A Beginners Handbook to GNU/Linux

A Booklet Prepared by

GNU/Linux User Group ,Trichy

for

Meet GNU/Linux '07

Contents

About GLUG-T

GNU/Linux User Group,Trichy or GLUG-T as it is fondly called is a loose conglomeration of individuals who got together because of one common love: the Linux operating system. However, we soon discovered that we had much more in common than Linux: the GNU and Free Software ideals, strong views about the privacy of the individual, and a desire to Change the World.

We are primarily bound together by our mailing list, with over 2000 members as of time of writing. It's fairly active, and instructions for subscription are on the link on the Home page if you want to join up. Don't worry if you're not in trichy , we have Linux users from all over the world subscribed to the list!

How to reach out ?

The best place to reach out to use is to use our

- Mailing List

Our mailing list has over 2000 subscribers and well interacted .You can subscribe to our mailing list by visiting the URI http://glugt.org/list
or more easily by sending an email to glug_t-requests@freelists.org and further instructions will be mailed to you .

- website http://glugt.org

It contains a wealth of information including a forum where you can post your questions , and also a wiki which will be used to aggregate knowledge .

Enjoy reading the booklet. Most of the content was taken from the web and then edited by a group of people. So its most likely that you find a lot of mistakes. Please bear with that. Please help us improve the booklet in any form by mailing to `glugt@glugt.org`

What is an Operating System ??

At the simplest level, an operating system does these things:

It manages the hardware and software resources of the system. In a desktop computer, these resources include such things as the processor, memory, disk space, etc.

It provides a stable, consistent way for applications to deal with the hardware without having to know all the details of the hardware.

The first task, *managing the hardware and software resources*, is very important, as various programs and input methods compete for the attention of the **central processing unit** (CPU) and demand memory, storage and input/output (I/O) bandwidth for their own purposes. In this capacity, the operating system plays the role of the good parent, making sure that each application gets the necessary resources while playing nicely with all the other applications, as well as husbanding the limited capacity of the system to the greatest good of all the users and applications.

The second task, *providing a consistent application interface*, is especially important if there is to be more than one of a particular type of computer using the operating system, or if the hardware making up the computer is ever open to change. A consistent **application program interface** (API) allows a software developer to write an application on one computer and have a high level of confidence that it will run on another computer of the same type, even if the amount of memory or the quantity of storage is different on the two machines.

Even if a particular computer is unique, an operating system can ensure that applications continue to run when hardware upgrades and updates occur. This is because the operating system and not the application is charged with managing the hardware and the distribution of its resources. One of the challenges facing developers is keeping their operating systems flexible enough to run hardware from the thousands of vendors manufacturing computer equipment. Today's systems can accommodate thousands of different printers, disk drives and special peripherals in any possible combination.

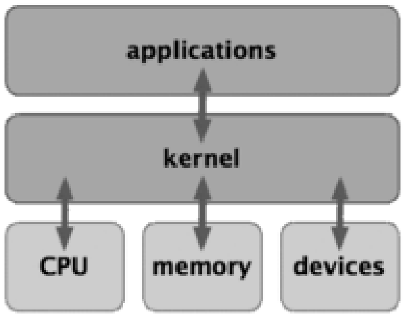


Fig-1 : Schematic diagram of components of an operating system.

The most popularly used operating systems are Linux OS, Windows and Macintosh, Sun Solaris.

The Linux kernel & operating system

Linux is a phenomenon of the Internet. Born out of the hobby project of a student it has grown to become more popular than any other freely available operating system. To many Linux is an enigma. How can something that is free be worthwhile? In a world dominated by a handful of large software corporations, how can something that has been written by a bunch of ``hackers'' hope to compete? How can software contributed to by many different people in many different countries around the world have a hope of being stable and effective? Yet stable and effective it is and compete it does. Many Universities and research establishments use it for their everyday computing needs. People are running it on their home PCs and I would wager that most companies are using it somewhere even if they do not always realize that they do. Linux is used to browse the web, host web sites, write theses, send electronic mail and, as always with computers, to play games. Linux is emphatically not a toy; it is a fully developed and professionally written operating system used by enthusiasts all over the world.

Most people use Linux as a simple tool, often just installing one of the many good CD ROM-based distributions. A lot of Linux users use it to write applications or to run applications written by others. Many Linux users read the HOWTO's avidly and feel both the thrill of success when some part of the system has been correctly configured and the frustration of failure when it has not. A minority are bold enough to write device drivers and offer kernel patches to Linus Torvalds, the creator and maintainer of the Linux kernel. Linus accepts additions and modifications to the kernel sources from anyone, anywhere. This might sound like a recipe for anarchy but Linus exercises strict quality control and merges all new code into the kernel himself. At any one time though, there are only a handful of people contributing sources to the Linux kernel.

Linux is an operating system that was initially created as a hobby by a young student, **Linus Torvalds**, at the University of Helsinki in Finland. Linus had an interest in **Minix**, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released. The kernel, at the heart of all Linux systems, is developed and released under the **GNU- General Public License** and its source code is freely available to everyone. It is this kernel that forms the base around which a Linux operating system is developed. There are now literally hundreds of companies and organizations and an equal number of individuals that have released their own versions of operating systems based on the Linux kernel. More information on the kernel can be found at our sister site, LinuxHQ and at the official Linux Kernel Archives. The **current full-featured version is 2.6** (released December 2003) and development continues.

Apart from the fact that its freely distributed, Linux's functionality, adaptability and robustness, has made it the main alternative for proprietary Unix and Microsoft operating systems. IBM, Hewlett-Packard and other giants of the computing world have embraced Linux and support its ongoing development. Well into its second decade of existence, Linux has been adopted worldwide primarily as a server platform. Its use as a home and office desktop operating system is also on the rise. The operating system can also be incorporated directly into microchips in a process called embedding and is increasingly being used this way in appliances and devices.

Throughout most of the 1990's, tech pundits, largely unaware of Linux's potential, dismissed it as a computer hobbyist project, unsuitable for the general public's computing needs. Through the efforts of developers of desktop management systems such as **KDE** and **GNOME**, office suite project **OpenOffice.org** and the **Mozilla web browser project**, to name only a few, there are now a wide range of applications that run on Linux and it can be used by anyone regardless of his/her knowledge of computers. Those choosing to using Linux can find a variety of versions or distributions of Linux that are easy to install, configure and use. A detailed description of various distributions will made further in the booklet.

A Small History

The GNU Project, with the goal of creating a UNIX-like operating system composed entirely of free software, had begun development in 1984, and a year later **Richard Stallman** had created the Free Software Foundation and wrote the first draft of the **GNU General Public License (GPLv1)**. By the early 1990s, the project had produced or collected many necessary operating system components, including libraries, compilers, text editors, and a Unix shell, and the upper level could be supplied by the X Window System, but development of the lower level, which consisted of a kernel, device drivers and daemons had stalled and was incomplete.

In 1991, **Linus Torvalds** began to work on the Linux kernel while he was attending the University of Helsinki. Torvalds originally intended Linux to be a non-commercial replacement for Minix, an educational operating system developed by **Andrew S. Tanenbaum**.

Code licensed under the GNU GPL can be used in other projects, so long as they too are released under the GPL. In order to make the Linux kernel compatible with the components from the GNU project, Torvalds changed his original license to the GPLv2. Linux and GNU developers worked to integrate GNU components with Linux. Thus Linux became a complete, fully functional free operating system.

Why Linux is Better ??

Forget about viruses

If your computer shuts itself down without asking you, if strange windows with text you don't understand and all kinds of advertisements appear when you don't ask for them, if emails get sent to all your contacts without your knowing it, then your computer probably has a virus.

Linux hardly has any viruses. Of course, a Linux virus is not impossible to get. However, Linux makes it very hard for this to happen, for several reasons.

Is your system unstable ?

Have you ever lost your precious work because your 'OS' crashed? Do you always shut down your computer the proper way, or do you sometimes just switch it off because your OS has gone crazy and doesn't let you do anything anymore? Have you ever gotten the "blue screen of death" or error messages telling you that the computer needs to be shut down for obscure reasons or worse “Restart our computer for blah blah ...”

Of course, no operating system is perfect, and people who tell you that theirs can never ever crash are lying. However, some operating systems can be so stable that most users never see their systems crash, even after several years. This is true for Linux. Here's a good way to see this. When a system crashes, it needs to be shut down or restarted. Therefore, if your computer can stay up and running for a long time, no matter how much you use it, then you can say the system is stable. Well, Linux can run for *years* without needing to be restarted (most Internet servers run Linux, and they usually never restart). Of course, with heavy updates, it still needs to be restarted (the proper way). But if you install Linux, and then use your system as much as you want, leaving your computer on all the time, you can go on like that for years without having any trouble.

Most of the time, you won't leave your computer on for such a long time, but this shows how stable Linux is.

Linux protects your computer

Open Source software (e.g. Linux) means more eyes to check the code. Every programmer on Planet Earth can download the code, have a look, and see whether it might have security flaws. On the other hand, the only people allowed to look at the Windows source code (its "recipe") are people working for Microsoft. That's hundreds of thousands of people (maybe millions) versus a few thousand. That makes a big difference.

But actually, it isn't exactly a matter of *how many* flaws a system has, compared to the others. If there are many flaws, but nobody has discovered them yet (including pirates), or they are minor (they don't compromise an important part of the system), pirates won't be able to do great damage. It is really a matter of *how fast a security flaw can be solved once it has been discovered*. If a security flaw is discovered in an open source program, anyone in the open source community can have a look and help solve it. The solution (and the update) usually appears within a few days, sometimes even a few hours. Microsoft doesn't have that much manpower, and usually releases security patches within about a month after the flaw has been discovered (and sometimes published): that's more than enough for pirates to do whatever they want with your computer.

Freedom!

Linux and "Open Source" software are "free". This means their license is a "free license", and the most common is the GPL (General Public License). This license states that anyone is allowed to copy the software, see the source code (the "recipe"), modify it, and redistribute it as long as it remains licensed with the GPL.

Too many windows? Use workspaces.

Once you have your word processor, your web browser, your email application, your instant messenger software and some windows open to explore your files, how do Windows users manage not to **get lost** in this clutter?

Workspaces is a feature I would never trade for anything else. You probably only have one screen, right? Try Linux, and you have four. Well, you can't actually look at the four of them at the same time, but this doesn't matter since your eyes can't look in two directions at once, right? On the first screen, lets put your word processor. On the second one, your instant messenger software. On the third one, your web browser. So when you're writing something in your word processor and you want to check out something on the web, no need to review all your windows to find your browser, stacked all the way behind the others. You just switch to your third screen and voilà, here it is.

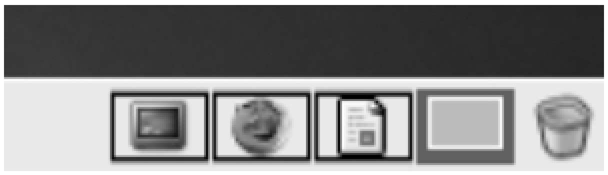


Fig : 2 – This one shows the four workspaces with a terminal,Firefox,open office and another application running all at the same time !!

Jump into the next generation of desktops:

Do you feel like your Desktop is featureless and boring ?? Tired of same old desktop with some wallpaper and some icons on them ??? Welcome to the word of 3-D desktops which will leave you mesmerized. Beryl, Compiz and many other

Desktop managers brings back life to your desktop and makes your work more enjoyable !! Whats better ?? You don't have to “upgrade” the system as you do generally when you have to add some more functionality to your present OS.

In fact, any recent computer with a decent processor, and a 512 MB RAM without even a graphics card is more than enough to have a good 3-D desktop running and it does not eat away your RAM or Processor !!

Let your old computer have a second life

Linux runs perfectly well on older hardware, on which Windows XP would probably even refuse to install, or let you wait 20 seconds after each click. Of course, Linux won't make a race-winner out of your 12-year old computer, but it will run very well on it and allow you to perform usual tasks (surfing the web, writing documents, etc.) just fine.

Are your tired of restarting your computer all the time?

Linux basically doesn't need to restart. Whether you install new software (even very big programs) or perform routine upgrades for your system, you will not be asked to restart the computer. It is only necessary when a part from the heart of the system has been updated, and that only happens once every several weeks.

Do you know Internet servers? They're the big computers that answer you when you ask for a web page, and send the information to your browser. Most of them run Linux, and since they need to always be available (a visitor could come anytime), they aren't restarted very often (services aren't available while the system is restarting). Actually, many of them haven't restarted for **several years**. Linux is stable, it runs perfectly well without restarting all the time.

You'll probably not let your computer on for several weeks but the point is:

the system won't bother you with restarting all the time.

So the point is not that we are not saying BAD about any other Operating system ‼ All operating systems have their own advantages and disadvantages. But Linux has really very few and gives you more worth for your money; in fact you have to spend less only for a decent computer configuration , which can run Linux at a really good speed which is of course free of cost ‼

Go to the last few pages of the booklet to find out equivalent and unique applications available in Linux which make your life a lot more easier. In fact, Linux has a damn lot of applications available freely from the Internet which can fulfill any of your need .

Installation of Linux operating system

Installation of Linux is a really Simple task once you know what to do. Its like same in other operating systems, creating hard disc partitions and formatting them and sorts.

There are numerous ways to use Linux with a computer. The most common method is to allocate part of your hard disk to Linux and put all the software you need on it. It's also possible to use Linux without touching your hard disk at all, either by getting the software from another computer on a network or by using a cd or dvd(known as Live CD / Live DVD). When people talk of installing Linux though, they invariably mean using the hard disk to store all their required software.

To install Linux on your hard disk, you first need to be able to allocate a section of the hard disk to Linux. Thankfully all sorts of computer systems understand the methods of dividing hard disks(nothing but portioning)

There might be some problems in case if your hardware uses proprietary software, but mostly nowadays the concerned companies themselves release patches or drivers , so that should not be a problem anymore.

Note for Beginners :

Dual Booting : Having both Windows and Linux in the same machine

You have heard so much about Linux, it's faster, more stable, cheaper, more efficient. You say, " I want to learn more but I don't want to loose my Windows files and applications... Don't despair. You can have both Linux, and Windows (and other operating systems as well, living on the same computer, even better, living on the same hard drive. This is really useful if you **DO** want to learn, and take the bother of switching to a better OS.

Windows and Linux can live comfortably on the same the same machine, even the same hard drive. The choice of operating systems can be made at the boot-up sequence when you throw the power switch. This configuration is known as the "dual-boot" configuration. This is how most Linux users start off ...

Preparation before the Installation:

There are a few things that we need to have before we can start. Obviously, we need a copy of Linux installation discs.

We need at least two partitions for installing Linux – one partition called SWAP partition which serves as virtual memory and another partition for the Linux file system to be installed.

The Swap partition should be twice the RAM size for optimum performance and the '/' or the root partition can be anywhere from 6GB-10GB (actually this depends upon the amount of packages you install) to how much ever you can allot. If you have a lot of Disc space you can allot and you want to try out many new things then simply create one more partition of say 10GB which can be mounted under '/home' where your documents and other files can be saved and '/' partition can be used only for softwares and other important things.

The file format of the Swap partition is “Linux swap space” or “SWAP” itself hence the name. And the file format of the '/' partition is either ext2 or ext3 . More on partitioning and file formats , check “How to Pages” at the end of the booklet.

Once you free up space to need , then you can create these partitions during Installation of Linux itself (Check the next section). All Linux installation discs now a days have a standard Partition Manager.

Partitioning, Formatting, and Installing Linux:

The Linux install is only slightly complicated, but is faster (~20min).Few things like partitioning , file format and Boot Loader might be confusing you very much in the beginning. But believe it, once you do a Linux installation yourself or twice you can explain anyone !!

Pop-in the Linux CD/DVD and restart your computer. Your system should to boot from CD-ROM, so the Linux installation screen should come up. You want to do just an install. When it asks you what kind of installation you would like to perform you want to select custom or expert (custom is the easier of the two, only use expert if you are sure that you know what you are doing!), since you'll need to tell Linux what partition to install on.

Setup will eventually ask you what program you want to use to partition your hard drive. Select Manually Set the partition table or a similar option.

WARNING: Don't select default partitioning. This might format your complete hard disc as it will warn.

Once you get into the manual partition setup, Linux will recognize all your other

Operating System partitions and free or unpartitioned space. If you have not done the partitioning yet, this is the place you have to do it . Now create a new partition for double the size of your RAM, say 1 GB for a 512 MB ram and select the file format as SWAP. We will see what SWAP is sooner. Then format the other free space as 'ext3' and it mount it under '/ '. If you want more space to be added under Linux partition, then format that also 'ext3' and mount it under '/home' .

Once this is done, you can continue through the rest of the setup, just make sure that when you reach the point of what Boot loader you want to use, **pick GRUB and make sure it installs to the MBR(Master Boot Record)**.Most of the recent Installers detect your other operating system like Windows Installation and it will show up in while Installing Linux itself. This will prompt you to choose the Operating system you want to work on during boot up instead of just directly booting into windows as such. More on Grub can be found at “How to pages” at the end of the booklet.

Once your done picking and choosing what you want installed along with the Linux OS, setup will do the rest for you. Make sure you make a user account for yourself other than root when prompted by the Linux setup, because when you are logged into the root account you can do anything and everything to the partition Linux resides on, including **DELETING THE WHOLE PARTITION** by accident.

Desktop Applications

As people are quite familiar with Windows and its popular applications, it is useful to list out the equivalent applications used in Linux. The Application equivalents are enumerated below and as such there is a small difference. The applications in Windows mostly have to be installed afterwards, but most of the Linux packages are available freely and mostly can be installed while installing Linux itself by default.

	Windows Applications	Linux Applications

1) **INTERNET**

Web Browser	Internet Explorer Netscape	**Mozilla Firefox** Konquerer Nautilus Epiphany galeon
E-Mail Clients	Outlook Express Opera Eudora Lite	**Evolution** Kmail Gnus Gnumail Aethera Liamail
Address Book	Outlook	Rubrica Gnome-addressbook
Download Agents	FlashGet Gozilla Getright Download Accelerator Plus	Downloader for X Caitoo Prozilla **Wget** httrack Aria Axel

		ktorrents
FTP Clients	SmartFTP CuteFTP	**Gftp** Kbear Igloo FTP Nftp tkFTP
Instant Messaging Clients	MSN Messenger Yahoo Messenger Google Talk	Simple Instant Messenger aMSN Yahoo Messenger for Unix **GAIM(for all google, xchat and other chats)** AIM Kmess xchat
P2P Clients	Morpheus Napster eDonkey Bittorrent	ML Donkey **Limewire** Lopster Gtk-Gnutella GNU-Net Qtella Mutella Xmule Snark

2) Working With Files

File Compression	Winzip Winrar	**Gzip, bzip,zip** Ark (kdeutils).

	7Zip Zip Genius	Gnozip Karchiveur Gnochive **FileRoller** Unace TkZip

3) Desktop / System Software

Text Editor	Notepad Wordpad Textpad	Kedit **Gedit** Gnotepad Kate **Vim** **Xemacs** **emacs** Xcoral
PDF Creaters/Viewers	Adobe Acrobat Distiller	Tex2pdf ReportLab GGV Panda PDF Generator **Evince** **kPdf** xPdf GhostView
Antivirus	AVG AntiVirus NAV	“as such not required”

	Dr. Web Kaspersky	
Hard disk partitions manager	PowerQuest Partition Magic Acronis Partition Expert Paragon Partition Manager	PartGUI **GNU Parted - gparted** Partition Image Diskdrake Paragon Partition Manager

4) **Multimedia**

MP3 Players	Windows Media Player Winamp RealPlayer VLC Media Player	**XMMS** **Banshee** **Xine** **Amarok** Xamp **Mplayer** **VLC Media Player**
Video Players	VLC Media Player Windows Media Player RealPlayer	**Mplayer** **VLC Media Player** **Totem Video Player** **Xine** Aviplay Noatun Xmps KDE Media Player
CD Burning	Nero Roxio Easy CD Creator	**K3b** XCDRoast KonCD

		Gnome Toaster Kreate-CD dd <command prompt>
Audio Editors	SoundForge Cooledit Audacity	**Audacity** Glame Rezound Sweep **WaveForge** GNUSound
DVD Players	PowerDVD WinDVD MicroDVD VLC Media Player	Ogle **Mplayer** **Xine** Aviplay **VLC Media Player** OMS
Video Creation and Editing	Windows Movie Maker Sony Vegas	**Avidemux** MainActor iMira Editing

5) **Graphics**

Graphic Editors	Adobe Photoshop Macromedia Fireworks Corel PhotoPaint	**GIMP** **ImageMagick** Pixel32 CinePaint RubyMagick
3D Graphics	D Studio MAX Maya	Blender KpovModeler

	Povray	K3Studio Moonlight **Wings 3D**

6) Office

Office Suite	MS Office Star Office	**Open Office** Star Office Koffice HancomOffice Gnome office Crystal Office

7) **Programming**

Programming Tools and IDE	Notepad ++ TurboC Dev CPP Net Beans	**GCC** **Eclipse** **Anjuta** **Gedit** **Net Beans** PythonIDE **Emacs**

8) **Servers**

Web Servers	IIS	**Apache** Xitami Caudium Roxen Zeus

FTP Servers	War FTP ServU	Pure-ftpd wu-ftpd proftpd **vsftpd**
Database Engines	MS SQL MySQL Oracle	**MySQL** **PostgreSQL** **Oracle** Linter SAP DB

The File Hierarchy

Introduction

The first thing that most new users shifting to Linux will find confusing is navigating the Linux file system. The Linux file system does things a lot more differently than the Windows file system.

For starters, there is only a single hierarchal directory structure. Everything starts from the root directory, represented by '/', and then expands into sub-directories. Where DOS/Windows had various partitions and then directories under those partitions, Linux places all the partitions under the root directory by mounting them under specific directories. Closest to root under Windows would be C:

Under Windows, the various partitions are detected at boot and assigned a drive letter. Under Linux, unless you mount a partition or a device, the system does not know of the existence of that partition or device. This might not seem to be the easiest way to provide access to your partitions or devices but it offers great flexibility.

This kind of layout, known as the *unified file system*, does offer several advantages over the approach that Windows uses.

Directory Structure

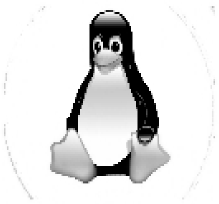
- **`/sbin`** - This directory contains all the binaries that are essential to the working of the system. These include system administration as well as maintenance and hardware configuration programs. These are the essential programs that are required by all the users. Another directory that contains system binaries is */usr/sbin*. This directory contains other binaries of use to the system administrator.
- **`/bin`** - In contrast to /sbin, the bin directory contains several useful commands that are used by both the system administrator as well as non-privileged users. This directory usually contains the shells like bash, csh etc. as well as much used commands like cp, mv, rm, cat, ls. There also is /usr/bin, which contains other user binaries. These binaries on the other hand are not essential for the user. The binaries in /bin however, a user cannot do without
- **`/boot`** - This directory contains the system.map file as well as the Linux kernel. The
- **`/dev`** - This is a very interesting directory that highlights one important characteristic of the Linux filesystem - everything is a file or a directory. Look through this directory and you should see hda1, hda2 etc, which represent the various partitions on the first master drive of the system. /dev/cdrom and /dev/fd0 represent your CDROM drive and your floppy drive.

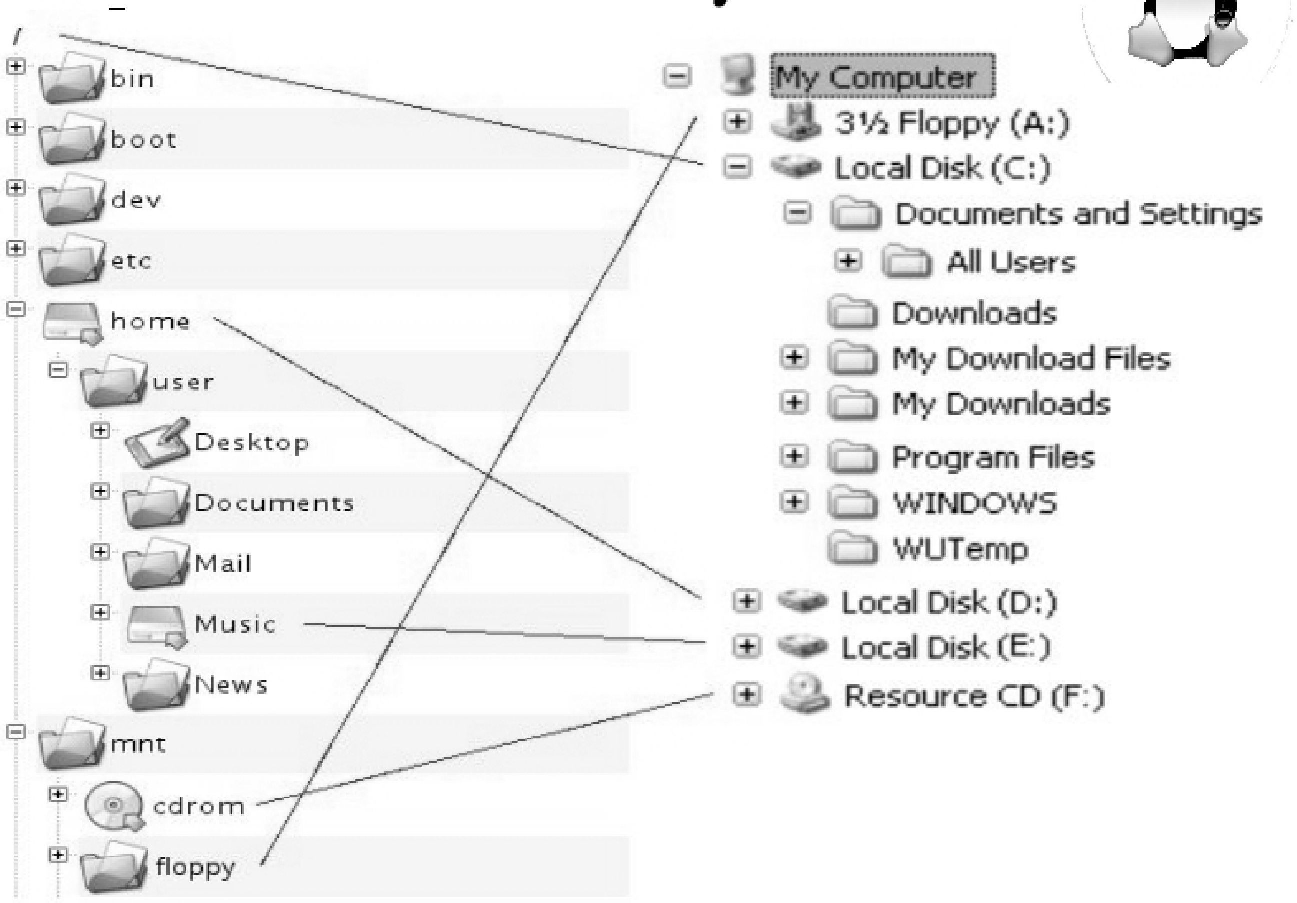
Take /dev/dsp, for instance. This file represents your speaker device. So any data written to this file will be re-directed to your speaker.

- **`/etc`** - This directory contains all the configuration files for your system. Your hosts, resolv.conf and fstab. Under this directory will be X11 sub-directory which contains the configuration files for X. More importantly, the /etc/rc.d directory contains the system startup scripts.
- **`/home`** - Linux is a multi-user environment so each user is also assigned a specific directory which is accessible only to them and the system administrator. These are the user home directories, which can be found under /home/username.
- **`/lib`** - This contains all the shared libraries that are required by system programs. Windows equivalent to a shared library would be a DLL file.
- **`/lost+found`** - Linux should always go through a proper shutdown. Sometimes your system might crash or a power failure might take the machine down. Either way, at the next boot, a lengthy filesystem check using Fsck will be done. Fsck will go through the system and try to recover any corrupt files that it finds. The result of this recovery operation will be placed in this directory.
- **`/mnt`** - This is a generic mount point under which you mount your filesystems or devices. Mounting is the process by which you make a filesystem available to the system. After mounting your files will be accessible under the mount-point. This directory usually contains mount points or sub-directories where you mount your floppy and your CD.
- **`/opt`** - This directory contains all the software and add-on packages that are not part of the default installation. Generally you will find KDE and StarOffice here. Again, this directory is not used very often as its mostly a standard in Unix installations.

- **`/proc`** - This is a special directory on your system.
- **`/root`** - We talked about user home directories earlier and well this one is the home directory of the user root. This is not to be confused with the system root, which is directory at the highest level in the filesystem.
- **`/tmp`** - This directory contains mostly files that are required temporarily. Many programs use this to create lock files and for temporary storage of data.
- **`/usr`** - This is one of the most important directories in the system as it contains all the user binaries. X and its supporting libraries can be found here.
- **`/var`** - This directory contains spooling data like mail and also the output from the printer daemon. The system logs are also kept here in /var/log/message

Unix directory structure





File Systems: On the left is a typical Linux file system, and on the right is Windows' Explorer. The Linux file system consists of one "tree" with each drive attached to that tree and acting like a folder. The Windows file heirarchy consists of each physical drive having its own seperate file system. The lines between the two point to where these drives appear in each graphic.

Basic Shell Operation

Intoduction – What is a shell ??

The **Shell** is a powerful tool of the GNU/Linux operating system. The shell is an interface between the user and the linux kernel. Long before the GUI - graphical user interfaces became common, the shell (variously called "the the command prompt", CLI (command line interface), console prompt) was the only tool by which the user could instruct the computer.

Now kernel is simply a set of routines in C which interacts with the hardware and handles user interrupts (passes the informations back and forth between the user and the hardware seamlessly). Now the kernel is the heart of the Linux OS . But the user cannot directly work with the kernel as its too low level and requires good knowledge of assembly and C language.

Thats where the shell comes in. The shell can be thought as an interpreter which has certain easy to remember commands which acts between the kernel and the user and the appropriate action is performed . So if you know how to work in a shell , thats means you have the whole computer under your control starting from as simple as creating your files, editing and removing your files to starting processes, stopping(killing processes) and process management to as complicated and dangerous as erasing your whole hard disc. In fact the whole hard disc can be erased by *one* single command which make shell so powerful and yes, a bit dangerous !

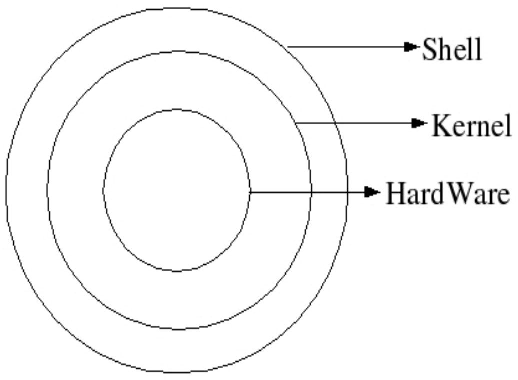


Fig : shell – kernel – hardware

There are actually many types of shell like BASH, TSH, CSH so on. The default shell is BASH – Bourne Again Shell

Why Shell?

1. Why should I learn to use the shell? Can't I do everything through the GUI?

It is true that the GUI is itself quite powerful, and using the shell requires a fair amount of typing, but once you get used to it, you will find that for many things the shell is far more superior. For example, using regular expressions, one can make complex selections of files, which cannot be done with the mouse.

2. Aarrgh! I hate typing on this keyboard! It is easy to double click rather than typing "cd my_dir/"

You don't really need to type too much on the shell, because most Linux shells will "autocomplete" file/command names. Typing is painful, if you are not used to it/have a horrible keyboard. The best solution is to practice/get a better keyboard.

3. Do I have to mug all these commands?

No, not at all. You can use them when you need them. If you are searching for any command, just google for common shell commands. You'll remember most of the useful commands once you begin using them.

First steps into shell :

When the user first logs into his account he is by default logged into his home directory as the working directory. Say if the account logged in is *"mgl"* the the default location is *"/home/mgl"* and that place is the home directory of that user. So now the /home/mgl folder is the place where he can create files, edit them, remove them, create, edit, remove directories and all other things. At any given point of time after logging in, a user at a terminal will have a "current working directory"(which is his home directory immediately after login). There is only one current directory for a user, but many users can be present at a directory at the same time.

Shell commands.

Command [options] [arguments]

This is the way all the shell commands will be like. A *command keyword* followed by *options* that command can take and the *arguments* on which the command has to act. Not all commands have/need options and arguments. But as such this is the default syntax.

Now you will type these commands in a shell prompt or command prompt like this

[mgl@glugt [~] $

This means this a shell prompt of the user *mgl* in the computer *glugt* and '~' signifies the *mgl's* Home directory and it actually the current working directory. Suppose the user was in /home/folder/hello somehow then it would look like *[mgl@glugt [home/folder/hello] $*

Now we know what a shell is and what a shell prompt is . Lets learn some commands to type in at that shell !!

The very first command we will see is "*ls*" - which will list the current working directory contents and return to the shell prompt. The ls command takes many options like `ls -l` : displays the contents of the directory in long format giving details like permissions , size in bytes, date and time of creation of the file/directory and the file name.
`ls -R` : displays the contents of the current recursively (ie) id displays the contents of sub-folders and thier sub-folders and so on .

To get the full list of the options check '*`man ls`*' where you can get everything you want to know about the command.

`echo <some text>` : write some text on your screen

`man` - to go to the manual pages. "man cd" will open the manual pages of the command cd.

`ls` - to list files. "man ls" will give you a list of the options available with ls.

`cd` - to navigate between directories. "`cd /var/`" will take you to the directory /var.

`cd ..` will take you one directory up in the linux file system tree.

`cat` - to read a file. "cat myfile" will output the file at the terminal.

`pwd` - to print the current working directory.

`mkdir` - to create a new directory inside the current directory. "mkdir newdir" will create a new directory named newdir.

`Rmdir` – removes a directory if its empty

`mv` - i) to move a file from a directory to another; ii) to rename a file. "`mv file1 file2`" will rename file1 to file2. "`mv file1 /home/tom/`" will move the file file1 from the current directory to the directory /home/tom.

`rm` - to remove files. "`rm file1`" will delete a file file1 from the current directory, once you confirm that you really want to delete the file, by entering "y" when you're asked to confirm.

`cp <source-destination> <final -destination>` - to copy files from source destination to the final destination .

`mv` `<source-destination> <final -destination>` - to moves files from source destination to the final destination .Effectively can be used for renaming also.

`who,finger` - to list all the users currently logged in.

`passwd` - to change your password.

`touch` - creates a new blank file. "touch a" will create a new file called "a".

`more , less` – to view large files one screen at a time

`head<file>, tail <file>`– displays the first and last 10 lines of the file respectively.

There are a lot more commands in the shell cheat sheet part though we will mostly need only the following shell commands only regularly.

Users and Permissions :

Linux is strict about the file permissions. You cannot access or edit any file as such if you don't have the permissions to do so.

There are three levels of permissions namely : read, write and execute. And there are three categories within which all the users are classified as : the owner of the file, the group to which the file owner belongs and to others. So we can specify different permissions for each and every user category.

For a file the permissions are meant litreally – read , write or execute them. But for a directory permissions are a bit different : read means read the contents of the directory, write means ability to edit/manipulate the contents of the directory and execute implies for all the actions or commands on the directory.

Setting the permissions:

The permissions are denoted both numerically and alpha-numerically as follows :

```
read : 4 or r
write: 2 or w
execute: 1 or x
```

So if a user has the permission to read and write a file, then he would have '6' denoted as his permission for that file. And so on.

Now to change or set the permissions we use a command `chmod` . Suppose we have a file named

`chmod u+rwx <file name>` : adds read, write and execution permissions to the owner

`chmod g-w <file name>` : removes the write permission for group `chmod o=r <file name> :` sets read permissions for others.

So for `rwx = 4+2+1 =7, rx = 4+1 = 5` and so on can be used for numerical values for permissions.

Numerically,

`chmod 755 <file-name>` : sets all 3 permissions for owner , read and execute permissions for the group and others.

There are also commands like `chown` and `chgrp` which changes the ownership and group of the files.

For eg : `chown <user-name> <file-name>`

changes the ownership of file to the user

`chgrp <group-name> <file name>`

changes the group to which the file belongs.

All `chmod` , `chown` and `chgrp` take the option -r for recursively setting the permissions to all folders and files inside a directory.

Intermediate Shell Operations:

Pipes, redirection and backtick

They are not really commands but they are very important concepts.

Pipes :

'|' send the output (stdout) of one program to the input (stdin) of another program.

```
grep "hello" file.txt | wc -l
```

finds the lines with the string hello in file.txt and then counts the lines.

The output of the grep command is used as input for the wc command. You can concatinate as many commands as you like in that way (within reasonable limits).

Redirection:

writes the output of a command to a file or appends data to a file

> writes output to a file and overwrites the old file in case it exists

>> appends data to a file (or creates a new one if it doesn't exist already but it never overwrites anything).

`ls > a` : will actually redirect all the output of the ls command to the file named 'a'.

`ls>>a` : append the output of the ls command to the file named 'a'

Backtick:

The output of a command can be used as command line arguments (not stdin as above, command line arguments are any strings that you specify behind the command such as file names and options) for another command. You can as well use it to assign the output of a command to a variable.

The command

ls and echo gives the contents of the directory and prints text some text respectively. ls will actually give the ouput one per line.

But when you use a backtick like this

```
echo `ls`
```

the whatever is the output of ls is the argument of the echo command. Its some what similar to piping in function.

Archiving :

`tar` is an archive of files created to transfer an organised version of a list of files to be transferred in a network. A tar file when zipped produces what is called a **`tar ball(tar.gz)`** . One main advantage of taring files into an archive is that the size occupied by the files is lesser than when directly zipped.

One can create a tar archive archive by using the **tar** command with -cf as options. Eg:- `tar -cf archive.tar file1 file2 file3` creates a tar file with the files as file1 file2 file3 named archive.tar. Eg:- To store all .cpp files in a tar named CPP.tar one can do

```
tar -cf archive.tar *.cpp
```

To extract files from a tar, one can

```
tar -xf archive.tar
```

To just know the names of the files in the tar archive one can do

```
tar -tvf archive.tar
```

to list all the files.

Zipping :

Many of you might be familiar with zipping , It is just a utility available in most OS for compressing the files.

There are two types of zipping formats namely **GNUzip** (gz) , **bzip2**(bz2) and the common **zip** file (.zip).

To create a .zip file use `zip <filename>`.

One can gzip a file as *`gzip <filename>`*. A file named filename.gz is created.

Bzip2(.bz2) is another type of zipping format. Command is *`bzip2 <filename>`* .

To Unzip a .gz file one can either use *`gunzip`* or *gzip -d* with the appropriate filename. Similarly to unzip a .bz2 file one can use *`bunzip`* .

Processes :

What is Processes:

Process is any kind of program or task carried out by your PC. For e.g. `ls -lR` , is command or a request to list files in a directory and all subdirectory in your current directory. It is a process. A process is program (command given by user) to perform some Job. In Linux when you start process, it gives a number (called PID or process-id), PID starts from 0 to 65535.

Why Process required :

Linux is multi-user, multitasking o/s. It means you can run more than two process simultaneously if you wish. For e.g.. To find how many files do you have on your system you may give command like

```
ls / -R | wc -l
```

This command will take lot of time to search all files on your system. So you can run such command in Background or simultaneously by giving command like

```
ls / -R | wc -l &
```

The ampersand (&) at the end of command tells shells start command `ls / -R` and run it in background takes next command immediately. An instance of running command is called process and the number printed by shell is called process-id (PID), this PID can be use to refer specific running process.

Linux Command Related with Process:

`ps` Shows all the process currently running started by the user in the current login. With process id-s

`ps -A or ps -e` Shows all system processes.

`Top -` Shows all operations in decreasing order of memory and process usage.

Provides options to kill process with thier id's also

`kill <PID>–` kill processes when the process-id is given

`killall <process-name> :` kills processes when the process-name is given

Advanced Shell Operation:

Process and process manipulation:

Filters

Filters are commands which accept input from the standard input, manipulate it and print the result at the standard output. Redirection and pipelining can be combined with filters to provide more functionality. Some of the most useful filters are listed here.

Cut **-**

To slice a file vertically, with tab as the default delimiter. Suppose a file called mylist contains the following data.

```
cs10501 12 ; 5.6
cs10502 14 ; 5.7
cs10503 12 ; 5.8
cs10504 13 ; 5.9
cs10505 11 ; 5.1
```

The command "`cut -c 3-7 mylist`" will output what's below.

```
10501
10502
10503
```

```
10504
10505
```

Output ends with the line before this one. Columns are numbered starting from 1 and the "-c" option has to be followed by a list of columns you want printed. All these commands are valid: "`cut -c 1-10 mylist`", "`cut -c 1-10,13-15 mylist`" and "cut -c 3 mylist".

The "-d" option lets you specify the delimiter. It must be used along with "-f". The list of fields to be printed follows "f". The command "cut -d ';' -f 1 mylist" will print the following.

```
cs10501 12
cs10502 14
cs10503 12
cs10504 13
cs10505 11
```

sort

to order a file. The command "`sort mylist`" sorts the file mylist. The command sort has several options.

`-t char` : uses char as the delimiter

`-k n` : sorts on the ntht field

`-k m,n` : starts sort on mth field and ends on nth field

`-u` : removes repeated lines

`-n` : sorts numerically

`-r` : reverses sort order

`-f` : converts lowercase letters to equivalent uppercase ones

`-c` : checks if the file is sorted

`-o fname` : writes output into fname

tr – to translate individual characters.

Syntax : tr 'list-of-characters-to-be-changed' 'list-of-characters-to-replace-them-with'

"`tr '; '  '|*' < mylist`", without the quotes, will output the following.

```
cs10501*12*|*5.6
cs10502*14*|*5.7
cs10503*12*|*5.8
cs10504*13*|*5.9
cs10505*11*|*5.1
```

Semicolons in the original file are replaced with a pipeline symbols and spaces are replaced with asterisks in the output.

Note that cut or tr do not change the original file unless the output is redirected to the original file. In the about example, input to "tr" is given from the file mylist through a redirection operator and the output is printed at the standard output.

Redirection and Pipelining

Filters take input from the standard input but input can be supplied to them through files too. The output of a filter can also be supplied as the input to a filter. This is achieved by using redirection and pipelining. The redirection operators are < and >. The command "file1 > file2" simply makes a copy of file1 called file2. The command "date" prints the date at the terminal. The command "date > mydate" writes the date into a file and names it mydate. If a file called mydate already exists, it will be overwritten. If '>' and '<' are replaced with '>>' and '<<' respectively and if the file already exists, the new content is appended to its end. The command "`cut -c 1-10 < mylist`" does the same as the command "`cut -c 1-10 mylist`".

The output of a command/filter can be given as input to another command/filter using pipes. A pipe is a "|". The command "date" outputs something like "Thu Jul 19 22:34:41 IST 2007 ".

"`date|cut -c 1-10`" outputs "`Thu Jul 19`"

(columns one to ten). Here, the output of the date command is fed as input to the cut command through the pipeline.

Shell Scripting

A shell script is a set of commands written in a text file. The file mostly has ".sh" as its extension with execute permissions but shell scripts can have any/no extension and be run even without an execute permission.

- Can take inputs from the user or a file and output them.
- Useful to create our own commands.
- Save lots of time.
- Can be used to automate some tasks in day-to-day life.
- System Administration can be automated.

Shell variables:

Sometimes to process our data/information, it must be kept in computers RAM memory. RAM memory is divided into small locations, and each location had unique number called memorylocation/address, which is used to hold our data. Programmer can give a unique name to this memory location/address called memory variable or variable (Its a named storage location that may take different values, but only one at a time). In Linux, there are two types of variable

1. System variables - Created and maintained by Linux itself. This type of variable defined in CAPITAL LETTERS.
2. User defined variables (UDV) - Created and maintained by user. This type of variable defined in lower LETTERS.

You can see system variables by giving command like $ set, Some of the important System
variables are

System Variable	Meaning
`BASH=/bin/bash`	shell name
`BASH_VERSION=1.14.7(1)`	shell version name
`COLUMNS=80`	No. of columns for our screen
`HOME=/home/mgl`	Our home directory
`LINES=25`	No. of columns for our screen
`LOGNAME=mgl`	logging name
`OSTYPE=Linux`	o/s type
`PATH=/usr/bin:/sbin:/bin:/usr/sbin`	Our path settings
`PS1=[\u@\h \W]\$`	prompt settings
`PWD=/home/students/Common`	current working directory
`SHELL=/bin/bash`	shell name
`USERNAME=mgl`	User name who is currently login to this PC

NOTE that Some of the above settings can be different in your PC. You can print any of the above variables contain as follows

```
$ echo $USERNAME
$ echo $HOME
```

Caution: Do not modify System variable this can some time create problems.

How to define User defined variables (UDV)

To define UDV use following syntax

Syntax: `variablename=value` (no spaces in between)

NOTE: Here 'value' is assigned to given 'variablename' and Value must be on right side = sign For

e.g.

```
$ no=10              # this is ok
$ 10=no              # Error, NOT Ok, Value must be on right side of = sign.
```

To define variable called 'vech' having value Bus

```
$ vech=Bus
```

To define variable called n having value 10

`$ n=10`

How to print sum of two numbers, let's say 6 and 3

```
$ echo 6 + 3
```

This will print 6 + 3, not the sum 9, To do sum or math operations in shell use expr, syntax is as

follows Syntax: expr op1 operator op2

Where, op1 and op2 are any Integer Number (Number without decimal point) and operator can be

```
+ Addition
–  Subtraction
* multiplication
/ Division
% Modular,
```

to find remainder For e.g. 20 / 3 = 6 , to find remainder 20 % 3 = 2, (Remember its integer calculation)

```
$ expr 6 + 3
```

Now It will print sum as 9 , But

```
$ expr 6+3
```

will not work because space is required between number and operator (See Shell Arithmetic)

Writing a Shell script

Shell sricpt is nothing put putting more shell commands together and with some looping constructs like for and conditional statements. Thats all.!! Let us consider a simple shell script which takes a backup of your home folder and stores the backup in a different location.

1. Write the shell commands in the script, one in a line using your favourite editor. Let the file have a ".sh" extension. Example - backup.sh.

2. Set execute permission for your script

```
user@localhost:~/chmod a+x backup.sh
```

or

```
user@localhost:~/chmod 755 backup.sh
```

3. Execute your script

```
user@localhost:~/ sh backup,sh
```

or

```
user@localhost:~/ ./backup,sh
```

What to write in backup.sh?

Let us assume that you want to take a backup of all the files in your home folder and store it in a different place (say /backups).

The command "`zip /backups/backup.zip -r ~/`" will create a backup of your home folder called backup.zip in /backups. But no, it's a boring name to give to a backup file. If you'd read through the section on the commands cut, tr and pipelining, you'll know that the command "date|cut -d ' ' -f 2,3,6" will output "Jul 19 2007", for example. This can be fed as input to the "tr" command, to replace the spaces with underscores and it's done by

```
date|cut -d ' ' -f 2,3,6|tr ' ' '_'
```

Jul_19_2007 will be a nice name for a backup file. Whenever this script is run, a backup with a similar name will be created. The command to do this is

```
zip `date|cut -d ' ' -f 2,3,6|tr ' ' '_'`.zip ~/
```

Now, write this command in a file and save it as backup.sh. It can be run as ./backup.sh. You can create a backup of your home folder whenever you want by just running this.

Shell Cheat Sheet

Here is a big but not an exhaustive list of shell-commands . As such not even 10% of these Shell commands is used regularly.Still its here just for a reference. Only developers, Programmers and others who need a better control and functionality use many of the shell commands. And as such there is no necessity to remember the attributes and other options for each and every command. Just *man <the command you want to know more about>.* In fact try ' *man man* ', you will come to know more about manual pages better known as man pages. With practice and regular use remembering these commands are very simple!!

A

- *alias* Create an alias
- *awk* Find and Replace text within file(s)

B

- *break* Exit from a loop
- *builtin* Run a shell builtin

C

- *cal* Display a calendar
- *case* Conditionally perform a command
- *cat* Display the contents of a file
- *cd* Change Directory
- *chgrp* Change group ownership
- *chmod* Change access permissions
- *chown* Change file owner and group
- *chroot* Run a command with a different root directory
- *cksum* Print CRC checksum and byte counts
- *clear* Clear terminal screen
- *cmp* Compare two files
- *comm* Compare two sorted files line by line
- *command* Run a command - ignoring shell functions
- *continue* Resume the next iteration of a loop
- *cp* Copy one or more files to another location
- *cron* Daemon to execute scheduled commands
- *crontab* Schedule a command to run at a later time
- *csplit* Split a file into context-determined pieces
- *cut* Divide a file into several parts

D

- *date* Display or change the date & time

- *dc* Desk Calculator
- *dd* Data Dump - Convert and copy a file
- *declare* Declare variables and give them attributes
- *df* Display free disk space
- *diff* Display the differences between two files
- *diff3* Show differences among three files
- *dir* Briefly list directory contents
- *dircolors* Colour setup for `ls`
- *dirname* Convert a full pathname to just a path
- *dirs* Display list of remembered directories
- *du* Estimate file space usage

E

- *echo* Display message on screen
- *ed* A line-oriented text editor (edlin)
- *egrep* Search file(s) for lines that match an extended expression
- *eject* Eject CD-ROM
- *enable* Enable and disable builtin shell commands
- *env* Display, set, or remove environment variables
- *eval* Evaluate several commands/arguments
- *exec* Execute a command
- *exit* Exit the shell
- *expand* Convert tabs to spaces
- *export* Set an environment variable
- *expr* Evaluate expressions

F

- *factor* Print prime factors
- *false* Do nothing, unsuccessfully
- *fdformat* Low-level format a floppy disk
- *fdisk* Partition table manipulator for Linux
- *fgrep* Search file(s) for lines that match a fixed string
- *find* Search for files that meet a desired criteria
- *fmt* Reformat paragraph text
- *fold* Wrap text to fit a specified width.
- *for* Expand words, and execute commands
- *format* Format disks or tapes
- *free* Display memory usage
- *fsck* Filesystem consistency check and repair.
- *function* Define Function Macros

G

- *gawk* Find and Replace text within file(s)
- *getopts* Parse positional parameters
- *grep* Search file(s) for lines that match a given pattern
- *groups* Print group names a user is in
- *gzip* Compress or decompress named file(s)

H

- *hash* Remember the full pathname of a name argument
- *head* Output the first part of file(s)history Command History
- *hostname* Print or set system name

I

- *id* Print user and group idâ€™s
- *if* Conditionally perform a command
- *import* Capture an X server screen and save the image to file
- *info* Help info
- *install* Copy files and set attributes

J

- *join* Join lines on a common field

K

- *kill* Stop a process from running

L

- *less* Display output one screen at a time
- *let* Perform arithmetic on shell variables
- *ln* Make links between files
- *local* Create variables
- *locate* Find files
- *logname* Print current login name
- *logout* Exit a login shell
- *lpc* Line printer control program
- *lpr* Off line print
- *lprint* Print a file
- *lprintd* Abort a print job
- *lprintq* List the print queue
- *lprm* Remove jobs from the print queue

- *ls* List information about file(s)

M

- *m4* Macro processor
- *man* Help manual
- *mkdir* Create new folder(s)
- *mkfifo* Make FIFOs (named pipes)
- *mknod* Make block or character special files
- *more* Display output one screen at a time
- *mount* Mount a file system
- *mtools* Manipulate MS-DOS files
- *mv* Move or rename files or directories

N

- *nice* Set the priority of a command or job
- *nl* Number lines and write files
- *nohup* Run a command immune to hangups

P

- *passwd* Modify a user password
- *paste* Merge lines of files
- *pathchk* Check file name portability
- *popd* Restore the previous value of the current directory
- *pr* Convert text files for printing
- *printcap* Printer capability database
- *printenv* Print environment variables
- *printf* Format and print data

- *ps* Process status
- *pushd* Save and then change the current directory
- *pwd* shows Present Working Directory*

Q

- *quota* Display disk usage and limits
- *quotacheck* Scan a file system for disk usage
- *quotactl* Set disk quotas

R

- *ram* ram disk device
- *rcp* Copy files between two machines.
- *read* read a line from standard input
- *readonly* Mark variables/functions as readonly
- *remsync* Synchronize remote files via email
- *return* Exit a shell function
- *rm* Remove files
- *rmdir* Remove folder(s)
- *rpm* Remote Package Manager
- *rsync* Remote file copy (Synchronize file trees)

S

- *scp* Secure copy (remote file copy)
- *screen* Terminal window manager
- *sdiff* Merge two files interactively
- *sed* Stream Editor
- *select* Accept keyboard input

- *seq* Print numeric sequences
- *set* Manipulate shell variables and functions
- *shift* Shift positional parameters
- *shopt* Shell Options
- *shutdown* Shutdown or restart linux
- *sleep* Delay for a specified time
- *sort* Sort text files
- *source* Run commands from a file `.'
- *split* Split a file into fixed-size pieces
- *ssh* secure Shell client (remote login program)
- *su* Switch to super-user (root)
- *sudo* Execute the command as super-user (root)
- *sum* Print a checksum for a file
- *symlink* Make a new name for a file
- *sync* Synchronize data on disk with memory

T

- *tac* Concatenate and write files in reverse
- *tail* Output the last part of files
- *tar* Tape ARchiver
- *tee* Redirect output to multiple files
- *test* Evaluate a conditional expression
- *time* Measure Program Resource Use
- *times* User and system times
- *touch* Change file timestamps
- *top* List processes running on the system
- *traceroute* Trace Route to Host

- *trap* Run a command when a signal is set(bourne)
- *tr* Translate, squeeze, and/or delete characters
- *true* Do nothing, successfully
- *tsort* Topological sort
- *tty* Print filename of terminal on stdin
- *type* Describe a command

U

- *ulimit* Limit user resources
- *umask* Users file creation mask
- *umount* Unmount a device
- *unalias* Remove an alias
- *uname* Print system information
- *unexpand* Convert spaces to tabs
- *uniq* Uniquify files
- *units* Convert units from one scale to another
- *unset* Remove variable or function names
- *unshar* Unpack shell archive scripts
- *until* Execute commands (until error)
- *useradd* Create new user account
- *usermod* Modify user account
- *users* List users currently logged in
- *uuencode* Encode a binary file uudecode Decode a file created by uuencode

V

- *v* Verbosely list directory contents
- *vdir* Verbosely list directory contents

- *vi* Text Editor

W

- *watch* Execute/display a program periodically
- *wc* Print byte, word, and line counts
- *wget* Retrieve web pages or files via HTTP, HTTPS or FTP
- *whereis* Report all known instances of a command
- *which* Locate a program file in the user's path.
- *while* Execute commands
- *who* Print all usernames currently logged in
- *whoami* Print the current user id and name

X

- *xargs* Execute utility, passing constructed argument list(s)

Y

- *yes* Print a string until interrupted
- *period* Run commands from a file
- *###* Comment / Remark

How to's – A Simple How to do things in Linux

How to's – How to do things in Linux

This list will tell you how to do a few everyday things in linux, just that. The internet will help you greatly if you're interested in knowing more about any of the things told here. All commands are to be typed at the terminal, without any quote. In fact these might not be the best ways of doing things but these things work and you can actually find better methods of doing it with little playing around and the beauty of this is once you do it yourself then you will never forget it. Probably thats the way people learn and master Linux.

What is remote Logging and How can I do it ??

Direct Login :

One of the many ways of logging into a computer is direct logging in, wherein you sit at the computer and type in the user name and password.

Remote Login :

Remote login is a way by which you can log into another remote computer somewhere else connected to the computer you are working by LAN/Internet(remote meaning not the computer you are currently working – it can be anywhere else !!)

To login, You must have an account in that Linux computer , naturally. The ways of logging in are telnet, ssh, VNC , Putty and many more methods.

Telnet:

Telnet gives you a command line access to a remote computer. You will be asked to enter a user name+password to login into the computer that you specify by its host name or IP address. In windows, this can be done by selecting Run at the Start button and typing "telnet <server name or IP address>" ("telnet silver", for example without the quotes). Once you enter the user name and the password in the window that appears, you are logged into the computer silver. In Linux, this can be done by typing "telnet silver" at the terminal.

SSH :

SSH is now the de facto standard of remote computer logins. SSH stands for Secure Shell. The big difference between ssh and telnet is that ssh provides enhanced security to your login session. SSH is a secure telnet which encrypts everything you type in and then unencrypts it on the server so your commands and passwords are all *safe*. From the terminal, you can login into a remote computer using ssh by typing "ssh username@hostname" and then entering the password when asked for it.

For example , if you are already logged into a Linux machine and you want to another computer whose name is "glugt" and account is "mgl" then just type

ssh mgl@glugt

VNC (Virtual Network Computing) :

VNC allows graphical desktop sharing while accessing a remote computer. To access this desktop, a vncserver should be started first. The command "vncserver" does this. The application vncviewer is a viewer for vnc. Once a vnc session is started, by typing "hostname : vncnumber" at the window that opens once vncviewer is launched, the remote desktop can be accessed. :) A password will have to be entered when you're asked for it, by the vncviewer. This password can be set by the user using the command vncpasswd.

How to copy files from a remote computer running Linux, to a computer connected to it by a network?

This is done with the command scp. Stands for secure copy. It provides the same security as ssh.

"scp user1@host1:myfile user2@host2:mynewfile"

will transfer the file myfile from the account of user1 in host1 to user2's account in host2. The transferred file in host2 will be called mynewfile.

Suppose u have to copy a file say file.txt from computer say "delta" to another server "spider" (for which you should have a login in both the computers) type

"scp user@delta:<the path to the file.txt>/file.txt user@spider:"

the colon at the end is must !!

Then it will ask for passwords for both the accounts then, after which the files are securely copied from computer to another.

How to run c, c++, java, php, perl programs from the command line in Linux?

C : "gcc -o outputfilename programmename.c" will compile prognamename.c and the object file created will be called filename. It can be run by typing "./filename".

If no " -o outputfilename " is not given then the default output file is a.out and can be run by ./a.out

C++ : Same as for C, replacing gcc with g++.

Java : "javac filename.java" will compile filename.java. An object file called filename (without any extension) will be created and it can be run by typing "java filename".

PHP : "php -f filename.php" will run filename.php.

Python : “python filename.py”

How to mount a pen drive if it's not getting mounted by itself?

By default your pen drive or cd or any external disc when connected to the computer they are mounted in /media folder. So you can check there. But if its not getting mounted there then go to /dev and there you can see all the hardware in the system as files. Your hard discs will be like sda1,sda2 or hda1,hda2 for IDE drives. So now Your pen drive will be like sdb1 or hdb1 like that. Now there is a simple command to mount that manually by this command :

“mount -t vfat /dev/sdb1 /mnt”

Now what this does is mount the sdb1 partition pen drive in /mnt of file format type fat32 or fat16 mentioned by -t. you can unmount the pen drive like this :

“umount /mnt”

(ie) mentioning only the place where the pen drive was mounted !!

you can also mount any hard disc partitions like putting the harware to be mounted properly. Suppose you have an iso image and then you want to check whats inside it then you can mount also mount an iso file just by changing the file format type like this “mount -o loop -t iso9660 <filename.iso> /mnt”

How to show off with the command line, to people who don't use Linux?

Best learnt by oneself. Practice helps a lot, as it always has a way of doing. But try to hide what you are doing from others or do things very fast so that people don't understand what you are doing !!

How to find the ip address, knowing the hostname of a server or vice versa?

“ping hostname” or “ping ipaddress” will send ICMP packets to a host computer and the host computer would reply by echoing those packets. It's generally used to test whether a host is active or not.

Mounting windows drives:

playing mp3 files and videos

By default Linux distributions cannot ship with mp3 support because it has legal problems but you can always download and install mp3 support. Mostly you will find all the support for mp3, mp4, ogg,mpeg, avi, wmv playable by players like amarok, xine,mplayer and vlc from repositeries.

How to find which services are run by a host computer?

A host computer would be running several services (like telnet, ssh, ftp, etc.). “nmap www.nitt.edu” would get you a list of services run by the server, unless it doesn't let this information out.

Connecting to net(broad band):

Just connect the computer with the normal Internet connection given by your ISP and then select "Administration>Network" .In that select the active network device (mostly eth0) and the double click it. Now in that select "Obtain ip address automatically from dhcp" and then close that dialog box. Then in the dns tab add the dns addresses provided by your ISP. Eureka !! just open Firefox and start surfing !!

Auto-completion -

In the unix shell command prompt, there is an option to perform auto-completion of filenames and paths. This task is accomplished with the help of the "tab" key. In the shell prompt, the first few letters of the filename is entered followed by the tab key which auto-completes the filename. If more than one filename exists beginning with the same combination of letters, then all the possible filenames are displayed from where further letters are typed. This feature is particularly useful when browsing through a huge database of files and typing down in shell becomes very fast .

Configuring Gtalk in GAIM

1. Open Gaim.
2. From The buddy list screen, Click Tools.
3. Then Click Accounts.
4. Click the Add Button.

5. Select Jabber as the protocol.
6. Your Screen Name is your Gmail login.
7. Server is gmail.com.
8. Resource is GAIM.
9. Your password is gmail password.
10. Click “Show More Options”.
11. Select the “use TLS if available” checkbox.
12. Enter server type as talk.google.com.
13. Click Save.
