a path to nano text editor on where to create file.

‘chown’ and ‘chmod’ command has –R option which mean ‘recursive’, meaning it affects subdirectories and files as well, and their subdirectories and files, etc. For example:

* If we use 'chown’ on directory that means that not only will it affect the directory, but also all the files and subdirectories in it as well, including their subdirectories and file, etc., for example
* ‘sudo chown –R  dan:dan  ./mydir’ will change the user and group of ./mydir and all the files and directories it contains, and their files and directories.

**rm**

You can use the rm command to remove files, but also directories

* ‘-d’ option removes empty directories
* ‘-r’ option removes non-empty directories
* ‘-rf’ option also removes non-empty directory, but ignores prompt when deleting write protected file
* We can use ‘\*’ before file extension to remove multiple files of the same type.
* ‘rm \*.txt’
* We can use ‘\*’ after extension to remove file with specific name
* ‘rm file.\*’
* We can ‘\*’ by itself to remove all file
* ‘rm \*’

**touch**

If you want to create a file without opening a text editor, you can use the ‘touch’ command, for example:

‘touch <file1> [file2[file3[...]]]’

or

‘touch file1.txt file2.txt file3.cpp file4.cpp main.cpp’

‘../’ and ‘..’ both go back one directory.

**cp**

‘cp’ command is ‘copy’. Copies a file to a specific location, I.e.

‘cp filename [path]filecopy’

**mv**

The ‘mv’ command move a file or directory from one location to another

* If you move the file to same location, but with a different name, then the name of the file changes, I.e.
* ‘mv file.txt filerenamed.txt’
  + - You can omit the .txt extension in Linux dists because files are automatically saved as plain text.
* Or you can specify a path and the file to another directory, I.e.
* ‘mv file.txt newlocation/’

**find**

If we want to find file in a directory we can use the ‘find’ command, the syntax is:

* ‘find directorypath –type f –name “filename”’
* i.e., ‘find . -type f –name “\*.php”’
* This looks for all files with .php extension
* ‘directorypath’ is the path of the directory you want to search
* i.e., ‘.’ means the current directory
* ‘-type f’ denotes the type of object you want to look for.
* ‘f’ means to look for a file
* ‘-name “filename”’ indicates the name of the file you want to look for
* “file.txt” means to look for ‘file.txt
* “\*.txt” means to looks for any files with the ‘.txt’ extension
* Note: “filename” is case-sensitive.
* You can use ‘-iname “filename”’ instead of ‘-name “filename”’, this removes case-sensitivity for “filename”
* You can use wildcard for ‘-iname’ and ‘-name’ options to look for multiple files.
* i.e. ‘... -name “file\*”’
* ‘find’ is a recursive command, so if we use it a directory with subdirectories, it will search those as well,
* i.e., ‘find /etc/ -type f -name "\*.conf"’
* This we find all files ending in ‘.conf’ in the /etc/ directory and its subdirectories
* We can omit the ‘-type f’ and this will look for all files and directories in the current directory and its subdirectories.
* Results include the directory you’re searching.
* You can also specify the ‘-perm’ option to search for files and directories with a specific permission
* i.e. ‘find . -perm 0664’ or ‘find . -perm 664’ will find all files and directories in the current directory with ‘664’ permissions.
* Results include current directory
* Left padded 0s are ignored.
* You can also specify the ‘-size’ option to look for files with specific sizes
* i.e., ‘find . -size 100k’
* This looks for files or directories that are exactly 100k
* ‘M’ denotes megabytes
* A number with no specifier means bytes
* If you prepend a ‘+’ it means over that file size
* i.e., ‘find . -size +100k’
* This looks for files over 100 kilobytes
* If you prepend a ‘-’ it means files under that size
* i.e., ‘find . -size +100k’
* This looks for files under 100 kilobytes
* We can use the ‘-not’ option to search for object that don’t have a specific criteria
* i.e., ‘find . -not –type f’
* This searches for all objects in directory that are not a file.
* i.e., ‘find . -type f –not –name “\*.php”’
* This searches for all files and directories that do not have ‘.php’ extension
* The ‘-maxdepth’ option limits the recursive capability of ‘find’
* It will only go to the depth of directories specified.
* 1 means the only search the current directory, 2 means the current directory and 1 level down, etc.
* i.e. ‘find /etc/ -maxdepth 1 -type f  –name “\*.conf”
* This searches for all the configurations files only in the /etc/ directory.
* Since ‘-maxdepth’ is a global/normal option (always true), it must be specified before other options, but after arguments.
* Note: /etc directory contains configuration files for applications.
* Have ‘.conf’ extension.

**Grep**

The grep command allows you to search for strings in files, I.e.:

* ‘grep ‘text’ file1 [file2[file3[…]]]’

You can use the -I option to ignore case sensitivity, I.e.

* ‘grep -I ‘abc’ file1.txt’

You can use wildcards to specify multiple files, I.e.:

* ‘grep -I ‘abc’ \*’
* This checks all files in the current directory

You can use the recurisve ‘-R’ option, to search all files in the current and subdirectories as well, for example:

* ‘grep -I -R ‘abc’ \*’

You can use the –n option in grep and it will give you the line number where the search for string was found in the file, I.e.

* ‘grep -I –n ‘abc’ \*’

The ‘–exec’  option/command, which stands for execute, is an action that lets you pass the results of ‘find’ as arguments for ‘grep’, I.e.

* ‘find -type f -iname ‘\*.txt’ -exec grep -I –n ‘hello’ {} +’
* The {} + specifies the end of the ‘-exec’ option and passes the results of find as an argument to grep

You can redirect the output of a command by using the ‘>’ command, I.e.

* ‘find -type f -iname ‘\*.txt’ -exec --exclude=output.txt grep -I –n ‘hello’ {} + > output.txt’
* This redirects all the lines in files containing ‘hello’ to ‘output.txt’
* ‘--exclude=output.txt’ must be included so newly created file is not part of grep command.
* ‘exclude’ option excludes a specific file to be searched by ‘grep’

The ‘tee’ command is similar to ‘>’. It takes standard input and outputs it to standard output as well as to a specified text file, I.e.

* find . -type f -iname "\*.txt" -exec  grep -i -n --exclude=f.txt "hello" {} + | tee outfile.txt
* The ‘|’ command converts standard output into standard input to the following command, I.e. ‘tee’

**Processes**

Task manager in windows shows running applications.

A process is similar, it is a program/application that Is currently running.

**Top**

The ‘top’ command is used to show a list of top linux processes.

* ‘q’ is pressed to exit the screen that is shown by top

The output has the following column:

* **PID:** Shows task’s unique process id.
* **PR:** The process’s priority. The lower the number, the higher the priority.
* **VIRT:** Total virtual memory used by the task.
* **USER:** User name of owner of task.
* **%CPU:** Represents the CPU usage.
* **TIME+:** CPU Time, the same as ‘TIME’, but reflecting more granularity through hundredths of a second.
* **SHR:** Represents the Shared Memory size (kb) used by a task.
* **NI:** Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
* **%MEM:** Shows the Memory usage of task.
* **RES:** How much physical RAM the process is using, measured in kilobytes.
* **COMMAND:** The name of the command that started the process.

‘ps aux’ shows the entire list of processes running at the time the command Is executed

If we want to look for a particular process from the entire list of processes, we can pass the output of ‘ps aux’ to ‘grep’, I.e.

* ‘ps aux | grep “firefox”'

If you want to look for the process id of a particular process, then you can use the pgrep command, I.e.

* ‘pgrep firefox’
* ‘firefox’ is the name of the command.

**Kill**

The kill command stops a process from running given the PID, I.e.

* ‘kill -9 123’
* -9 option means the ‘kill’ process cannot be stopped or ignored.
  + - Program that is killed does not perform cleanup

‘**killall’**

If there are more than one process ids associated with a particular command, instead of executing kill and a list of those PIDs, you can use killall followed by the command name, I.e.’

* ‘killall firefox’

**‘cat’**

Used to concatenate the contents of files and read them sequentially in standard output.

* The contents from each file are separated by a newline character.
* You can use ‘cat’ with a single file to just read the contents of that file itself

**‘rmdir’**

Used to remove empty directories

**‘man’**

Means ‘manual’, used to get information on a specific command and how to use it.

* ‘— -help’ is built in ‘usage’ of ‘man’. Not all commands implement it.
* ‘man’ is a command itself that is a pager program that reads a manual.

**Linux Command Line: What’s happening on this machine?**

As a Linux sysadmin some things you will want to know are:

* Who’s logged in?
* What processes are running?
* What’s happening on the network?
  + What ports are listening?

**‘w’**

This command shows you the uptime, current time, how many users are logged in (each new session for a user counts as a new user), CPU load average.

* Used to see if there is any malicious activity occurring on network.

**‘top’**

Command that shows processes and how much CPU they are using as well as memory usage.

**‘netstat’**

Lets you see what ports are open and listening.

* ‘-t’ lets you see tcp ports
* ‘-u’ lets you see udp ports
* ‘-p’ lets you see programs using ports
* ‘-l’ lets you what ports are listening
* ‘-n’ means numeric
* You must use sudo to get all information about ports.
* Command must be installed using ‘sudo apt install net-tools’

**Linux: Text Editors**

‘nano’ is the default text editor for Ubuntu and all Debian distributions of Linux.

* Ctrl+ ‘o’ to save
* Ctrl + ‘x’ to quit
* Ctrl + ‘w’ can be used to search for word.

‘vim’ is a popular editor as well, means ‘vi improved’

* ‘:q’ means quit
* ‘:q!’ means quit without prompting (i.e. force quitting)
* ‘:w’ means ‘write’ or save what you write
* ‘:wq’ means save and quit
* Press ‘I’ to go to ‘insert’ mode and insert text
* Press ‘q’ to quit insert mode.

‘emacs’ is another popular text editor.

* Has a higher learning curve
* Faster and more efficient
* Good for programming/writing code
  + Other text editors are fine for sys admins or for security jobs

**Linux Shell Features: Pipes and Redirections**

**‘echo’**

Writes input to standard output

* i.e. ‘echo “hello world”’
  + output will be displayed to standard output

In Linux/Unix, everything is a file. Regular file, Directories, and even Devices are files. Every File has an associated number called File Descriptor (FD).

Your screen also has a File Descriptor. When a program is executed the output is sent to File Descriptor of the screen, and you see program output on your monitor. If the output is sent to File Descriptor of the printer, the program output would have been printed.

Standard input is input you put with keyboard.

* Input can also come from files, network, etc.
* File descriptor ‘0’ in shell
* STDIN

Standard Output is output displayed in shell.

* Other outputs could file, network, or standard error.
* File descriptor ‘1’ in shell.
* STDOUT

Standard Error output is displayed in shell,

* File descriptor ‘2’ in shell
* STDERR

We can use the ‘>’, or create and overwrite operator, to redirect output of echo or any command to a file and overwrite its existing content.

* If we prepend ‘>’ with a 0, 1, or 2 then the output will be redirected to either STDIN/OUT/ERR.
  + The default behavior for ‘>’ is ‘1>’ and will redirect standard output of ‘echo’ or other function to file, i.e.
    - ‘echo “Some stuff’ > somefile.txt’
      * This redirects standard output to somefile.txt
    - ‘echo “Some stuff” 0> somefile.txt’
      * This redirects output to standard input and somefile.txt will be created, but it will not have any content
      * ‘echo “Some stuff” 2> err.txt’ will redirect standard error to ‘err.txt’

We can use the ‘>>’, or create or append operator, to redirect output of some command to a file and append to it (no overwrite) with the same options as above.

The ‘<’ operator command can redirect standard input into a command, i.e.

* Mail -s "Subject" to-address < Filename
  + This will attach ‘Filename’ to the email created by the mail program.
  + The mail program allows one to send emails from the shell.

The ‘<<’ command can redirect standard input into a file without overwriting the file.

Pipes – denoted by ‘|’

The ‘|’ command allows you to use the output of one command as the input to another command.

* I.e. ‘ps aux | less’
  + Less gets a large amount of text paginates it so it fits on screen.

You can pipe multiple times, i.e.

* ‘ps aux | uniq | sort | grep “some search”’

You can use pipe to chain commands and filter, search, and sort data and more.

Differences between Pipe and Redirect:

Pipes is used to pass output as input to another program or utility

Redirect is used to pass output to file or stream.

**Filter Output and Finding Things (&&, cut, sort, uniq, wc, grep):**

**‘&&’**

Used to run programs consecutively contingent on preceding programs run successfully, i.e.

* ‘prog1 && prog2’
  + If prog1 runs successfully, then program2 will run
    - For example, prog1 may create a file, and prog2 may perform actions on newly created file, but if prog1 never creates a file, then prog2 will not run.
* ‘ls file.txt && echo “file successfully opened”’

**‘cut’**

Used to cut strings on a specified delimiter and then access fields of the cut string, i.e.

* ‘cat file.txt | cut -d: -f2’
  + This says cut the output of the contents of ‘file.txt’ by delimiter ‘:’ and output field 2 of the cut string(s)
  + Given input:

FName:Daniel

LName:Medina

Output is

Daniel

Medina

* Delimiter must be a single character.

**‘sort’**

Will sort output of a program alphabetically, i.e.

* ‘cat file.txt | sort -bf’
  + ‘-b’ option ignores blanks
  + ‘-f’ option folds lowercase letters to uppercase

**‘uniq’**

Used to print unique lines from output, i.e.

* Before uniq
  + Daniel

Daniel

Medina

* After uniq
  + Daniel

Medina

* Ignores trailing whitespace

**‘wc’**

Used to print the number of lines, words, and characters in a given text, i.e.

* ‘wc file.txt’
  + If file has contents:
    - Dan

Ed Med

* + - Output would be:

‘2 3 11 file.txt’

* + - Escape characters are included in character count

**‘grep’**

Finds lines in text that match word or pattern, i.e.

* ‘grep FName file.txt’ given contents in file:

FName:Daniel

FName:Daniel

LName:Medina

Will output:  
FName:Daniel

FName:Daniel

* You can also search for multiple files in a directory, i.e.
  + ‘grep FName ./\*’
    - Output will be:

./file.txt:FName:Daniel

/file.txt:FName:Daniel

* + - * Output includes the directory and file name where the lines were found.
* You can chain multiple program together like so with grep, i.e.
  + ‘grep FName ./\* | uniq | cut -d: -f1’
    - This will give you the unique file names that contain the phrase FName

**Package Management with apt-get**

Installing security and software updates on system is very important as a SysAdmin.

This can be done with apt-get program and update and upgrade commands, i.e.

* ‘sudo apt-get update’
  + Updates all software repositories on system with latest version
  + But doesn’t upgrade installed binaries
* ‘sudo apt-get -upgrade’
  + Will update all binaries with latest version

You need root permissions to do this

* ‘root’ is superuser on Linux
  + Can do anything on system
  + Must use ‘sudo’