# Introduction to GNU/Linux

Rupam Das

Email: info@rupamdas.com info@nxnvision.com

Website: www.rupamdas.com www.nxnvision.com

Contact: +91 98407 84107

Seek.Learn.
Achieve.Grow.

## Objectives

- Linux Introduction
- Linux anatomy, processes, file system, file structure
- Boot process, shutdown and reboot
- Login, logout process
- Standard input, standard output, standard error, redirection, pipes, filters
- Managing passwords
- Getting help – man command
- Files and related commands

## Prerequisites for this session

- Basic understanding of Operating Systems
- Familiarity with any desktop computer

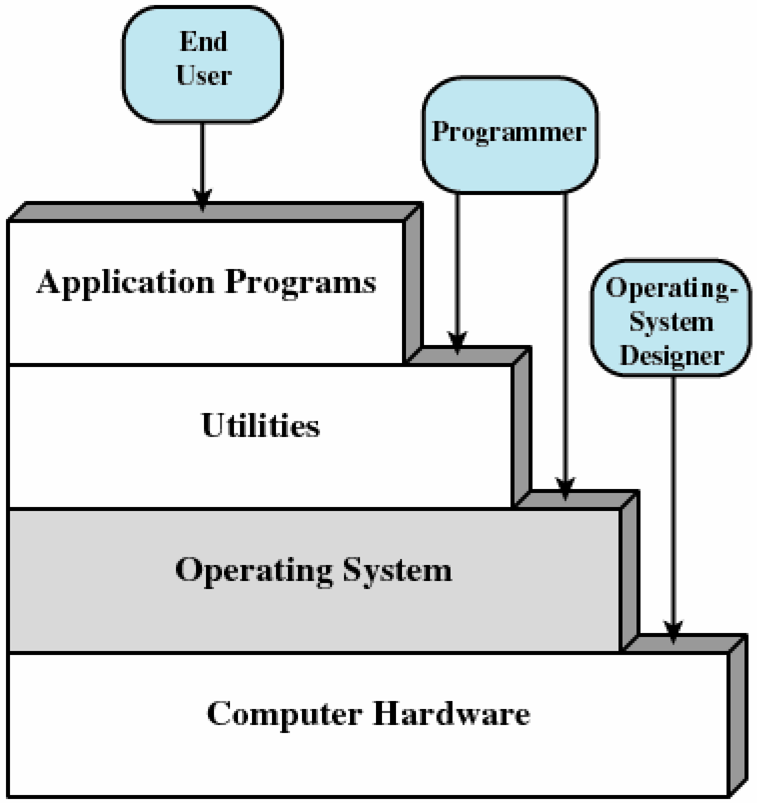
## Operating system

- What is an Operating system?
  - A system program that controls the execution of application programs
  - An interface between applications and hardware
  - A program that manages the system memory and other resources of the system

## Operating System Objectives

- Convenience
  - Makes the computer more convenient to use
- Efficiency
  - Allows computer system resources to be used in an efficient manner
- Ability to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with service

## Layers of Computer System



Layers and Views of a Computer System

## Services Provided by Operating System

- Program development
  - Editors, Compilers, Linkers and debuggers
- Program execution
  - User applications and Services
- Access to I/O devices
  - Hard disk drive, DVD drive, Ethernet, USB, Bluetooth
- Controlled access to files
  - File system management
- System access
  - Control Panel

## Services Provided by Operating System

- Error detection and response
  - Internal and external hardware errors
    - Memory error
    - Device failure
  - Software errors
    - Arithmetic overflow
    - Access forbidden memory locations
- Accounting
  - Collect usage statistics
  - Monitor performance
  - Used to anticipate future enhancements
  - Used for billing purposes

## What is GNU/Linux?

- GNU/Linux is a free, open source, UNIX-like operating system (OS) that runs on diverse computing hardware platforms
- GNU provides the shell, library, compilers
- Linux provides the kernel
- First released in 1991, based on Linux *kernel* developed by Linus Torvalds and the *programs* developed by Richard Stallman
- *This combination is called GNU/Linux OS* or mostly just referred to Linux OS
- For all practical purposes, Linux refers to GNU/Linux operating system
- For more information on Linux and GNU/FSF visit sites, http://www.kernel.org, http://www.gnu.org & http://www.fsf.org

## Brief History of Linux

- First version of Linux was released by Linus Torvalds in 1991
- Further release history in brief
  - Version 1.0.0 of March 14 1994 supported only single-processor i386 machines.
  - Version 1.2.0 of March 1995 added support for Alpha, Sparc and MIPS.
  - Version 2 of June 9 1996 included SMP support and added support for more types of processors.
  - Version 2.2.0 of January 25 1999
  - Version 2.4.0 - January 4 2001 – added support for USB, ISA Plug-and-Play
  - Version 2.6 -  December 17, 2003

## Modern Linux Systems

- Several distributions are available that use Linux kernel and GNU software packages
- Popular Linux distributions are
  - RedHat Enterprise Linux (not free)
  - Fedora
  - Ubuntu
  - Debian
  - Suse
  - Mandriva
  - and several others available. Check http://en.wikipedia.org/wiki/List_of_Linux_distributions

## Linux as a Development Platform

- Ample support for software development, whether kernel or applications
- Development is possible in C, C++, Shell Scripts, Java, PHP, Python, Perl, HTML, CGI
- For applications development several IDEs are available:
  - Eclipse for C, C++ and Java
  - Source Navigator, Glade, KDevelop, KDE Studio for C/C++
  - NetBeans, Sun One Studio, WebSphere for Java
  - Komodo and PerlComposer for Perl
  - Komodo, Zend for PHP
  - Cooledit, ActivePython, wxPython for Python

## Linux as a Server

- As a server, Linux has achieved the most recognition
- The stable and highly evolved networking features and options make Linux based servers a good choice
- Linux based servers are available for
  - Web server – Apache Web server
  - File server - Samba
  - Mail server - sendmail
  - DHCP server
  - Printer server - CUPS

## Linux Kernel development model

- Linux development is primarily driven by Linux users and developers, spread all over the world
- The developers (or kernel hackers) test Linux releases and locate software bugs and provides fixes OR add more functionality w.r.t. the kernel and the hardware supported by the kernel. All these are reported to the Linux Kernel maintainers
- All modifications are then verified, integrated and then released in the following version.
- The releases are maintained at http://www.kernel.org. Each kernel version tree has at least one maintainer
- Currently the active streams are 2.2.x, 2.4.x and 2.6.x. Currently more focused work is going on the 2.6.x stream

## Brief History of GNU and FSF

- GNU is a project started in 1984 to develop a complete OS which would be like Unix but ‘free’ (http://www.gnu.org)
- GNU is read as “GNU is Not Unix”, a recursive acronym
- Started by Richard Stallman (http://www.stallman.org/) who also started the Free Software Foundation (http://www.gnu.org/philosophy/free-sw.html)
- When GNU programs were finished, the GNU kernel was not.
- So, GNU used the Linux kernel and the GNU/Linux OS was born in 1991.
- Free Software Foundation (FSF) is the principal sponsor for GNU projects (http://www.fsf.org)

## Significance of GNU/Linux today

- GNU/Linux offers a cheap and stable alternative OS
- Large number of developers spread over worldwide who contribute to its development and sustainment
- Cheaper or even free alternatives for existing paid/proprietary software
- It runs on even very low end computing hardware

## Applications of Linux

- Linux
  - As an alternative Desktop
  - As a Development platform
  - As a Server
  - In Embedded Systems
- Advantages and Disadvantages of Linux
- Popular Windows applications and their Linux equivalents
- Popular Linux distributions

## Linux as an alternative Desktop

- Linux has all the features of a normal Desktop
  - Stable and evolving GNU/Linux OS
  - Built in desktop features like file browser/manager, Internet browser, system settings, taskbar, desktop screen and shortcuts to applications
  - GUI, with options to choose from – Gnome, KDE
  - Command line interface for the advanced user
  - Capability to install 3rd party applications like FireFox, Thunderbird, OpenOffice
  - Built-in (or downloadable) drivers for various hardware peripherals like CD/DVD drive, printers, monitors, mouse, keyboards
  - Various softwares available for documentation, audio, video, Internet browsing, and much more

# Linux as a Development Platform

- Ample support for software development, whether kernel or applications
- For kernel, source code of kernel, C libraries, compilers, linkers are available for free download
- Development is possible in C, C++, Java, PHP, Python, Perl, HTML, CGI
- For applications development several IDEs are available:
  - Eclipse for C, C++ and Java
  - Source Navigator, Glade, KDevelop for C/C++
  - NetBeans, Sun One Studio, WebSphere for Java
  - Komodo and PerlComposer for Perl
  - Cooledit, ActivePython, wxPython for Python

## Linux as a Development Platform (2)

- For source code control and management (SCM) several softwares are available
  - CVS, Subversion, Source Integrity
- Simulators
  - NS2 for Networking
  - SkyEye for hardware and OS/RTOS simulator:
    - CPU cores: ARM7TDMI, ARM720T, StrongARM, XScale
    - OS/RTOS: uClinux, ARM Linux, Nucleus, eCos, uC/OSII
- Emulators
  - CPU emulator: QEMU
  - Even Windows emulation is possible using WINE (WINdows Emulator) software on Linux!

## Linux as a Server

- As a server, Linux has achieved the most recognition
- The stable and highly evolved networking features and options make Linux based servers a good choice
- Linux based servers are available for
  - Web server – Apache Web server
  - File server - Samba
  - Mail server - sendmail
  - DNS server
  - DHCP server
  - Printer server

## Linux in Embedded Applications

- Linux has been customized for numerous embedded applications:
  - Networking – Routers, Switches, Gateways, Thin Clients
  - Audio/Video – Set top box, DVR, Portable Music Players
  - Communications – Mobile phones, PDAs, IP Telephones
  - Robotics – Entertainment and Utility robots
- For a glimpse of Linux in embedded devices, visit
  - http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/The-Linux-Devices-Showcase

## Popular Windows software and Linux equivalents

- This list is an incomplete comparison between MS Windows XP and Fedora distribution
  - File Browser – Windows Explorer, Nautilus
  - Internet Browser – Internet Explorer, Konqueror/Mozilla Firefox
  - Documentation (Documents, Presentations, Spreadsheets)– MS Office, OpenOffice
  - PDF Reader – Adobe, Evince/KPDF
  - Simple Graphics – MS Paint, KolourPaint
  - Advanced Graphics – Adobe Photoshop, GIMP
  - Text editor – Notepad/Wordpad, Gedit
  - Image Viewer – Windows Picture and Fax Viewer, gThumb
  - Music player – Windows Media player, Audacious/Rhythmbox
  - Video player - Windows Media player, Totem Movie Player
  - Email client – MS Outlook, Evolution/Mozilla Thunderbird

## Popular Linux Distributions

- Linux Distribution refers to a package of a particular stable Linux Kernel version along with host of other programs and applications that are combined together
- There are several distributions (or versions) available
- Few of the well known desktop distributions are:
  - RedHat Enterprise Linux (not free)
  - Fedora
  - Ubuntu
  - Debian
  - Mandriva
  - Suse

## Advantages of Linux

- Low Cost
- Open Source software packages
- Stable
- Networking support
- Compatibility
- Configurability
- Multi-tasking
- Multi-user
- Secure
- Evolving constantly without adding cost to the end user

## Challenges in Linux

- Learning curve is higher for people exposed only to MS Windows
- It was initially designed by programmers for programmers, so for normal users its difficult to grasp at the beginning
- Not all MS Windows application equivalents present in Linux
- Administration of Linux systems is tough for beginners
- Not all hardware are compatible, especially peripherals, whose drivers may not exist

## Linux Architecture and Features

- Linux architecture
- Linux Operating System features
- Linux File System
- Linux Kernel Architecture and Features
- Kernel and User interface
- Linux GUI
- Linux Console and basic commands

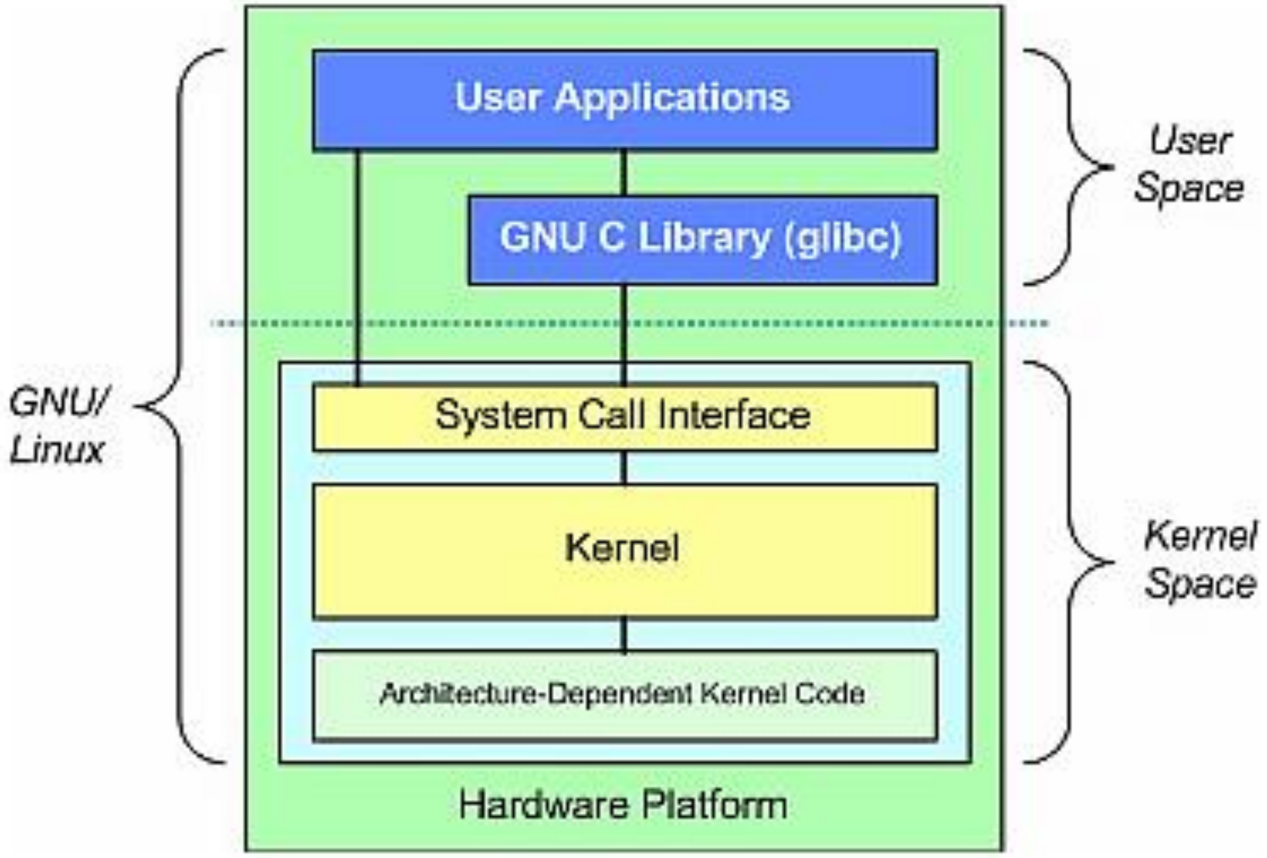
## Linux OS features

- Some of the main features of Linux OS are:
  - File system
  - Multitasking
  - Multiuser
  - Multiple platform support
  - Memory protection between processes
  - Virtual Memory using paging to disk
  - Networking support

## Linux Architecture

- Hardware is surrounded by the operating system software
- Operating system is called the system kernel
  - Implemented mostly in C and Assembly
  - Interacts with applications and Shells
- Comes with a number of user services and interfaces
  - Shell
  - Components of the C compiler

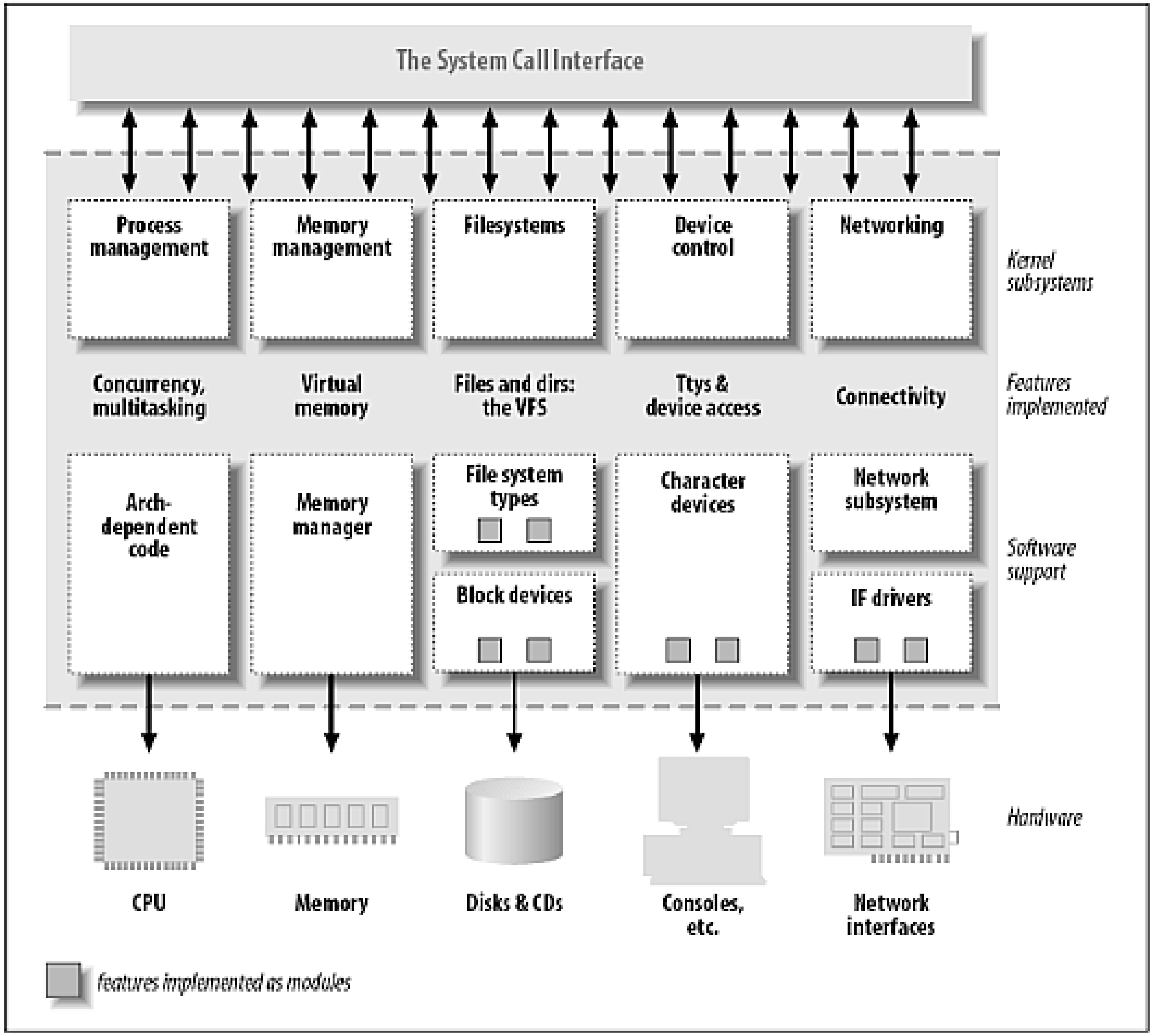
## Basic Architecture of Linux



## Linux Architecture (2)

- Linux is primarily divided into two parts, Kernel space and User space
- User Space
  - User space contains all the user applications
  - It contains the OS services like command shells and window system
  - It also contains libraries and compilers, the programming interface to the Kernel
  - Those applications that need to access the kernel, use the System call feature provided by the Kernel
- Kernel Space
  - Kernel contains entire code necessary to manage the processor, peripherals and memory resources
  - Part of Kernel code is generic and remaining code depends on the architecture of the processor on which it has to run
  - Kernel also provides TCP/IP networking support

# Linux Kernel Architecture



## Linux Kernel and Responsibilities

- Responsible for
  - Scheduling running of user and other processes.
  - Allocation of memory.
  - Managing the swapping between memory and disk.
  - For moving data to and from the peripherals.
  - Receives service requests from the processes and honours them

## Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources
- Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

## User Mode and Kernel modes of process

- At any time only one process can engage a given CPU
- This process may be in user mode or kernel mode
- User applications run in user mode normally
- Kernel code runs in kernel or privileged mode
- If a user mode application needs kernel privileges, then the application switches to privileged mode to access privileged resources
  - The above method allows the kernel to keep a watch over the user mode application's actions
  - The switching to kernel mode takes place with "System Call"

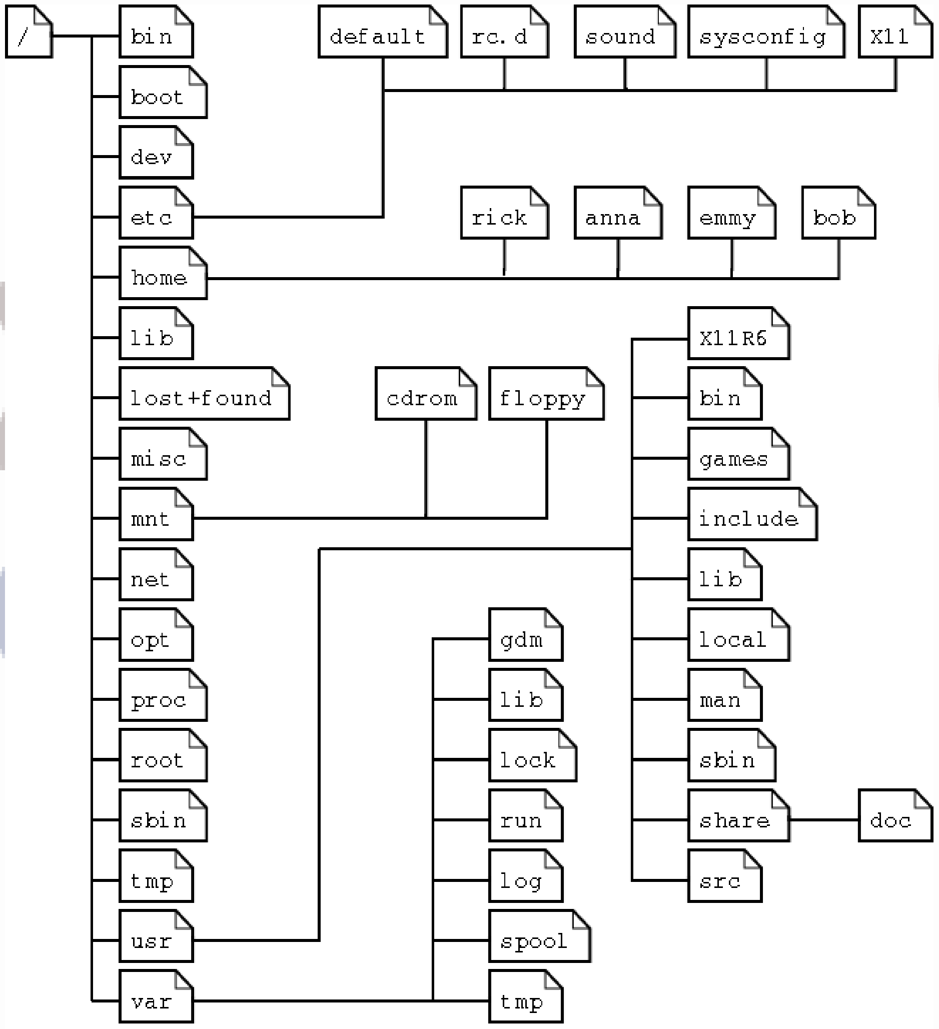
## Linux File System

- **In Linux everything is a file!**
- Everything can be accessed and handled like a way a normal file is handled
  - Open, Close, Read, Write, Create, Remove
- This means regular files, directories, links, devices and peripherals, pipes and sockets
- The files names are Case Sensitive! Can contain any character except ‘/’
- File name length is not restricted
- Examples
  - README, temp.txt, .bashrc, New File, simple.doc.backup

## File Paths

- A file path in Linux is a sequence of nested directories
  - Last element can be a file or directory
- Path can be relative or absolute
  - Relative path – Documents/Design/Software/DDD-XYZ.pdf
  - Absolute path - /home/jack/work/software/proj1/source1.c
- The ‘/’ represents the ‘root’ of the file system
  - All absolute file paths start from root directory
- The Linux filesystem structure is defined by the Filesystem Hierarchy Standard (FHS): http://www.pathname.com/fhs/

# Linux File System



## Directory Description – Important ones

| | |
|---|---|
| bin | Essential command binaries |
| boot | Static files of the boot loader |
| dev | Device files |
| etc | Host-specific system config |
| home | User login and data folders |
| lib | Essential shared libraries and kernel modules |
| mnt | Mount point for mounting a filesystem or media |
| opt | Add-on application software packages |
| proc | Virtual folder that has information about system |
| root | Home folder of the Linux superuser |
| sbin | Essential system binaries |
| tmp | Temporary files |
| usr | Secondary hierarchy |
| var | Variable data |

- /bin:
  - The bin directory contains several useful commands that are of use to both the system administrator as well as non−privileged users. It usually contains the shells like bash, csh, etc.
  - Example commands :cp, mv, rm, cat, ls
- /boot:
  - This directory contains everything required for the boot process except for configuration files not needed at boot time
  - Example : /boot/vmlinuz, /boot/vmlinuz−kernel−version
- /dev
  - /dev is the location of special or device files. Look through this directory and you should hopefully see hda1, hda2 etc. which represent the various partitions on the first master drive of the system.
  - /dev/cdrom and /dev/fd0 represent your CD−ROM drive and your floppy drive.

- /etc:
  - This is the nerve center of your system, it contains all system related configuration files in here or in its sub directories
  - Example: /etc/fstab, /etc/passwd, /etc/shadow

- /home:
  - /home can get quite large and can be used for storing downloads, compiling, installing and running programs, your mail, your collection of image or sound files etc. Each user is also assigned a specific directory that is accessible only to them and the system administrator, typically in /home path

- /lib:
  - The /lib directory contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root filesystem, ie. by binaries in /bin and /sbin.

- /root:
  - This is the home directory of the System Administrator.
- /sbin:
  - /sbin should contain only binaries essential for booting, restoring, recovering, and/or repairing the system in addition to the binaries in /bin.
- /usr:
  - /usr usually contains by far the largest share of data on a system. Hence, this is one of the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc.
- /var:
  - Contains variable data like system logging files, mail and printer spool directories, and transient and temporary files.
  - Example:/var/cache: Is intended for cached data from applications
  - /var/log: Log files from the system and various programs and services

- /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.
- /proc:
  - /proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo−file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).

## Linux Booting

- For desktop PCs the typical boot sequence is:
  - BIOS – After system power-on, to identify, test, and initialize system devices such as the video display card, hard disk, floppy disk and other hardware. BIOS loads the Bootloader, mostly GRUB (earlier LILO)
  - GRUB loads the Linux Operating System (Kernel and File System) into the memory. It checks the system hardware and peripherals and executes the Kernel.
  - Kernel execution first calls the Init program which is the 1st user space program
  - Init runs /etc/inittab which then runs the necessary start-up scripts
  - After all the start-up scripts are executed, the system comes to multi-user mode

## Shutdown and reboot in Linux

- Click on System icon in the task bar, then click on Shut Down
- When system asks "Shut down this system now", the different options provided can be selected
  - Cancel to avoid shutdown
  - Restart to reboot the system
  - Shut down to shut the system

## Login in Linux

- Power-on the Linux system and wait till it displays the login screen
- The login is an important security feature of Linux
  - Without logging in, little can be done
- So, first thing one needs to work in Linux is a valid Username and Password set
  - Username – An alphanumeric ID used to identify the user who is logged in. Only the Linux Administrator can add users to any Linux system
  - Password – An alphanumeric password string used to authenticate the user logging in. Its stored in encrypted form in the system

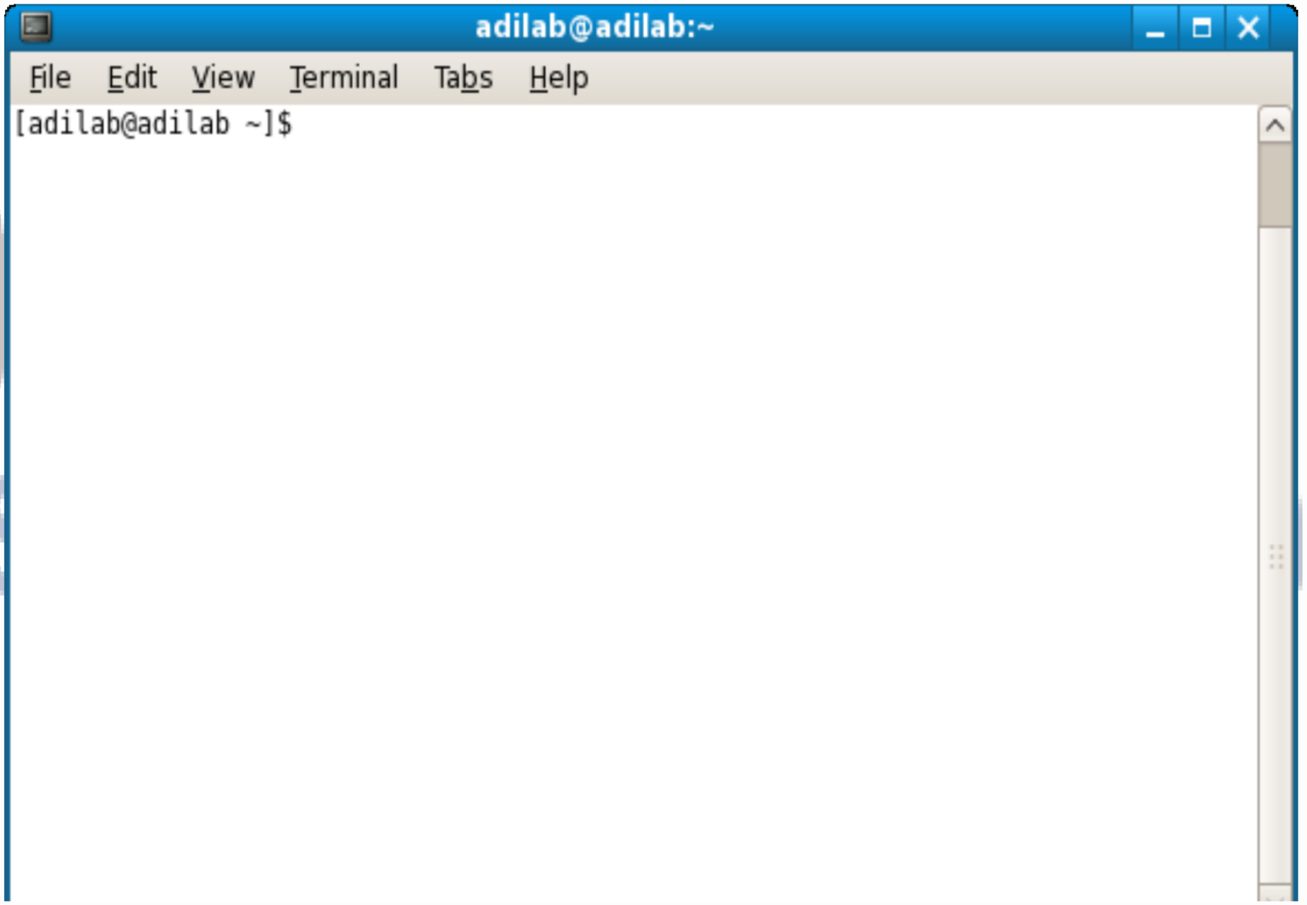
## Login in Linux (2)

- One Username and password set plus the root password is created at the time of Linux installation
  - If you do not know the Username and Password combination, ask your System Administrator

## Logout in Linux

- When in console, logging out is done using the command “exit”
- In GUI, System-> Log Out <Username> will bring up the dialog, “Log out of this system now?”
  - User can then chose to log out or cancel

## Linux console



```
[adilab@adilab ~]$
```

## Change of password

- Passwords can be changed later through GUI or console
- From console the command is “passwd”
  - Given the command, first the old password will be asked
  - If the old password is correct, then it prompts for the new password, twice
  - If new password is typed correctly both times, the password is modified
- In GUI, the following sequence has to followed
  - System-> Administration->User and Groups
  - Then select the user whose password is to be modified and click on Properties
  - Under the Account tab, the password can be changed

## More on passwords

- Password tips
  - NEVER tell anyone else your official password
  - Don’t write it down
  - A good password typically is:
    - 8 (or more) characters long
    - uses a mix of uppercase and lowercase letters, numbers, and symbols. For example, yGnt4^%a12

## Users in Linux

- Linux has some default users preconfigured
  - root, daemon, nobody, bin, uucp
  - These are to be used by the system for allowing access to various utilities and system programs
  - User "root" is the Administrator, or better known as the "super user"
    - "root" has ALL the privileges to add/delete/modify anything within the system
    - Unless its absolutely necessary, logging into "root" is not advised
- The list of all users is available in "/etc/passwd"
- The passwords of these users are available in "/etc/shadow" file (in encrypted manner)
- All users are members of some group and the list of all groups is in "/etc/group" file

## Users and Groups

- User accounts are used within computer environments to verify the identity of the person using a computer system.
- Groups are logical constructs that can be used to cluster user accounts together for a  specific purpose.
- For instance, if a company has a group of system administrators, they can all be placed in a system administrator group with permission to access key resources and machines.

## User and Group Permissions

- There are three permissions for files, directories, and applications.
  - r — Indicates that a given category of user can read a file.
  - w — Indicates that a given category of user can write to a file.
  - x — Indicates that a given category of user can execute the file.
  - A fourth symbol (-) indicates that no access is permitted
- Each of the three permissions are assigned to three defined categories of users. The categories are:
  - owner — The owner of the file or application.
  - group — The group that owns the file or application.
  - everyone — All users with access to the system.
- Reading file permissions : `ls -l`

```
> ls -l
myfile -rwxr-x--- 1 george administrators 10 2006-03-09
  21:31 myfile
```

## User and Group Permissions (2)

- The first character simply indicates the type of file as indicated in the table below:

| ***Character*** | ***Type of file*** |
|---|---|
| d | directory |
| - | regular file |
| l | symbolic link |
| s | socket |
| p | named pipe |
| c | character device file |
| b | block device file |

| ***Letter*** | ***Permission*** |
|---|---|
| r | Read |
| w | Write |
| x | Execute |
| - | No permission |

## User and Group Permissions (3)

| Letter | Type of users |
|---|---|
| u | User (owner of the file) |
| g | Group (group to which belong the file) |
| o | Other (users who are neither a member of the Group nor the owner of the file) |
| a | All (everybody) |

- So, in our example myfile features the following set of permissions :

  **`rwxr-x---.`**

  - This means that George has all three rights on it, that members of the Administrators group can only read (R) and execute (X) the file, and that everybody else can't do anything with the file.

## User and Group Permissions (4)

- chmod command

  `chmod o+r myfile`
  - adds read permission to the others on myfile

  `chmod ug+rx myfile`
  - adds read and execute permissions to both the owner (user) and the group on myfile

  `chmod a-rwx myfile`
  - removes all permissions to everybody (all) on myfile

  `chmod 755 myfile`
  - **`rwxr-xr-x`**, all rights to the owner, other people only read and execute

# User and Group Permissions (5)

- Command line tools to manage users and groups:
  - useradd, usermod, and userdel — methods of adding, deleting and modifying user accounts
  - groupadd, groupmod, and groupdel — methods of adding, deleting, and modifying user groups
  - gpasswd — method of administering the /etc/group file
  - Some Examples:

```
> useradd jack
> groupadd trainee
> gpasswd -a jack trainee
> userdel jill
> groupdel trainee
```

- Chown Command (Change Ownership)

```
> chown newowner file
```

- Chgrp Command (Change Group)

```
> chgrp newgroup file
```

## Getting help in Linux – Man pages

- Man pages
  - “man <command>” will show the help on the “<command>”
  - For example, “man man” will show the information about man command itself
  - Man pages are documentation of the Linux commands and utilities
  - To quit a man page press ‘q’/’Q’
- Shell help
  - In case you know the command name, try “<command> --help”
  - In most cases, the basic command line options and explanation would be displayed
  - Example,

```
mv --help
```

## Getting help in Linux - Apropos

- Apropos
  - “`apropos <topic>`” will display a list of commands in Linux that have “topic” included in its documentation
  - Good way to start searching on a topic, if the exact command is not known
  - By default the apropos database may not be present. To generate the database use the following command (while logged in as “root”)
    - `/usr/bin/catman -w`

## Basic console commands – ls

- “ls” command displays the list of files present in a given directory except those starting with a ‘.’
- Other options of “ls”
  - `ls -a`
    - Lists all the files (including .* files)
  - `ls -l`
    - Long listing (type, date, size, owner, permissions)
  - `ls -t`
    - Lists the most recent files first
  - `ls -S`
    - Lists the biggest files first
  - `ls -r`
    - Reverses the sort order
  - `ls -h`
    - Enables display of file sizes in K, M and G
  - `ls -ltr`
    - Long listing, most recent files at the end

## Basic console commands – ls (2)

- Using wild card character '*'
- `ls - *.txt`
  - Displays list of all files that end in "txt", except those starting with '.'
- `ls -d .*`
  - Lists all the files and directories starting with '.'
  - '-d' tells ls not to display the contents of directories
- `ls temp.*`
  - Lists all files starting with name temp.
  - Like temp.doc, temp.txt, temp.xls

## Special diretories

- If we give the command “ls –a”, we see two directories, ‘.’ and ‘..’
- The ‘.’ represents the current directory
  - Used as ‘./’.
  - Files ./temp.txt and temp.txt are the same in a given directory
- The ‘..’ represents the parent directory
  - Used as ‘../’, typically for moving up by one level with ‘cd’ command
- The ‘~’ represents the home directory of users
  - If there is a user “user1” and user’s home path is “/home/user1”, this path can be referenced as “~user1”

## Basic console commands – cd

- “cd” or change directory command allows the user to traverse through the directories and file paths
  - “cd <dir>” changes the current directory to the directory <dir>, provided the directory exists and the user has access privileges to the directory
  - <dir> can be relative or absolute
    - `cd /home/user1/Desktop/Folder1`
    - `cd Documents/group-docs/engg`
  - “cd – “ gets back to the previous current directory.
  - “pwd” displays the current directory ("working directory").

## Basic console commands – cp

- Copy files/directories
  - "cp" command is used to copy file(s) from a source to a destination
  - `cp <source_file> <target_file>`
    - Copies the source file to the target.
  - `cp file1 file2 file3 ... dir`
    - Copies the files to the target directory (last argument).
  - `cp -i`
    - Asks for user confirmation if the target file already exists
  - `cp -p`
    - Preserves the attributes of the source file(s) such as timestamp, mode, ownership
  - `cp -r <source_dir> <target_dir>`
    - Copies the whole directory, recursively

## Basic console commands – mv and rm

- Move files
  - `mv <old_name> <new_name>`
    - Renames the given file or directory.
  - `mv -i`
    - If the new file already exits, asks for user confirm
  - `mv -p`
    - Preserves the attributes of the source file(s) such as timestamp, mode, ownership
  - `mv -f`
    - Force the move, overwriting the destination file
- Remove files
  - `rm file1 file2 file3`
    - Removes the given files.
  - `rm -i`
    - Always ask for user confirm.
  - `rm -r dir1 dir2 dir3`
    - Removes the given directories with all their contents
  - `rm -f`
    - Force the remove, no questions asked

## Creating and removing directories

- Create directories
  - `mkdir dir1 dir2 dir3`
    - Creates directories with the given names.
- Remove directories
  - `rmdir dir1 dir2 dir3`
    - Removes the given directories
  - Safe: only works when directories and empty.
  - Alternative: rm -r, (doesn't need empty directories).
  - `rm -rf <dir>`
    - Deletes files/directories without asking.
    - SHOULD BE USED WITH EXTREME CARE!

## File contents display

- Several ways of displaying the contents of files:

```
cat file1 file2 file3
```

  - Concatenates and outputs the contents of the given files

```
more file1 file2 file3
```

  - After each page, asks the user to hit a key to continue
  - Can also jump to the first occurrence of a keyword (/ command)

```
less file1 file2 file3
```

  - Does more than more with less
  - Doesn't read the whole file before starting
  - Supports backward movement in the file (w)

## File contents display - partial

- `head [<n>] <file>`
  - Displays the first <n> lines (or 10 by default) of the given file.
  - Doesn't have to open the whole file to do this!
- `tail [<n>] <file>`
  - Displays the last <n> lines (or 10 by default) of the given file.
  - No need to load the whole file in RAM! Very useful for huge files.
- `tail -f <file>`
  - Displays the last 10 lines of the given file and continues to display new lines when they are appended to the file.
  - Very useful to follow the changes in a log file

## Search – grep command

- Format: `grep <pattern> <files>`
- Scans the given files and displays the lines which match the given pattern.

- `grep error *.log`
  - Displays all the lines containing error in the *.log files
- `grep -i error *.log`
  - Same, but case insensitive
- `grep -ri error .`
  - Same, but recursively in all the files in . and its subdirectories
- `grep -v info *.log`
  - Outputs all the lines in the files except those containing info.

## Ordering – sort command

- Format: `sort <file>`
  - Sorts the lines in the given file in character order and outputs them.
- `sort -r <file>`
  - Same, but in reverse order
- `sort -ru <file>`
  - u: unique. Same, but just outputs identical lines once

## Linux - Standard I/O

- Linux provides standard facilities for using Input and Output on Linux terminals
- There is no difference between reading the input from command-line or a file, or for that matter writing output
- The Linux standard I/O has three components
  - Standard Output - stdout
  - Standard Input - stdin
  - Standard Error - stderr

## Standard Output

- All the commands that give text output on the terminal, do it by writing it to the *standard output*.
  - Standard output can be written (redirected) to a file using the > symbol
    - Creates a new file

```
ls -la ./ > folder_list.txt
```

  - Standard output can be appended to an existing file using the >> symbol

```
cat file1 > bigfile
cat file2 >> bigfile
cat file3 >> bigfile
```

    - The above will generate a file, bigfile, in which all three files will present in the order, file1:file2:file3

## Standard Input

- Some commands, when not given input arguments, can take their input from *standard input*.
- Try this:

```
sort
windows
linux
[Ctrl][D]
linux
windows
```

  - sort takes its input from the standard input
  - in this case, what you type in the terminal and ended by [Ctrl][d]
    - `sort < namelist.txt`
  - The standard input of sort is taken from the given file.

## Standard Error

- Error messages are usually output (if the program is well written) to *standard error* instead of standard output
- Standard error can be redirected through 2> or 2>>
- Example:

```
cat file1 file2 nofile > newfile 2> errfile
```

  - Note: 1 is the descriptor for standard output, so 1> is equivalent to >
- Can redirect both standard output and standard error to the same file using '&>'

```
cat f1 f2 nofile &> wholefile
```

## Pipes

- Linux pipes are very useful to redirect the standard output of a command to the standard input of another one.
- Examples

```
cat *.log | grep -i error | sort

grep -ri error . | grep -v “ignored” \
| sort u > serious_errors.log

cat /home/*/homework.txt | grep mark \
| more
```

- This one of the most powerful features in Linux shells!

## More Linux Shell commands (1)

- echo
- gzip
- ln/link
- find
- cut
- pg
- tr
- diff
- cmp

## More Linux Shell commands (2)

- who
- which
- whoami
- date
- time
- script
- bc

## Command - echo

- Displays the given text on the screen
- Format: echo [OPTION]... [STRING]...

| | |
|---|---|
| -n | do not output the trailing newline |
| -e | enable interpretation of backslash escapes |
| -E | disable interpretation of backslash escapes (default) |

- If -e is in effect, the following sequences are recognized:

| | |
|---|---|
| \\ | backslash |
| \b | backspace |
| \c | suppress trailing newline |
| \n | new line |
| \r | carriage return |
| \t | horizontal tab |

## Environment Variables

- Environment variables are global settings that control the function of the shell and other Linux programs
- They are sometimes referred to global shell variables.
- Examples of setting:

```
VAR=/home/fred/doc
export TERM=ansi
SYSTEMNAME=`uname -n`
```

- Using Environment Variables:

```
echo $VAR
cd $VAR
cd $HOME
echo "You are running on $SYSTEMNAME"
```

- Displaying - use the following commands:

```
set (displays local & env. Vars)
export
```

## Some Important Environment Variables

- HOME
  - Your home directory (often be abbreviated as “~”)
- TERM
  - The type of terminal you are running (for example vt100, xterm, and ansi)
- PWD
  - Current working directory
- PATH
  - List of directories to search for commands

## PATH Environment Variable

- Controls where commands are found
  - PATH is a list of directory pathnames separated by colons. For example: `PATH=/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/home/scully/bin`
  - If a command does not contain a slash, the shell tries finding the command in each directory in PATH. The first match is the command that will run
- Usually set in `/etc/profile` as global to all
- Local settings modified in `~/.profile`

## Command – gzip

- Compression/decompression utility for files
- gzip compresses files and gunzip uncompresses files
- Each file is compressed individually
- Format: gzip [OPTIONS] [-S suffix] [file …]

|  |  |
|---|---|
| -d --decompress | decompress |
| -f --force | force overwrite of output file and compress links |
| -l --list | list compressed file contents |
| -q --quiet | suppress all warnings |
| -r --recursive | operate recursively on directories |
| -v --verbose | verbose mode |
| file... | files to (de)compress. If none given, use standard input. |

## Command - tar

- Compression/decompression utility for files
- gzip compresses files and gunzip uncompresses files
- All files go into a single archive while compressing
- Format: tar [OPTION...] target [FILE]...

|  |  |
|---|---|
| -c, --create | create a new archive |
| -x, --extract, --get | extract files from an archive |
| -p, --preserve-permissions, --same-permissions | extract information about file permissions (default for superuser) |
| -f, --file=ARCHIVE | use archive file or device ARCHIVE |
| -j, --bzip2 | filter the archive through bzip2 |
| -z, --gzip, --gunzip, --ungzip | filter the archive through gzip |
| -Z, --compress, --uncompress | filter the archive through compress |
| -v, --verbose | verbosely list files processed |
| -t, --list | list the contents of an archive |
| -u, --update | only append files newer than copy in archive |

## tar

- Examples

```
tar -cf file.tar f1 f2      # Create file.tar from files foo and bar
tar -tvf file.tar           # List all files in archive.tar verbosely
tar -xf file.tar            # Extract all files from archive.tar
tar -czf file.tar f1 f2     # Create file.tar from files foo and bar using gzip
tar -cjf file.tar f1 f2     # Create file.tar from files foo and bar using   bzip
tar -cvf file.tar f1 f2     # Create file.tar from files foo and bar, verbose
```

## Command - ln

- ln creates symbolic reference to existing file
- Softlink is just a reference to an existing file
- Hardlink is a physical copy of an existing file
- Removing a link does not affect the original file

- To create softlink

```
ln -s <target_file> <link_name>
ln -s <target_dir> <link_name>
```

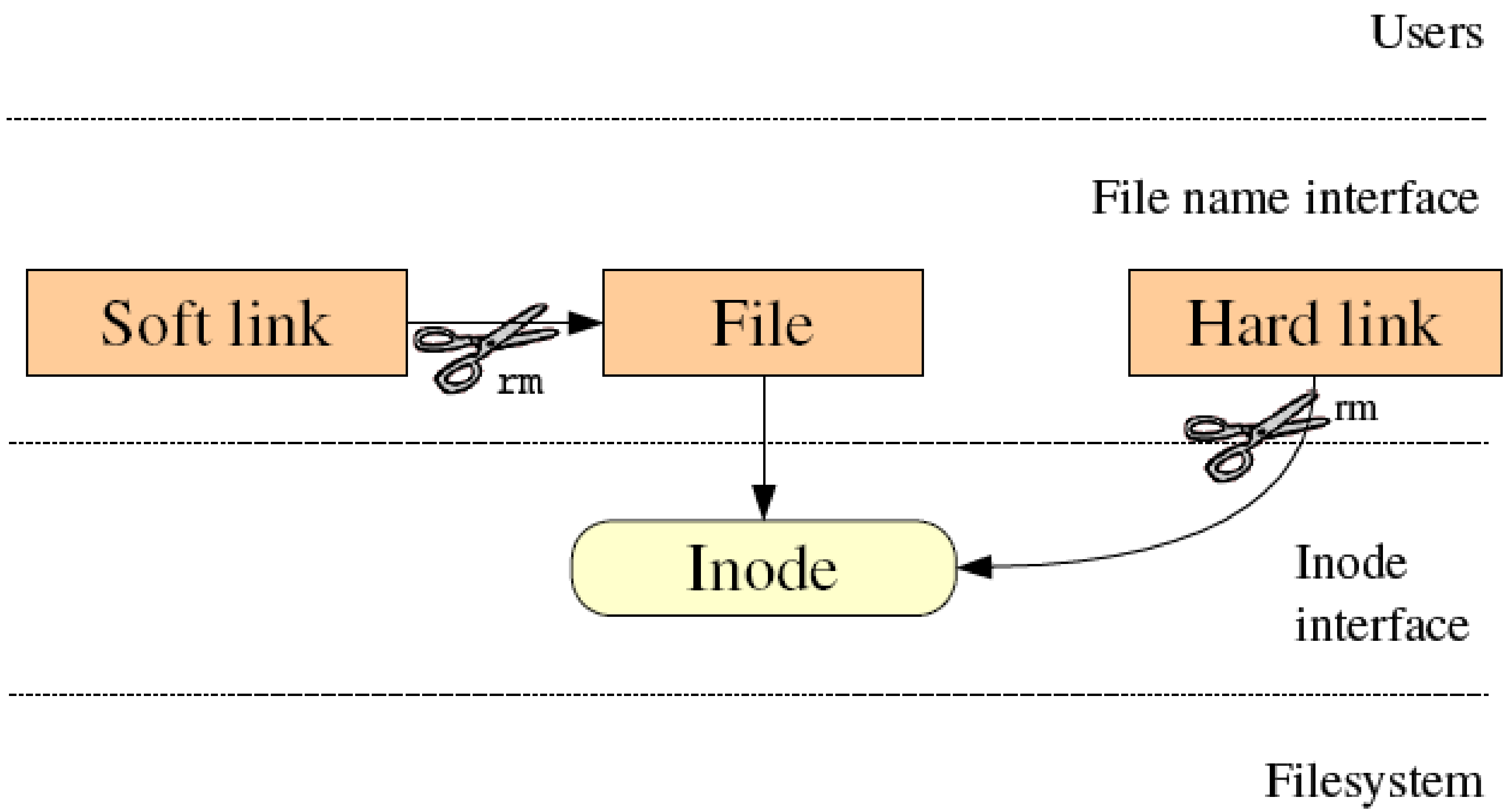
- To create hardlink

```
ln <target_file> <link_name>
ln <target_dir> <link_name>
```

- To remove a link

```
rm <link_name>
```

## Linking – visual understanding



## Command - link

- link command is almost same as ln
- Format:

```
link src-file link-file
```

- Deleting the link file has no effect on the source file

## Command - find

- Searches for a file in the directory hierarchy
- Format:
  - find [-H] [-L] [-P] [path...] [expression]
- Examples:

```
find . -name "*.pdf"
```

Lists all the *.pdf files in the current (.) directory or subdirectories. You need the double quotes to prevent the shell from expanding the * character.

```
find docs -name "*.pdf"
```

Finds all the *.pdf files in the docs directory

```
find / -name "*.pdf"
```

Finds all the *.pdf files starting from root folder

## Command - cut

- Used to select parts of file(s) and display on the standard output
- Format: cut [OPTION]... [FILE]...

- Main options

  -b select only these bytes
  -c select only these characters
  -d use DELIM instead of TAB for field delimiter
  -f select only these fields

- You can use one, and only one option from -b, -c or -f.

  N N'th byte, character or field, counted from 1
  N- from N'th byte, character or field, to end of line
  N-M from N'th to M'th (included) byte, character or field
  -M from first to M'th (included) byte, character or field

- Examples:

  `cut -b5-10 file.txt` (Displays bytes 5-10 of each line of file.txt)

  `cut -c-10 file.txt` (Displays first 10 of each line of file.txt)

  `cut -d ' ' -f2` (Displays the text between 1st and 2nd space character in file.txt)

## Command pg

- Paginates the file selected, for viewing one screen full at a time
- Format:
  - pg [-number] [-p string] [-cefnrs] [+ linenumber] [+/ pattern /] [filename]...
- Important options:

  -number – window size in lines

  +linenumber – start display at <linenumber>

  +/pattern/ - start display at first instance of /pattern/

  filename – file to be displayed, can be several

  n – a number will take you the nth screen

# Command - tr

- Translate, squeeze, and/or delete characters from standard input, writing to standard output.
- Format: `tr [OPTION]... SET1 [SET2]`

- Main options:

  -d delete characters in SET1, do not translate

  -s replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character
- SETs are sequences of characters, and the following sequences are recognized:

| | |
|---|---|
| \\ | backslash |
| \b | backspace |
| \c | suppress trailing newline |
| \n | new line |
| \r | carriage return |
| \t | horizontal tab |

## Examples

- `cat file0.txt | tr -s ' '`
  - removes multiple spaces and replaces by single space
- `cat file0.txt | tr -s ' \n'`
  - removes multiple spaces and newlines, and replaces by single space and newline, respectively

## Command - diff

- Shows differences between two files or folders, line by line
- Format: `diff [OPTION]... FILES`

- Main options

| | |
|---|---|
| -i --ignore-case | Ignore case differences in file contents. |
| -b --ignore-space-change | Ignore changes in the amount of white space. |
| -w --ignore-all-space | Ignore all white space. |
| -B --ignore-blank-lines | Ignore changes whose lines are all blank. |
| -a --text | Treat all files as text. |
| -q --brief | Output only whether files differ. |
| --suppress-common-lines | Do not output common lines. |
| -r --recursive | Recursively compare any subdirectories found. |
| -s --report-identical-files | Report when two files are the same |
| -i --ignore-case | Ignore case differences in file contents. |

## Examples

- `diff file0.txt file1.txt`
  - Shows line by line difference
- `diff -r folder1 folder2`
  - Shows line by line differences, in same files in the two folders
- `diff -bB file0.txt file1.txt`
  - Shows the differences, while ignoring all whitespaces and blank lines

## Command - cmp

- Compares two files, byte by byte
- Format:

```
cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]
```

- Main options

| | | |
|---|---|---|
| -b | --print-bytes | Print differing bytes. |
| -i SKIP | --ignore-initial=SKIP | Skip the first SKIP bytes of input. |
| -i SKIP1:SKIP2 | --ignore-initial=SKIP1:SKIP2 | Skip the first SKIP1 bytes of FILE1 and the first SKIP2 bytes of FILE2. |
| -l | --verbose | Output byte numbers and values of all differing bytes. |
| -n LIMIT | --bytes=LIMIT | Compare at most LIMIT bytes. |
| -s | --quiet --silent | Output nothing; yield exit status only. |

## Examples

- `cmp file0.txt file1.txt`
  - Compares the two files and shows where the 1st difference occurs
- `cmp -b file0.txt file1.txt`
  - Compares the two files and shows where the 1st difference occurs, and prints the differing bytes

## Command - who

- Shows who is logged on
- Format:

```
who [OPTION]... [ FILE | ARG1 ARG2 ]
```

- Main options:

| | |
|---|---|
| -H, --heading | print line of column headings |
| -l, --login | print system login processes |
| -m | only hostname and user associated with stdin |
| -p, --process | print active processes spawned by init |
| -q, --count | all login names and number of users logged on |
| -r, --runlevel | print current runlevel |
| -u, --users | list users logged in |

## Command - which

- Displays the full path of shell commands but only if the command is present in the system path
- Format:

```
which <progname>
```

## Command - date

- Prints the system date and time
- Can also update it
- Format

```
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

- Main options

| | |
|---|---|
| -d, --date=STRING | display time described by STRING, not `now' |
| -r, --reference=FILE | display the last modification time of FILE |
| -s, --set=STRING | set time described by STRING |
| -u, --utc, --universal | print or set Coordinated Universal Time |

## Examples

- `date -d "11/20/2003 12:48:00‟`
  - Displays date as Thu Dec 23 00:00:00 IST 2010
- `date -s "11/20/2003 12:48:00‟`
  - Sets system date as Thu Dec 23 00:00:00 IST 2010
- `date -r file0.txt`
  - Shows the last updated date and time of file0.txt

## Command - time

- Used for measuring time taken to execute a command or a sequence of commands
- It shows the real time, user time and system time
- Format:

```
time [-p] utility [argument]...
```

- Examples:

```
time ls -la
```

  - Shows the three times mentioned above, taken to execute ls –la command

## Command - script

- The script utility makes a record of everything printed on your screen. The record is written to filename. If no file name is given, the record is saved in the file typescript.
- Format:

```
script [-a] [filename]
```

  - if –a is provided with a filename the session is stored in filename
  - If –a is not provided then the session is stored in typescript file

## Command - bc

- Text mode calculator

## Questions??

Our contact details:

**NxNVision Solutions, Chennai**

**Email (primary) - info@nxnvision.com**

**Email (alternate) - nxnvision@gmail.com**

**Facebook: www.facebook.com/nxnvision**

**Website: http://www.rupamdas.com, www.nxnvision.com**

**Phone: +91 98407 84107**