Linux Notes

* Managing graphical interface sessions
  + - X Window System – X display manager starts a the end of the boot process
      * Responsible for starting graphics system, logging the user in, starting user’s desktop environment
    - GNOME – a desktop environment with a graphical user interface
    - System Startup and Logging In and Out
    - Graphical Desktop Background
    - gnome-tweaks – use this to access more setting options
      * allows installation of extensions by external parties
    - Changing theme – do this by using gnome-tweaks or gnome-extension-app
  + Logging in and out in Ubuntu, openSUSE and CentOS
  + Switching Users in Ubuntu
  + Shutting Down and Restarting on GNOME
  + Suspending the system
* Basic Operations
  + Finding applications
    - From the Applications menu in the upper-left corner.
    - From the Activities menu in the upper-left corner.
    - In some Ubuntu versions, from the Dash button in the upper-left corner.
    - For KDE, and some other environments, applications can be opened from the button in the lower-left corner.
  + Default Applications
    - Settings menu (on all recent Linux distributions) and then click on either Default Applications or Details > Default Applications.
  + Setting Default Applications
  + File manager – every distribution uses the Nautilus (File Manager) utility.
  + Home directories
* Applying system, display, and date and time settings using the System Settings panel.
  + For the GNOME Desktop Manager, one clicks on the upper right-hand corner and then selects the tools image (screwdriver crossed with a wrench or a gear).
  + System Settings Menus
  + gnome-tweaks. Can do the following with gnome-tweaks
    - selecting a theme
    - configuring extensions which you can get from your distribution or download from the Internet
    - control fonts
    - modify the keyboard layout
    - and set which programs start when you login.
  + Display Settings
  + Setting Resolution and Configuring Multiple Screens
  + Date and Time Settings
* Tracking the network settings and manage connections using Network Manager in Linux.
  + Network Time Protocol - The Network Time Protocol (NTP) is the most popular and reliable protocol for setting the local time by consulting established Internet servers.
  + Network Configuration
  + Wired and Wireless Connections
  + Configuring Wireless Connections
  + Mobile Broadband and VPN Connections
* Installing and update software in Linux from a graphical interface.
  + Installing and Updating Software
  + Debian Packaging
* **Introduction to the Command Line**
* **Launching Terminal Windows**
* **Some Basic Utilities**
  + **cat: used to type out a file (or combine files).**
  + **head: used to show the first few lines of a file.**
  + **tail: used to show the last few lines of a file.**
  + **man: used to view documentation.**
* **The Command Line**
  + **Command**: The command is the name of the program you are executing. It may be followed by one or more options (or switches) that modify what the command may do.
  + **Options**: Options usually start with one or two dashes, for example, -p or --print, in order to differentiate them from arguments
  + **Arguments**: represent what the command operates on.
  + **Sudo**
    - **Steps for Setting Up and Running sudo**
  + **Switching Between the GUI and the Command Line**
  + **Virtual Terminals**
  + **Turning Off the Graphical Desktop from the Command Line Terminal** 
    - **$ sudo systemctl stop gdm (or sudo telinit 3)**
    - and restart it (after logging into the console) with:
    - **$ sudo systemctl start gdm (or sudo telinit 5)**
    - On Ubuntu versions before 18.04 LTS, substitute **lightdm** for **gdm**.
* **Basic Operations**
  + **Logging In and Out**
  + **Rebooting and Shutting Down**
    - **The halt and poweroff commands issue shutdown -h to halt the system; reboot issues shutdown -r and causes the machine to reboot instead of just shutting down.**
  + **Locating Applications**
    - One way to locate programs is to employ the which utility. For example, to find out exactly where the diff program resides on the filesystem:

**$ which diff**

**/usr/bin/diff**

**If which does not find the program, *whereis* is a good alternative because it looks for packages in a broader range of system directories:**

**$ whereis diff**

**diff: /usr/bin/diff /usr/share/man/man1/diff.1.gz /usr/share/man/man1p/diff.1p.gz**

* + - **Accessing Directories**
      * **Pwd** Displays the present working directory
      * **cd ~** or **cd** Change to your home directory (shortcut name is ~ (tilde))
      * **cd ..** Change to parent directory (..)
      * **cd -** Change to previous directory (- (minus))
  + **Understanding Absolute and Relative Paths**
  + Exploring the Filesystem
    - **cd /**  - Changes your current directory to the root (/) directory (or path you supply)
    - **ls**  - List the contents of the present working directory
    - **ls –a**  - List all files, including hidden files and directories (those whose name start with . )
    - **tree**  - Displays a tree view of the filesystem
    - Hard Links: The ln utility is used to create hard links and (with the -s option) soft links, also known as symbolic links or symlinks.
    - Soft (Symbolic) Links: Soft (or Symbolic) links are created with the -s option, as in:

**$ ln -s file1 file3**

* + **Navigating the Directory History**
    - **The cd command remembers where you were last, and lets you get back there with cd -.**
    - For remembering more than just the last directory visited, use **pushd** to change the directory instead of **cd**; this pushes your starting directory onto a list.
    - Using **popd** will then send you back to those directories, walking in reverse order (the most recent directory will be the first one retrieved with **popd**).
    - **The list of directories is displayed with the dirs command.**
* **Working with Files**
  + **Viewing Files**
    - **cat Used for viewing files that are not very long; it does not provide any scroll-back.**
    - **tac Used to look at a file backwards, starting with the last line.**
    - **less Used to view larger files because it is a paging program. It pauses at each screen full of text, provides scroll-back capabilities, and lets you search and navigate within the file.**
    - **tail Used to print the last 10 lines of a file by default. You can change the number of lines by doing -n 15 or just -15 if you wanted to look at the last 15 lines instead of the default.**
    - **head The opposite of tail; by default, it prints the first 10 lines of a file.**
  + **Touch - often used to set or update the access, change, and modify times of files. By default, it resets a file's timestamp to match the current time.**

**$ touch <filename>**

* **mkdir** 
  + **mkdir sampdir - It creates a sample directory named sampdir under the current directory.**
* **rmdir: Removing a directory is done with rmdir. The directory must be empty or the command will fail. To remove a directory and all of its contents you have to do rm -rf**

**Moving, Renaming or Removing a File**

* **mv Rename a file**
* **rm Remove a file**
* **rm –f Forcefully remove a file**
* **rm –i Interactively remove a file**
* **Standard File Streams -** Through redirection you can direct the input and output of a command to and from other files and programs
  + standard input (standard in or stdin) - information inputted into the terminal through the keyboard or input device.
  + standard output (standard out or stdout) - information outputted after a process is run
  + and standard error (or stderr) - error message outputted by a failed process

Example:

$ echo "Hello" > hello.txt

The > command redirects the standard output to a file. Here, "Hello" is entered as the standard input, and is then redirected to the file hello.txt by > .

As a reminder, the cat command outputs the contents of a file to the terminal. When you type:

$ cat hello.txt

the contents of hello.txt are displayed.

* **I/O Redirection -** redirect the three standard file streams so that we can get input from either a file or another command, instead of from our keyboard, and we can write output and errors to files or use them to provide input for subsequent commands.

> takes the standard output of the command on the left, and redirects it to the file on the right.

**$ cat deserts.txt > forests.txt**

**Note that > overwrites all original content in forests.txt. When you view the output data by using cat on forests.txt, you will see only the contents of deserts.txt.**

>>

Now we know how to overwrite a file’s contents, but what if we want to be able to add to a file without losing the original text? We can use the >> command!

**$ cat deserts.txt >> forests.txt**

>> takes the standard output of the command on the left and appends (adds) it to the file on the right.

Here, the output data of forests.txt will contain the original contents of forests.txt with the content of deserts.txt appended to it.

<

< takes the standard input from the file on the right and inputs it into the program on the left. Here, deserts.txt is the standard input for the cat command. The standard output appears in the terminal.

**$ cat < deserts.txt**

* **Pipes -** output of one command or program into another as its input/ combinesthe actions of several commands into one

**Example:**

**$ command1 | command2 | command3**

Efficient because command2 and don’t have to wait for the previous pipeline commands to complete before they can begin hacking at the data in their input streams. Also, saves disk space and reduces reading and writing from disk

* Searching for Files
  + You can search for files in both your home directory space, or in any other directory or location on the system.
  + The main tools for doing this are the ***locate*** and ***find*** utilities
  + ***Locate:*** performs a search taking advantage of a previously constructed database of files and directories on your system, matching all entries that contain a specified character string
  + To get a shorter and more relevant list, we can use the **grep** program as a filter. grep will print only the lines that contain one or more specified strings, as in:

**$ locate zip | grep bin**

this will list all the files and directories with both zip and bin in their name

* + ***locate*** utilizes a database created by a related utility, ***updatedb***. Most Linux systems run this automatically once a day. However, you can update it at any time by just running ***updatedb*** from the command line as the root user.
* Wildcards and Matching File Names
  + **?** Matches any single character
  + **\*** Matches any string of characters
  + **[set]** Matches any character in the set of characters, for example [adf] will match any occurrence of a, d, or f
  + **[!set]** Matches any character not in the set of characters
  + ***Find:*** recurses down the filesystem tree from any particular directory (or set of directories) and locates files that match specified conditions. The default pathname is always the present working directory
  + Commonly used options to shorten the list include:
    - ***name*** (only list files with a certain pattern in their name)
    - ***-iname*** (also ignore the case of file names)
    - and ***-type*** (which will restrict the results to files of a certain specified type, such as d for directory, l for symbolic link, or f for a regular file, etc.).

Examples:

Searching for files and directories named **gcc**:

**$ find /usr -name gcc**

Searching only for directories named gcc:

**$ find /usr -type d -name gcc**

Searching only for regular files named gcc:

**$ find /usr -type f -name gcc**

* Can find files according to attributes, such as when they were created, last used, etc., or based on their size.
  + find files based on time:

**$ find / -ctime 3 -** -ctime is when the inode metadata (i.e. file ownership, permissions, etc.) last changed

* + To search for accessed/last read would use **-atime**
  + To search for modified/last written use **-mtime**
  + To find files based on sizes: **$ find / -size 0**
* Installing software
  + Fedora and RHEL 8 replaced the older yum utility with dnf, thereby eliminating a lot of historical baggage, as well as introducing many nice new capabilities. dnf is pretty much backwards-compatible with yum for day-to-day commands.
  + dnf is the open source command-line package-management utility for the RPM-compatible Linux systems that belongs to the Red Hat family. dnf has both command line and graphical user interfaces