**Linux Startup Course**

Summary:

1. **What is Linux ?**
2. **Login/Logout**
3. **The shell**
4. **Users**
5. **SUDO & SU**
6. **Moving around (through directories)**
7. **Basic Commands**
8. **Processes**
9. **Mounts**
10. **Editing tools: vi**
11. **What is Linux ?**

In 1984, an initiative was undertaken by Richard Stallman whose primary goal was to create a completely free, UNIX-compatible, open source operating system with global collaboration from software developers. The initiative was called the GNU (*GNU's Not Unix)* Project and by 1991 significant software had been developed. The only criticai piece missing was a kernel to drive the operating system. This gap was filled by the kernel created by Linus Torvalds in 1991 during his computer science studies at the University of Helsinki in Finland. The name "Linux" was given to the new operating system kernel that Linus developed which was very UNIX-like. Linux operating system was released under the GNU *General Public License* (GPL) and initially written to run on Intel x86 architecture computers. The first version (0.01) of the operating system was released in September 1991 with little more than 10,000 lines of code. In March 1994, the first major release (1.0.0) debuted followed by version 2.0.0 in June 1996, 2.2.0 in January 1999, 2.4.0 in January 2001 and 2.6.0 in December 2003. Presently, version 2.6.27.7 with several million lines of code is in circulation. The Linux kernel, and the operating system in general, has been enhanced with contributions from thousands of software programmers around the world under the GNU GPL. which provides general public access to the Linux source code free of charge with full consent for amendments and redistribution.

Today, Linux runs on a variety of computer hardware platforms, from laptop and desktop computers to massive mainframe systems. Linux also runs as the base operating system on a range of other electronic devices such as routers, switches, RAID arrays, tape libraries, video games and mobile phones. Numerous vendors including Red Hat. HP. IBM, Sun, Novell and Dell offer support to Linux users worldwide.

The functionality, adaptability. portability and cost-effectiveness that Linux offers has made this operating system the main alternative to proprietary UNIX and Windows operating systems. At present, more than hundred different flavors of Linux are available from various vendors, organizations and individuals; only a few of them are popular and have wide acceptance.

Linux is generally used by government agencies, corporate businesses, academic institutions, scientific organizations and home users. Linux usage in home computers is rapidly rising.

Just like Windows XP, Windows 7, Windows 8, and Mac OS X, Linux is an operating system. An operating system is software that manages all of the hardware resources associated with your desktop or laptop. To put it simply – the operating system manages the communication between your software and your hardware. Without the operating system (often referred to as the “OS”), the software wouldn’t function.

The OS is comprised of a number of pieces:

* **The Bootloader:** The software that manages the boot process of your computer. For most users, this will simply be a splash screen that pops up and eventually goes away to boot into the operating system.
* **The kernel:** This is the one piece of the whole that is actually called “Linux”. The kernel is the core of the system and manages the CPU, memory, and peripheral devices. The kernel is the “lowest” level of the OS.
* **Daemons:** These are background services (printing, sound, scheduling, etc) that either start up during boot, or after you log into the desktop.
* **The Shell:** You’ve probably heard mention of the Linux command line. This is the shell – a command process that allows you to control the computer via commands typed into a text interface. This is what, at one time, scared people away from Linux the most (assuming they had to learn a seemingly archaic command line structure to make Linux work). This is no longer the case. With modern desktop Linux, there is no need to ever touch the command line.
* **Graphical Server:** This is the sub-system that displays the graphics on your monitor. It is commonly referred to as the X server or just “X”.
* **Desktop Environment:** This is the piece of the puzzle that the users actually interact with. There are many desktop environments to choose from (Unity, GNOME, Cinnamon, Enlightenment, KDE, XFCE, etc). Each desktop environment includes built-in applications (such as file managers, configuration tools, web browsers, games, etc).
* **Applications:** Desktop environments do not offer the full array of apps. Just like Windows and Mac, Linux offers thousands upon thousands of high-quality software titles that can be easily found and installed. Most modern Linux distributions (more on this in a moment) include App Store-like tools that centralize and simplify application installation. For example: Ubuntu Linux has the Ubuntu Software Center (Figure 1) which allows you to quickly search among the thousands of apps and install them from one centralized location.

The most popular Linux distributions are:

* [Ubuntu Linux](http://www.ubuntu.com/)
* [Linux Mint](http://www.linuxmint.com/)
* [Arch Linux](https://www.archlinux.org/)
* [Deepin](http://www.linuxdeepin.com/index.en.html)
* [Fedora](http://fedoraproject.org/)
* [Debian](https://www.debian.org/)
* [openSUSE](http://www.opensuse.org/en/).

Each distribution has a different take on the desktop. Some opt for very modern user interfaces (such as Ubuntu’s Unity, above, and Deepin’s Deepin Desktop), whereas others stick with a more traditional desktop environment (openSUSE uses KDE). For an easy guide to Linux desktops check out [How to Find the Best Linux Desktop for You](https://www.linux.com/learn/tutorials/783109-how-to-choose-the-best-linux-desktop-for-you).

You can check out the top 100 distributions on the [Distrowatch](http://distrowatch.com/) site.

And don’t think the server has been left behind. For this arena, you can turn to:

* [Red Hat Enterprise Linux](http://www.redhat.com/)
* [Ubuntu Server](http://www.ubuntu.com/download/server)
* [CentOS](https://www.centos.org/)
* [SUSE Enterprise Linux](https://www.suse.com/products/server/).

Some of the above server distributions are free (such as Ubuntu Server and CentOS) and some have an associated price (such as Red Hat Enterprise Linux and SUSE Enterprise Linux). Those with an associated price also include support.

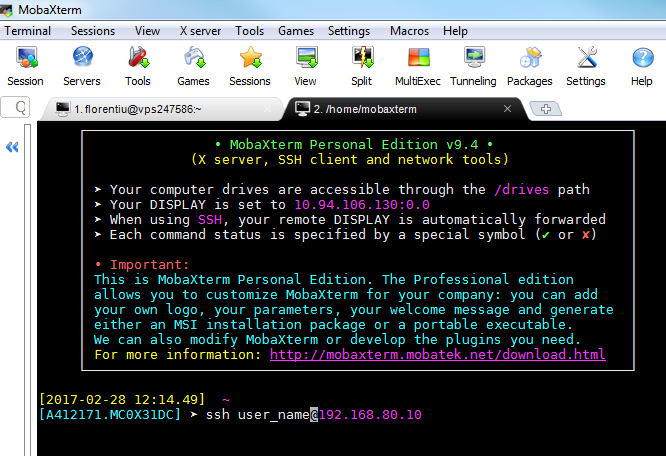
1. **Login/Logout**

Main tools that can be used to login remotely on a linux server are putty and mobaxterm. These are the links from where you can download them:

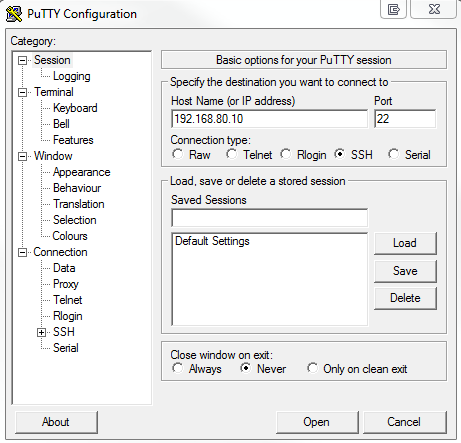
Mobaxterm: <http://mobaxterm.mobatek.net/download.html>

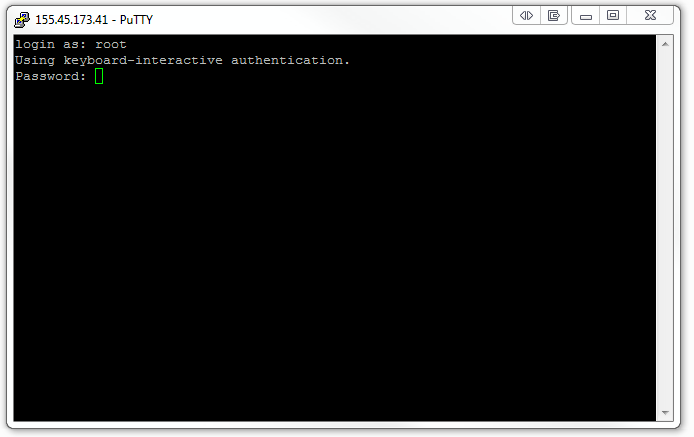
Putty: <http://www.putty.org>

Way to connect remotely with mobaxterm:



Way to connect remotely to linux server with putty:





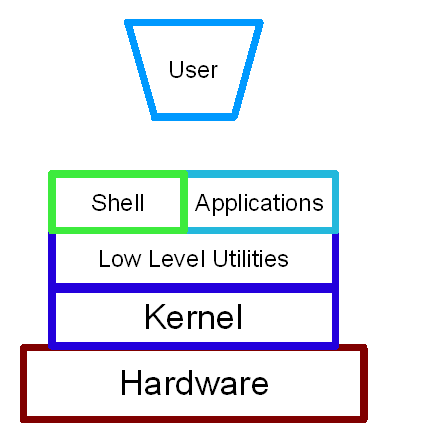
Logout is done with the command: **logout** or **exit**

1. **The shell**

The shell is the command prompt within Linux where you can type commands. If you have logged into a machine over a network (using ssh or telnet) then the commands you entered were run by the shell. If you are logged in using a graphical interface then you will may need to open a terminal client to see the shell. There are several different terminal clients available such as xterm, konsole and lxterm, or it may be just named Terminal Emulator. It's location in the menu differs between different distributions if the start menu allows searching (most do, but not all) then type term and it should show an appropriate terminal, if not then look under the system or utilities menu.

Windows users may be familiar with the concept of a command prompt, or DOS prompt, which looks similar to a UNIX shell. The UNIX shell has more features and is practically an entire programming language, although don't let that put you off as you can use the shell without any programming ability. Even if you don't "do programming" you may find that's it's worth learning a little bit of shell script programming as it can be used to make your life easier.

Often people seeing the shell will think that this is the UNIX / Linux operating system. It is in fact a program that is running on top of the operating system. To take a basic view of how Linux is built up see the diagram below:



The different layers of the Linux operating system

The kernel is the heart of the operating system. This is the bit that is actually Linux. The kernel is a process that runs continuously managing the computer. The kernel is a very specific task so to allow programs to communicate with it there are a number of low level utilities that provide an interface between the application and the kernel.

The shell is an application that allows users to communicate with the computer. It is a text based application that allows programs to be started and tasks to be run. The shell is within a collections of utilities known as GNU. Without the kernel the computer cannot run and without the GNU utilities it can't do anything useful which is why the operating system is sometimes called GNU/Linux; although this ignores the host of other applications that are also included (for brevity I am just using Linux to mean everything included on the Linux distribution).

Here's a list of the most common UNIX shells:

**Name of shell Command name Description**

Bourne shell sh The most basic shell available on all UNIX systems

Korn Shell ksh / pdksh Based on the Bourne shell with enhancements

C Shell csh Similar to the C programming language in syntax

**Bash Shell** **bash** **Bourne Again Shell combines the advantages of the Korn Shell and the C Shell. The default on most Linux distributions.**

When you login to a Linux machine (or open a shell window) you will normally be in the bash shell.

**Bourne Shell** - This is the oldest shell and as such is not as feature rich as many of the other shells. It's feature set is sufficient for most programming needs however it does not have some of the user conveniences that are liked on the command line. There is no option to re-edit previous commands or to control background jobs. As the bourne shell is available on all UNIX systems it is often used for programming script files as it offers maximum portability between different UNIX versions. Bash is fully backwards compatible with the Bourne Shell so running the bourne shell on Linux will often call the bash shell (using a link between the files).

**Korn Shell** - This is based on the Bourne shell. One enhancement that is particularly useful is its command-line editing facility. It is possible using either vi or emacs keys to recall and edit previous commands. This is not as easy to use as some of the other shells, but work well across a network or using a physical terminal (rare these days). It also has more powerful programming constructs than the bourne shell, however these are not as portable. To run the Korn shell you can run either ksh or pdksh from the normal shell (assuming it is installed).

**C Shell** - The c shell syntax is taken from the C programming language. As such it is a useful tool for anyone familiar with programming C.

**Bash Shell** - The Bash shell is a combination of features from the Bourne Shell and the C Shell. It's name comes from the Bourne Again SHell. It has a command-line editor that allows the use of the cursor keys in a more "user friendly" manner than the Korn shell. It also has a useful help facility allowing you to get a list of commands by typing the first few letters followed by the "TAB" key. It is the default shell on most Linux distributions and unless otherwise specified is the shell used for the future examples.

1. **Users**

Linux/Unix operating systems have the ability to multitask in a manner similar to other operating systems. However, Linux’s major difference from other operating systems is its ability to have multiple users. Linux was designed to allow more than one user to have access to the system at the same time. In order for this multiuser design to work properly, there needs to be a method to protect users from each other. This is where permissions come in to play.

**Read, Write & Execute Permissions**

Permissions are the “rights” to act on a file or directory. The basic rights are read, write, and execute.

**Read** - a readable permission allows the contents of the file to be viewed. A read permission on a directory allows you to list the contents of a directory.

**Write** - a write permission on a file allows you to modify the contents of that file. For a directory, the write permission allows you to edit the contents of a directory (e.g. add/delete files).

**Execute** - for a file, the executable permission allows you to run the file and execute a program or script. For a directory, the execute permission allows you to change to a different directory and make it your current working directory. Users usually have a default group, but they may belong to several additional groups.

**Viewing File Permissions**

To view the permissions on a file or directory, issue the command *ls -l <directory/file>.* Remember to replace the information in the **< >** with the actual file or directory name. Below is sample output for the ls command:



The first ten characters show the access permissions. The first dash (-) indicates the type of file (d for directory, s for special file, and - for a regular file). The next three characters (rw-) define the owner’s permission to the file. In this example, the file owner has read and write permissions only. The next three characters (r–) are the permissions for the members of the same group as the file owner (which in this example is read only). The last three characters (r–) show the permissions for all other users and in this example it is read only.

**Working with Users, Groups, and Directories**

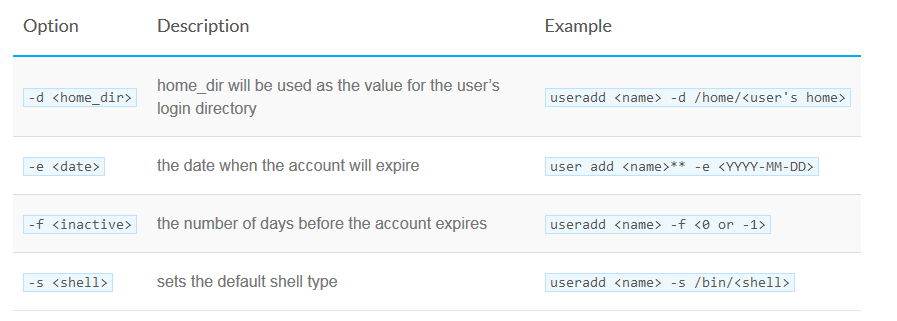
The following sections will go over the commands needed to create, delete, and modify user accounts. Groups will be covered, as well as commands for creating and deleting directories. You will be provided with the commands and descriptions needed for working with users, groups, and directories.

**Creating and Deleting User Accounts**

To create a new standard user, use the useradd command. The syntax is as follows:



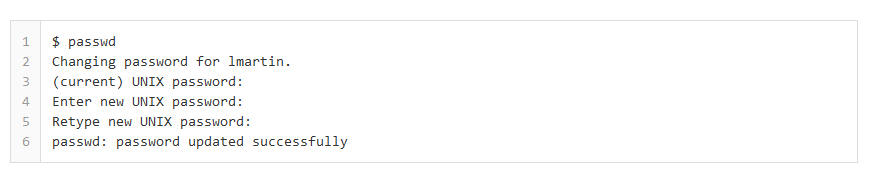
The useradd command utilizes a variety of variables, some of which are shown in the table below:



You will need to set a password for the new user by using the passwd command. Note, you will need root privileges to change a user password. The syntax is as follows:



The user will be able to change their password at any time using the passwd command with the syntax. Below is an example:



It is important to note that security should always be taken very seriously. Therefore, it is strongly recommended to use unique passwords for each account. Never share or give your password to other users.

To remove a user account, enter the following command:



Issuing the command above will only delete the user’s account. Their files and home directory will not be deleted.

To remove the user, their home folder, and their files, use this command:

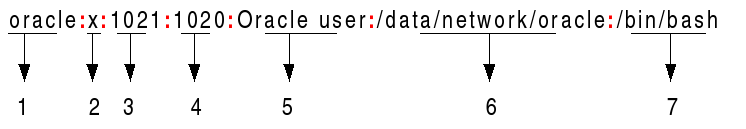


The main files for users are: **/etc/passwd** & **/etc/shadow**

Users are also included in groups. The main file for groups which is connected to the users files is: **/etc/group**

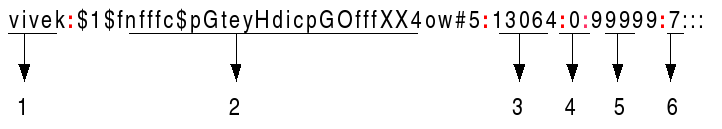
Content of the 3 files mentioned above

**/etc/passwd**

****

1. **Username**: It is used when user logs in. It should be between 1 and 32 characters in length.
2. **Password**: An x character indicates that encrypted password is stored in /etc/shadow file.
3. **User ID (UID)**: Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups.
4. **Group ID (GID)**: The primary group ID (stored in /etc/group file)
5. **User ID Info**: The comment field. It allow you to add extra information about the users such as user's full name, phone number etc. This field use by finger command.
6. **Home directory**: The absolute path to the directory the user will be in when they log in. If this directory does not exists then users directory becomes /
7. **Command/shell**: The absolute path of a command or shell (/bin/bash). Typically, this is a shell. Please note that it does not have to be a shell.

**/etc/shadow**

****

1. **User name**: It is your login name
2. **Password**: It your encrypted password. The password should be minimum 6-8 characters long including special characters/digits
3. **Last password change (lastchanged):** Days since Jan 1, 1970 that password was last changed
4. **Minimum**: The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
5. **Maximum**: The maximum number of days the password is valid (after that user is forced to change his/her password)
6. **Warn** : The number of days before password is to expire that user is warned that his/her password must be changed
7. **Inactive** : The number of days after password expires that account is disabled
8. **Expire** : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used

**/etc/group**

cdrom:x:24:vivek,student13,raj

| | | |

1 2 3 4

1. **Group name**: It is the name of group. If you run ls -l command, you will see this name printed in the group field.
2. **Password**: Generally password is not used, hence it is empty/blank. It can store encrypted password. This is useful to implement privileged groups.
3. **Group ID (GID)**: Each user must be assigned a group ID. You can see this number in your /etc/passwd file.
4. **Group List**: It is a list of user names of users who are members of the group. The user names, must be separated by commas.
5. **SUDO & SU**

**Understanding SUDO**

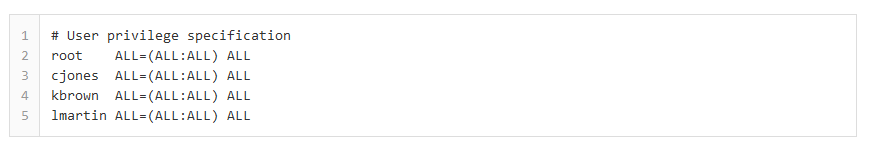
Root is the super user and has the ability to do anything on a system. Therefore, in order to have protection against potential damage sudo is used in place of root. Sudo allows users and groups access to commands they normally would not be able to use. Sudo will allow a user to have administration privileges without logging in as root. A sample of the sudo command is as follows:

sudo *command options [other]*

In order to provide a user with sudo ability, their name will need to be added to the sudoers file. This file is very important and should not be edited directly with a text editor. If the sudoers file is edited incorrectly it could result in preventing access to the system.

Therefore the visudo command should be used to edit the sudoers file. At a command line, log into your system as root and enter the command visudo.

Below is the portion of the sudoers file that shows the users with sudo access.



After you have given your user account sudo privileges, save the sudoers file and log out as root. Now log in as your user and test the privileges as your user with sudo access. When a new user needs sudo access, you will now be able to edit the sudoers file with your own login using the following command:



**Understading SU**

The su (short for substitute user) command makes it possible to change a login session's owner (i.e., the user who originally created that session by logging on to the system) without the owner having to first log out of that session.

Although su can be used to change the ownership of a session to any user, it is most commonly employed to change the ownership from an ordinary user to the root (i.e., administrative) user, thereby providing access to all parts of and all commands on the computer or system. For this reason, it is often referred to (although somewhat inaccurately) as the superuser command. It is also sometimes called the switch user command.

The advantages of Using su: su is usually the simplest and most convenient way to change the ownership of a login session to root or to any other user.

More importantly, it provides a safer way for administrators on multi-user systems (as well as for users on home computers or other single-user systems) to use the system than to routinely log on as the root user. That is, there is much less potential for accidental or malicious damage if an administrator first logs on as an ordinary user (who, by default, has very limited system privileges) and uses that account for routine tasks that do not require root's sweeping powers. su can then be used to switch to the root account for only those operations that actually require root access (e.g., making system repairs and managing user accounts).

**Syntax**

A simplified expression of the syntax of the su command is:

su [options] [commands] [-] [username]

Please this link for more info:

<https://www.linux.com/blog/how-use-sudo-and-su-commands-linux-introduction>

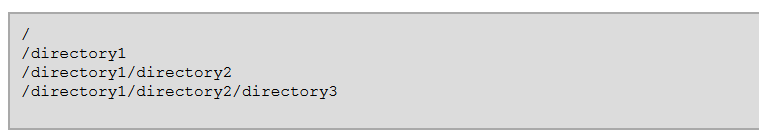
1. **Moving around (through directories)**

The main command to “move around” the server is the change directory command: cd

Changing directories is easy as long as you know where you are (your current directory) and how that relates to where you want to go.

To change directories, use the cd command. Typing this command by itself returns you to your home directory; moving to any other directory requires a pathname.

You can use absolute or relative pathnames. Absolute paths start at the top of the file system with / (referred to as root) and then look down for the requested directory; relative paths look down from your current directory, wherever that may be. The following directory tree illustrates how cd operates.



If you are currently in directory3 and you want to switch to directory1, you need to move up in the directory tree.

Executing the command



while you are in directory3, presents you with an error message explaining that there is no such directory. This is because there is no directory1 below directory3.

To move up to directory1, type:



This is an example of an absolute path. It tells Linux to start at the top of the directory tree (/) and change to directory1. **A path is absolute if the first character is a /.** Otherwise, it is **a relative path.**

Using absolute paths allows you to change to a directory from the / directory, which requires you to know and type the complete path. Using relative paths allows you to change to a directory relative to the directory you are currently in, which can be convenient if you are changing to a subdirectory within your current directory.

The command **cd ..** tells your system to go up to the directory immediately above the one in which you are currently working. To go up two directories, use the cd ../.. command.

Use the following exercise to test what you have learned regarding absolute and relative paths. From your home directory, type **the relative path**:



After using the full command in the example, you should be in the directory X11, which is where configuration files and directories related to the X Window System are available.

Take a look at your last cd command. You told your system to:

* 1. Go up one level to your login directory's parent directory (probably /home)
  2. Then go up to that directory's parent (which is the root, or /, directory)
  3. Then go down to the /etc/ directory
  4. Finally, go to the X11/ directory

Conversely, using an absolute path moves you to the /etc/X11/ directory more quickly. For example:



Absolute paths start from the root directory (/) and move down to the directory you specify.

| **Command** | **Function** |
| --- | --- |
| cd | Returns you to your login directory |
| cd ~ | Also returns you to your login directory |
| cd / | Takes you to the entire system's root directory |
| cd /root | Takes you to the home directory of the root, or superuser, account created at installation; you must be the root user to access this directory |
| cd /home | Takes you to the home directory, where user login directories are usually stored |
| cd .. | Moves you up one directory |
| cd ~*otheruser* | Takes you to *otheruser's* login directory, if *otheruser* has granted you permission |
| cd /dir1/subdirfoo | Regardless of which directory you are in, this absolute path takes you directly to subdirfoo, a subdirectory of dir1 |
| cd ../../dir3/dir2 | This relative path takes you up two directories, then to dir3, then to the dir2 directory |

1. **Basic Commands**

|  |  |
| --- | --- |
| **Command** | **Description** |
| cat [filename] | Display file’s contents to the standard output device (usually your monitor). |
| cd /directorypath | Change to directory. |
| clear | Clear a command line screen/window for a fresh start. |
| cp [options] source destination | Copy files and directories. |
| date [options] | Display or set the system date and time. |
| df [options] | Display used and available disk space. |
| du [options] | Show how much space each file takes up. |
| file [options] filename | Determine what type of data is within a file. |
| find [pathname] [expression] | Search for files matching a provided pattern. |
| grep [options] pattern [filesname] | Search files or output for a particular pattern. |
| kill [options] pid | Stop a process. If the process refuses to stop, use kill -9 pid. |
| less [options] [filename] | View the contents of a file one page at a time. |
| ln [options] source [destination] | Create a shortcut. |
| locate filename | Search a copy of your filesystem for the specified filename. |
| ls [options] | List directory contents. |
| man [command] | Display the help information for the specified command. |
| mkdir [options] directory | Create a new directory. |
| mv [options] source destination | Rename or move file(s) or directories. |
| passwd [name [password]] | Change the password or allow (for the system administrator) to change any password. |
| ps [options] | Display a snapshot of the currently running processes. |
| pwd | Display the pathname for the current directory. |
| rm [options] directory | Remove (delete) file(s) and/or directories. |
| rmdir [options] directory | Delete empty directories. |
| ssh [options] user@machine | Remotely log in to another Linux machine, over the network. Leave an ssh session by typing **exit**. |
| tail [options] [filename] | Display the last *n* lines of a file (the default is 10). |
| top | Displays the resources being used on your system. Press q to exit. |
| touch filename | Create an empty file with the specified name. |
| who [options] | Display who is logged on. |

1. **Processes**

A process is created in memory when a program or command is executed. A unique identification number, known as process identification (PID), is allocated to it, which is used by the kernel to manage the process until the program or command it is associated with, terminates. When a user logs in to the system, shell is started, which is a process. A process is any program, application or command that runs on the system.

Several processes are started at system boot up, many of which sit in memory and wait for an event to trigger a request to use their service. These background system processes are called daemons and are critical to system functionality.

**Viewing System processes**

There are two commands commonly used to view currently running processes. These are ps (process status) ant top.

The ps command without any options or arguments, lists processes specific to the terminal where the ps command is run:

# ps

PID TTY TIME CMD

4306 pts/1 00:00:00 bash

4319 pts/1 00:00:00 ps

The output has for columns: PID of the process in the first column, terminal the process belongs to in the second column, cumulative time the process given by the system CPU in the third column and actual command being executed in the last column.

Two options –e (every) and –f (full) are popularly used to generate detailed information on every process running in the system. Check the ps command man pages for more options.

# ps -ef

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 11:29 ? 00:00:01 init [5]

root 2 1 0 11:29 ? 00:00:00 [migration/0]

root 3 1 0 11:29 ? 00:00:00 [ksoftirqd/0]

root 4 1 0 11:29 ? 00:00:00 [events/0]

root 5 1 0 11:29 ? 00:00:00 [khelper]

root 21 1 0 11:29 ? 00:00:00 [kthread]

root 25 21 0 11:29 ? 00:00:00 [kblockd/0]

root 26 21 0 11:29 ? 00:00:00 [kacpid]

root 222 21 0 11:29 ? 00:00:00 [cqueue/0]

Now let’s see what happens when you run a command like find, to search for files ending with .bak in / :

# find / -type f -name \*.bak

/etc/mail/submit.cf.bak

…………………………………………………

# ps -ef

UID PID PPID C STIME TTY TIME CMD

root 4306 4303 0 11:31 pts/1 00:00:00 bash

root 4322 1 0 11:31 ? 00:00:00 gnome-screensaver

root 4889 4303 0 11:49 pts/2 00:00:00 bash

root 4917 4306 8 11:49 pts/1 00:00:00 find / -type f -name \*.bak

root 4918 4889 0 11:49 pts/2 00:00:00 ps –ef

The second method to view process information is the top command, which displays additional information including CPU and memory utilization. A sample output from a running top session is shown below:

top - 11:53:10 up 23 min, 3 users, load average: 0.11, 0.10, 0.10

Tasks: 117 total, 2 running, 115 sleeping, 0 stopped, 0 zombie

Cpu(s): 8.0%us, 3.3%sy, 0.0%ni, 87.7%id, 0.0%wa, 1.0%hi, 0.0%si, 0.0%st

Mem: 1921784k total, 704436k used, 1217348k free, 76548k buffers

Swap: 2048276k total, 0k used, 2048276k free, 426104k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

4064 root 15 0 218m 11m 6236 S 8.3 0.6 0:15.84 Xorg

4303 root 15 0 262m 15m 8480 R 2.3 0.8 0:01.84 gnome-terminal

5029 root 15 0 12744 1080 808 R 0.3 0.1 0:00.02 top

1 root 15 0 10352 700 588 S 0.0 0.0 0:01.69 init

2 root RT -5 0 0 0 S 0.0 0.0 0:00.00 migration/0

3 root 34 19 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0

4 root 10 -5 0 0 0 S 0.0 0.0 0:00.42 events/0

5 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 khelper

21 root 11 -5 0 0 0 S 0.0 0.0 0:00.00 kthread

25 root 10 -5 0 0 0 S 0.0 0.0 0:00.07 kblockd/0

26 root 20 -5 0 0 0 S 0.0 0.0 0:00.00 kacpid

222 root 11 -5 0 0 0 S 0.0 0.0 0:00.00 cqueue/0

225 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 khubd

227 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 kseriod

292 root 15 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd

293 root 16 0 0 0 0 S 0.0 0.0 0:00.00 pdflush

294 root 15 0 0 0 0 S 0.0 0.0 0:00.09 pdflush

Find out more information on top with the man top command.

**Process States**

There are several states that a process can go through in its life. There are five process states: running, waiting, sleeping, stopped and zombie.

- The running state shows that the process is currently being executed by the system CPU.

- The sleeping state means shows that the process is currently waiting for input from user or another process.

- The waiting state means that the process has received input it has been waiting for and is now ready to run as soon as its turn comes.

- The stopped state indicates that the process is currently halted and will not run even when its turn comes, unless it is sent a signal.

- The zombie state determines that the process is dead. A zombie process exists in process table just as any other process entry, but takes up no resources. The entry for zombie is retained until the parent process permits it to die.

1. **Mounts**

Unix systems have a single directory tree. All accessible storage must have an associated location in this single directory tree. This is unlike Windows where (in the most common syntax for file paths) there is one directory tree per storage component (drive).

**Mounting** is the act of associating a storage device to a particular location in the directory tree. For example, when the system boots, a particular storage device (commonly called the root partition) is associated with the root of the directory tree, i.e., that storage device is mounted on / (the root directory).

Example: You must mount the CD-ROM on a location in the directory tree (this may be done automatically when you insert the CD). Let's say the CD-ROM device is /dev/cdrom and the chosen mount point is /media/cdrom. The corresponding command is:

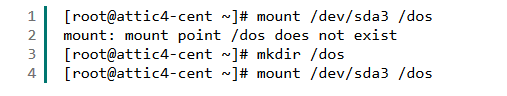
Mount /dev/cdrom /media/cdrom

Besides filesystems on partitions, floppy disks, and CDs, there are other types of filesystems. The tmpfs filesystem is a virtual memory filesystem. You can also mount filesystems from one system on another system using a networked filesystem such as NFS or AFS. You can even create a file in an existing filesystem and format that as a, possibly different, kind of filesystem and mount that too. This is often done with images of optical media, where you perhaps download an ISO CD or DVD image, then mount that file rather than burning it to real media. Swap space in a file rather than a dedicated swap partition is another example.

While the mount process actually mounts the filesystem on some device (or other resource), it is common to simply say that you "mount the device," which is understood to mean "mount the filesystem on the device."

Mounting and unmounting filesystems usually requires root authority. If you are logged in as an ordinary user, you will either use su - to switch to root or sudo. In our examples, when the command prompt ends with #, as in the listing below, you will need root authority.

The basic form of the mount command takes two parameters: the device (or other resource) containing the filesystem to be mounted, and the mount point. We'll mount our small FAT32 partition /dev/sda3 at the mount point /dos. The mount point must exist before you mount anything over it. If it does not, you will get an error and need to create the mount point or use a different mount point.



To mount via an explicit filesystem type:



Displaying the current filesystem the command **mount** or **df –h** can be used. The output of both commands is:

~]$ **df -h**

Filesystem Size Used Avail Use% Mounted on

/dev/sda8 1.9G 668M 1.2G 37% /

/dev/sda7 1.9G 597M 1.3G 33% /opt

/dev/sda6 1.9G 41M 1.8G 3% /tmp

/dev/sda5 4.2G 846M 3.2G 21% /home

/dev/sda3 2.9G 2.3G 444M 84% /usr

/dev/sda2 2.9G 1.6G 1.2G 58% /var

/dev/sda1 46M 18M 26M 41% /boot

tmpfs 24G 0 24G 0% /dev/shm

/dev/mapper/vg01-lv\_wiki

9.9G 264M 9.1G 3% /wiki

/dev/mapper/vg01-lv\_permev

5.0G 1.7G 3.1G 36% /var/permev

10.92.37.74:/vol/luxnavf203\_v002/home\_users

1.1T 867G 234G 79% /home/users

10.92.37.69:/vol/defthw990xysto\_v001/defthw990xysto\_dsl

4.9T 4.7T 187G 97% /rep/emea

10.92.37.69:/vol/defthw990xysto\_v002/defthw990xysto\_transfer

1.0T 648G 377G 64% /transfer/emea

10.92.37.69:/vol/defthw990xysto\_v003/defthw990xysto\_unix2pb

150G 76G 75G 51% /unix2pb

10.92.37.144:/CC\_fs/ccinfo

100G 60G 41G 60% /var/log/CCINFO

~]$ **mount**

/dev/sda8 on / type ext3 (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

devpts on /dev/pts type devpts (rw,gid=5,mode=620)

/dev/sda7 on /opt type ext3 (rw)

/dev/sda6 on /tmp type ext3 (rw)

/dev/sda5 on /home type ext3 (rw)

/dev/sda3 on /usr type ext3 (rw)

/dev/sda2 on /var type ext3 (rw)

/dev/sda1 on /boot type ext3 (rw)

tmpfs on /dev/shm type tmpfs (rw)

/dev/mapper/vg01-lv\_wiki on /wiki type ext3 (rw)

/dev/mapper/vg01-lv\_permev on /var/permev type ext3 (rw)

none on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw)

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw)

10.92.37.74:/vol/luxnavf203\_v002/home\_users on /home/users type nfs (rw,nfsvers=3,rsize=65536,wsize=65536,namlen=255,hard,nolock,proto=tcp,timeo=600,retrans=2,addr=10.92.37.74)

10.92.37.69:/vol/defthw990xysto\_v001/defthw990xysto\_dsl on /rep/emea type nfs (rw,nfsvers=3,rsize=65536,wsize=65536,namlen=255,hard,nolock,proto=tcp,timeo=600,retrans=2,addr=10.92.37.69)

10.92.37.69:/vol/defthw990xysto\_v002/defthw990xysto\_transfer on /transfer/emea type nfs (rw,nfsvers=3,rsize=65536,wsize=65536,namlen=255,hard,nolock,proto=tcp,timeo=600,retrans=2,addr=10.92.37.69)

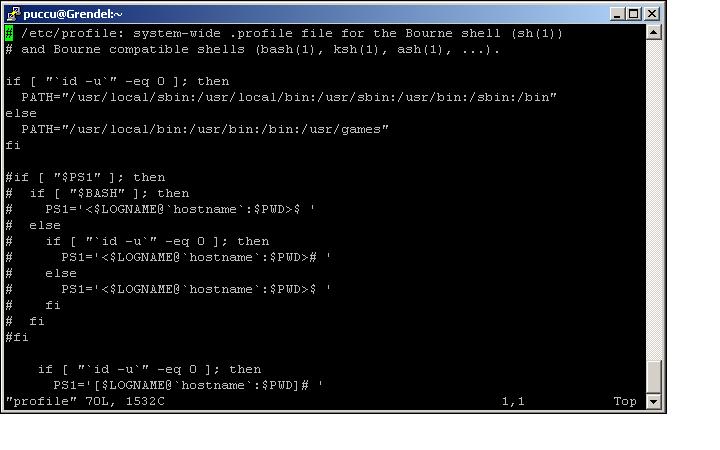
10.92.37.69:/vol/defthw990xysto\_v003/defthw990xysto\_unix2pb on /unix2pb type nfs (rw,nfsvers=3,rsize=65536,wsize=65536,namlen=255,hard,nolock,proto=tcp,timeo=600,retrans=2,addr=10.92.37.69)

10.92.37.144:/CC\_fs/ccinfo on /var/log/CCINFO type nfs (rw,nfsvers=3,rsize=65536,wsize=65536,namlen=255,hard,nolock,proto=tcp,timeo=600,retrans=2,addr=10.92.37.144)

**10. Editing tools: vi**

The advantage of learning **vi** and learning it well is that one will find **vi** on all Unix based systems and it does not consume an inordinate amount of system resources. Vi works great over slow network ppp modem connections and on systems of limited resources. One can completely utilize vi without departing a single finger from the keyboard. (No hand to mouse and return to keyboard latency).

Editing a file in vi is done by issuing the command: **vi file-to-edit.txt** (ex : **vi profile**)



The vi editor has three modes, command mode, insert mode and command line mode.

1. **Command mode:** letters or sequence of letters interactively command vi. Commands are case sensitive. The ESC key can end a command.
2. **Insert mode:** Text is inserted. The ESC key ends insert mode and returns you to command mode. From *Command mode* you can enter *Insert mode* with the "i" (insert), "a" (insert after), "A" (insert at end of line), "o" (open new line after current line) or "O" (Open line above current line) commands.
3. **Command line mode:** You enter this mode by typing ":" which puts the command line entry at the foot of the screen.

**Vi in Command Mode:**

|  |  |
| --- | --- |
| i | Switch to insert mode and type in your text |
| a | Append to current character position |
| A | Append to the end of the line |
| o | Jump to the beginning of a new below line |
| O | Jump to the beginning of a new above line |
| 0 | Jump to the beginning of the line |
| $ | Jump to the end of the line |
| 1 + G | Jump to the first line in a file |
| G | Jump to the last line of a file |
| 5 + G | Jump to the 5th line within a file |
| W | Jump a word to right |
| B | Jump a word to left |
| h | Move to left a character |
| l | Move to right a character |
| j | Move down a character |
| k | Move up a character |
| /key | Search key from up to down |
| ?key | Search key from down to up |
| yy | Copy the current line |
| 7 + yy | Copy 7 lines from the current posiont |
| p | Paste |
| u | Undo |
| Ctrl+r | Redo |
| dd | Delete current line |
| dw | Delete from current position to the end of word |
| db | Delete from current position to the beginning of the word |
| 3 + dd | Delete 3 lines |
| D | Delete from current position to the end of line |
| d$ | Delete from current position to the end of line |
| d0 | Delete from current position to the beginning of line |
| x | Delete one character at time |
| X | Delete everything before current position |
| r | Replace current character |
| R | Replace Mode -> ESC to return to Command Mode |
| ( | Move by sentence forward |
| ) | Move by sentence backword |
| { | Move by paragraph forward |
| } | Move by paragraph backword |

**Vi in Command Line Mode:**

|  |  |
| --- | --- |
| :q | Quite (if the file was not modified) |
| :wq | Write and quite |
| :w! | Force write |
| :q! | Quite without saving (if the file was modified) |
| :!command | Execute commands in shell and get output in vi |
| :r file | Reads from file |
| :w file | Writes in file |
| :split | Split the screen horizontally |
| :vsplit | Split the screen vertically |
| :close | Close a vi window |
| Ctrl+w | Move between vi windows |
| :sh | Escape to shell |
| :set nu | Set line numbering |
| :set nonu | Unset line numbering |
| :set ignorecase | Set vi to ignore case sensitive |
| :help | Get help |