INTRODUCTION TO LINUX

LEC 2

Agenda

- Interactive operating system
- Operating system user interface
- Implementation of Operating System
- Linux History
- Useful commands
- Lab

What is interactive Operating System?

- An operating system that allows users to run interactive programs. Pretty much all operating systems that are on PCs are interactive OS's.
- Accepting input from a human. Interactive computer systems are programs that allow users to enter data or commands. Most popular programs, such as word processors and spreadsheet applications, are interactive.
- A non interactive program is one that, when started, continues without requiring human contact. A compiler is a non interactive program, as are all batch processing applications.

Interactive operating system example

- In an interactive operating system, the user interacts directly with the operating system to supply commands and data as the application program executes and the user receives the results of processing immediately. The user is in direct two way communication with the computer.
- **Example:** ATM.

Operating System UI’s

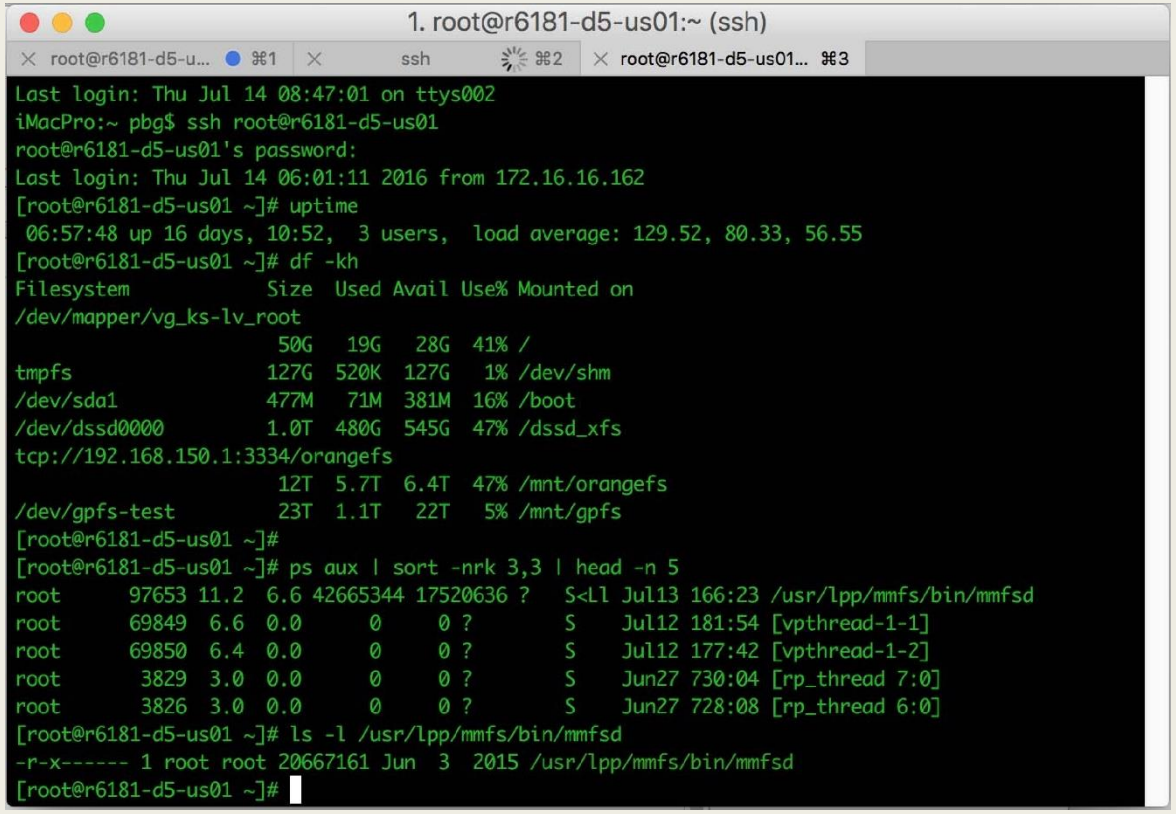
- A personal computer's user interface includes a display device, mouse, and keyboard that allow the user to view and manipulate the computing environment.
- It also includes software elements, such as icons, menus, and toolbar buttons.

- **User interface -** Almost all operating systems have a user interface (**UI**).
  - **Command-Line** (**CLI**)
  - **Graphics User Interface** (**GUI**)
  - **touch-screen**

Command Line interpreter

- CLI allows direct command entry
- Sometimes implemented in kernel, sometimes by systems program
- Sometimes multiple flavors are implemented – **shells**
- Primarily fetches a command from the user and executes it.
- CLI enables users to type commands in a terminal or console window to interact with an operating system.
- Users type a command or series of commands for each task they want to perform.

Shell Command Interpreter



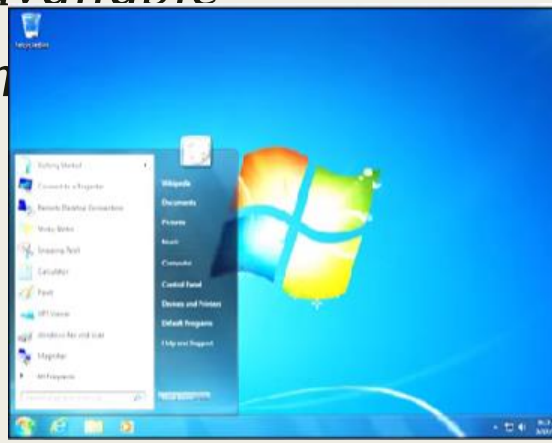
User Operating System Interface - GUI

- A graphical user interface (GUI) uses graphics, along with a keyboard and a mouse, to provide an easy-to-use interface to a program.

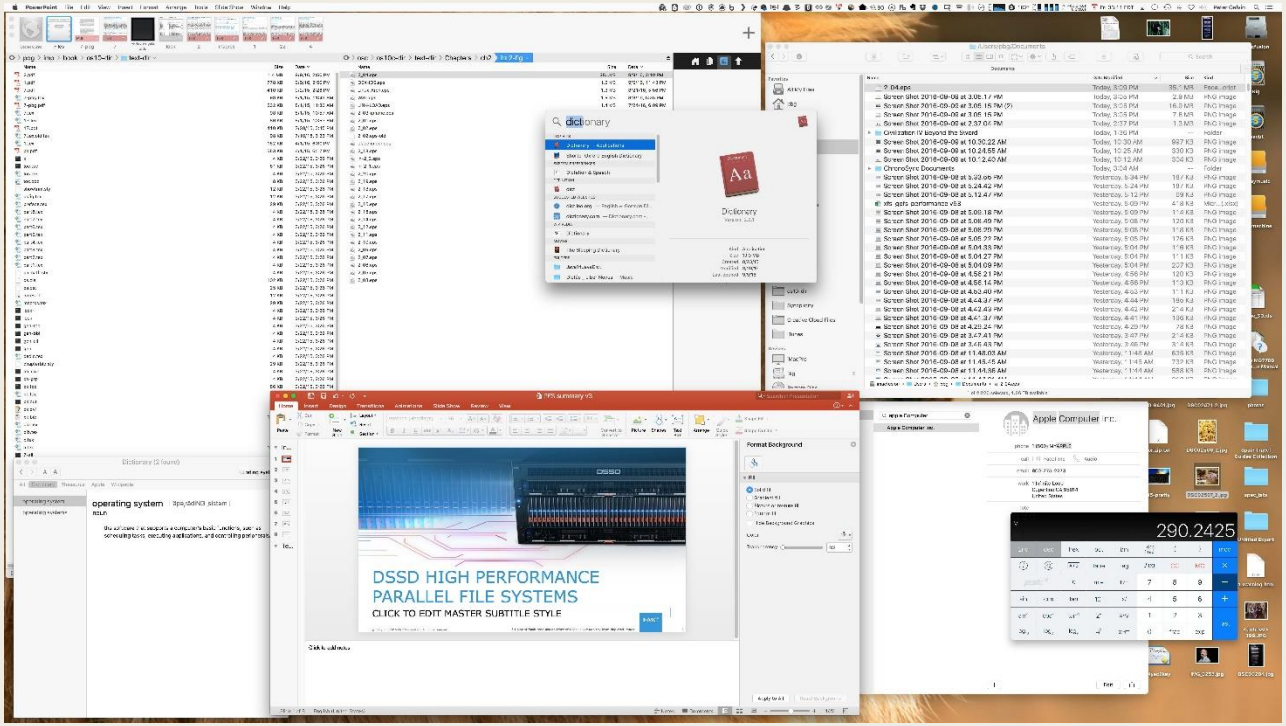
- User-friendly **desktop** metaphor interface
  - *Usually mouse, keyboard, and monitor*
  - ***Icons*** *represent files, programs, actions, etc*
  - *Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a* ***folder****)*

User Operating System Interface - GUI

- Many systems now include both CLI and GUI interfaces
  - *Microsoft Windows is GUI with CLI "command" shell*
  - *Apple Mac OS X is "Aqua" GUI interface with UNIX kernel underneath and shells available*
  - *Unix and Linux have CLI with option interfaces (CDE, KDE, GNOME)*

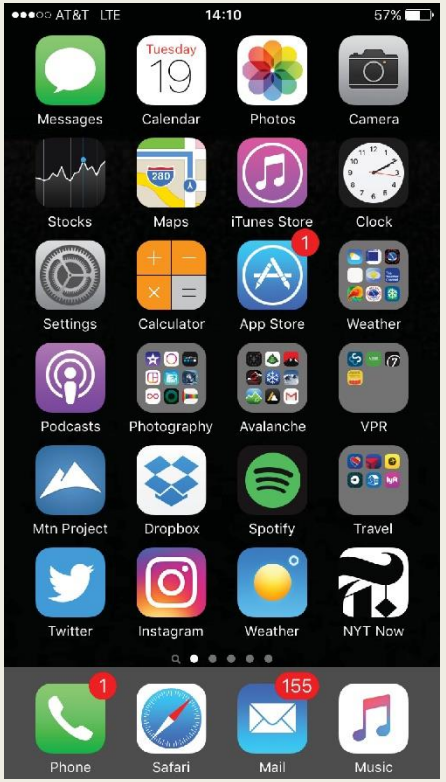


The Mac OS X GUI



Touchscreen Interfaces

- Touchscreen devices require new interfaces
  - *Mouse not possible or not desired*
  - *Actions and selection based on gestures*
  - *Virtual keyboard for text entry*
- Voice commands



Design and Implementation

- Design and Implementation of OS is not "solvable", but some approaches have proven successful
- Internal structure of different OS can vary widely
- Start the design by defining goals and specifications
- Affected by choice of hardware, type of system
- **User** goals and **System** goals
  - ***User goals*** – *OS should be convenient to use, easy to learn, reliable, safe, and fast*
  - ***System goals*** – *OS should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient*
- Specifying and designing an OS is a highly creative task of **software engineering**

Design and Implementation

Policy and Mechanism

- **Policy: What** needs to be done?
  - *Example: Interrupt after every 100 seconds*
- **Mechanism: How** to do something?
  - *Example: timer*
- Important principle: separate policy from mechanism
- The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later.
  - *Example: change 100 to 200*

Operating system Implementation

- **Much variation**
  - *Early OSes in assembly language*
  - *Now C, C++*
- **Actually usually a mix of languages**
  - *Lowest levels in assembly*
  - *Main body in C*
  - *Systems programs in C, C++, scripting languages like PERL, Python, shell scripts*
- More high-level language easier to **port** to other hardware
  - *But slower*

Operating System Design and structure

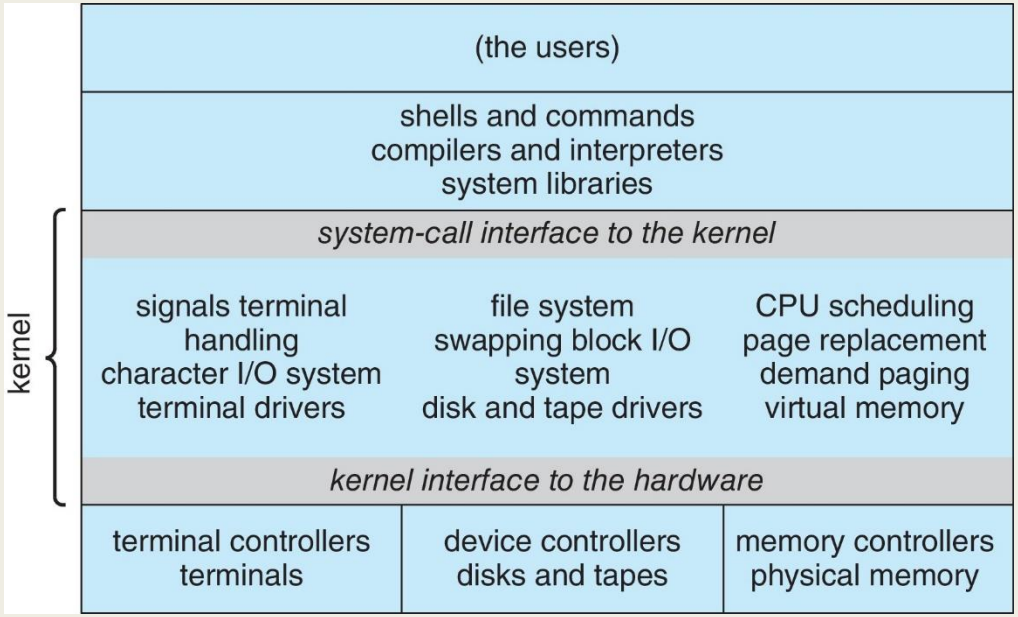
- General-purpose OS is very large program
- Various ways to structure ones:
  - *Simple structure – MS-DOS*
  - *More complex – UNIX*
  - *Layered – an abstraction*
  - *Microkernel – Mach*

Monolithic Structure – Original UNIX

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.
- The UNIX OS consists of two separable parts
  - *Systems programs*
  - *The kernel*
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

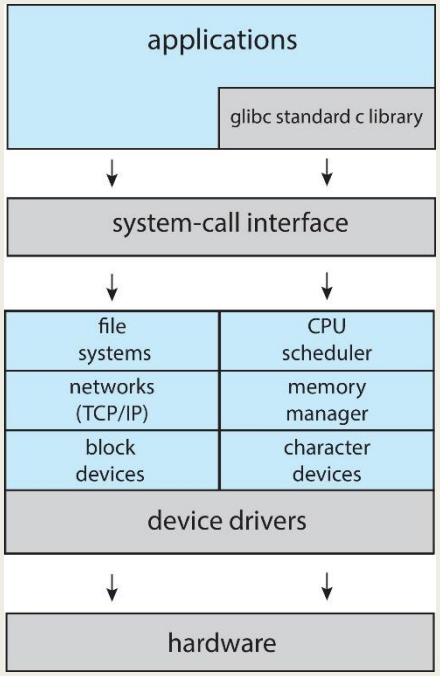
Traditional UNIX System Structure

Beyond simple but not fully layered



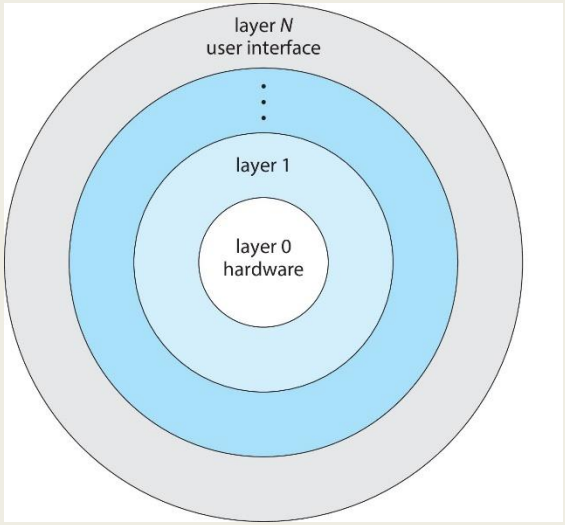
Linux System Structure

Monolithic plus modular design



Layered Approach

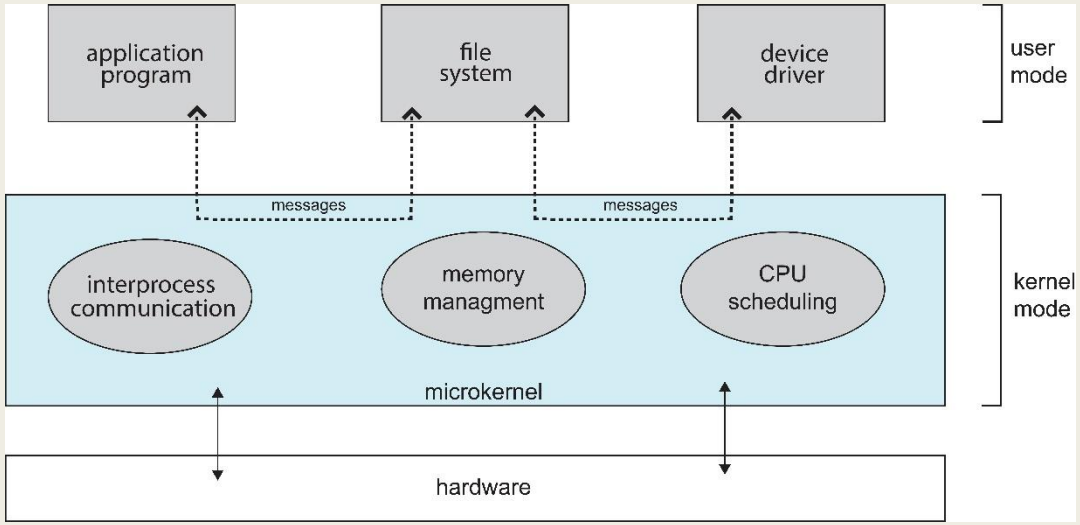
- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers



Microkernels

- Moves as much from the kernel into user space
- **Mach** is an example of **microkernel**
  - *Mac OS X kernel (**Darwin**) partly based on Mach*
- Communication takes place between user modules using **message passing**
- Benefits:
  - *Easier to extend a microkernel*
  - *Easier to port the operating system to new architectures*
  - *More reliable (less code is running in kernel mode)*
  - *More secure*
- Detriments:
  - *Performance overhead of user space to kernel space communication*

Microkernel System Structure



Modules

- Many modern operating systems implement **loadable kernel** mo**dules** (**LKMs**)
  - *Uses object-oriented approach*
  - *Each core component is separate*
  - *Each talks to the others over known interfaces*
  - *Each is loadable as needed within the kernel*

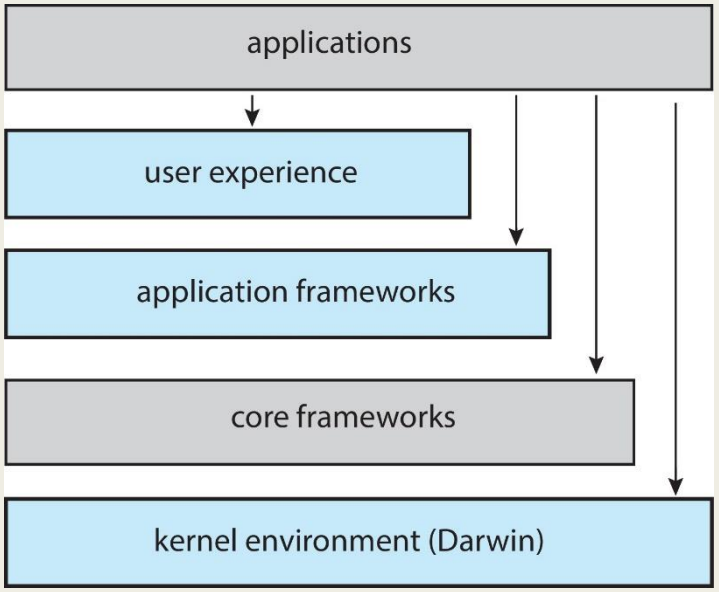
- Overall, similar to layers but with more flexible
  - *Linux, Solaris, etc.*

Hybrid Systems

- Most modern operating systems are not one pure model:
  - *Hybrid combines multiple approaches to address performance, security, usability needs*
  - *Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality*
  - *Windows mostly monolithic, plus microkernel for different subsystem* ***personalities.***

- Apple Mac OS X hybrid, layered, **Aqua** UI plus **Cocoa** programming environment:

macOS and iOS Structure



Early Computing History

- In the 1940s and 1950s, all computers were personal computers in the sense that a user would sign up to use the machine and then take over the whole machine for that period.

- The early 1960s were dominated by batch systems in which a user would submit a job on punched cards and wait, usually hours, before any printed output appeared.

Early Computing History

- To get around this unproductive environment, the concept of timesharing was invented by M.I.T.

- General Electric (GE)
- Bell Labs
- MTI

They created a second generation timesharing system named MULTICS (Multiplexed Information and Computing Service).

Early UNIX History

- Bell Labs, in further developments of what was now called UNICS 1970.
- the operating system in high-level language of his own design which he called B.
- The B language lacked many features then they designed a successor to B which he called C.
- They then rewrote UNIX in the C programming language to aid in portability.

GNU

- GNU is a project started by Richard M. Stallman (RMS) to write a completely free implementation of Unix available.
- GNU stands for “GNU is Not Unix”

- It is Free software
  - *The freedom to run the software, for any purpose*
  - *The freedom to study how the program works, and adapt it to your needs*
  - *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits*

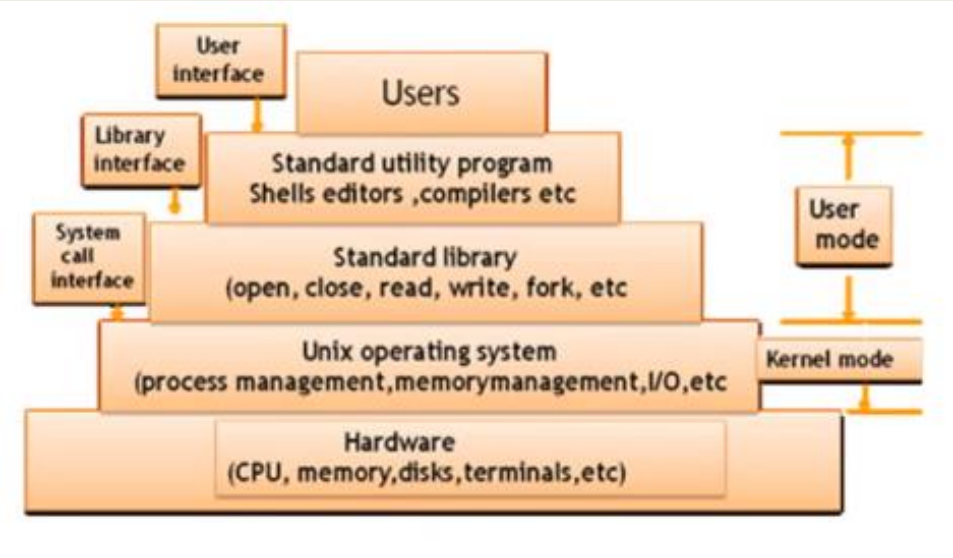
What is LINUX

- LINUX is free …you can edit the source code
- It is fully customizable ….you can edit the application
- It is stable
- Linux has better security
- Multi-user, Multitasking
- Coexists with other OS
- Runs with multiple platforms

Layers of LINUX/UNIX

LINUX/UNIX has three main important parts:

- Kernel
- Shell
- File system

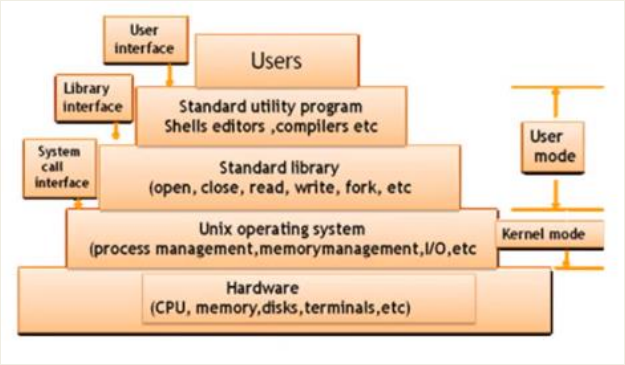


Layers of LINUX/UNIX

User interface: how the user is communicating with the shell.

Library interface: how the shell deals with the standard libraries.

System call interface: how libraries manage the hardware resources through the OS.



Getting Started

- Use username and password for login.
- Linux is case sensitive.
- Password can be changed any time by the user.

Linux Commands

- Control Keys: control keys perform special function

Ctrl s ……..Pause display

Ctrl q ……..Restart display

Ctrl c ……..cancel operation

Ctrl u ……..cancel line

Ctrl v ……..Treat following control character as normal character

Linux Commands

- Getting Help: in Linux whenever you need help, type “man” followed by the command name.

man command

i.e. man su

To display all the help options try

man -- help

Linux Commands

- Linux command options

Options	Descriptions
-a	lists all the files and directories, even hidden ones which are preceded by (“.”)
-l	lists the size, creation date and permissions about all the files and directories in the current directory
-d	lists the directory
-c	don’t create file if it already present
-f	Force
-k	block Size
-R	Recursive
-t	Type
-V	version.

Linux Commands

Activity:

Move from current location to home

cd /home/

Display what inside home

ls

Move inside linux

cd linux

Move back to the prevous location (2 levels backword)

cd ../../

To go from tty to another

chvt 5

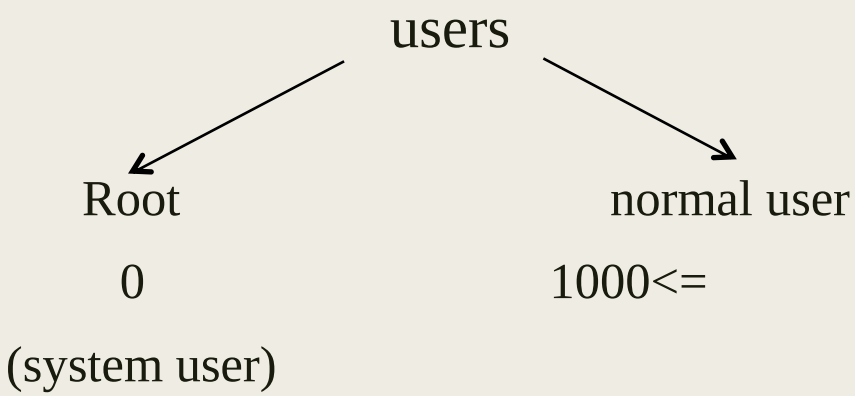
Getting Started

- Basic Commands:

- To display the date: date
- To display the calendar: cal
- To display the calendar for specific year : cal 2020
- To display the calendar for specific month: cal 3 2020

- To eject CD from the machine: eject
- To enter CD to your machine: eject -t

System Users



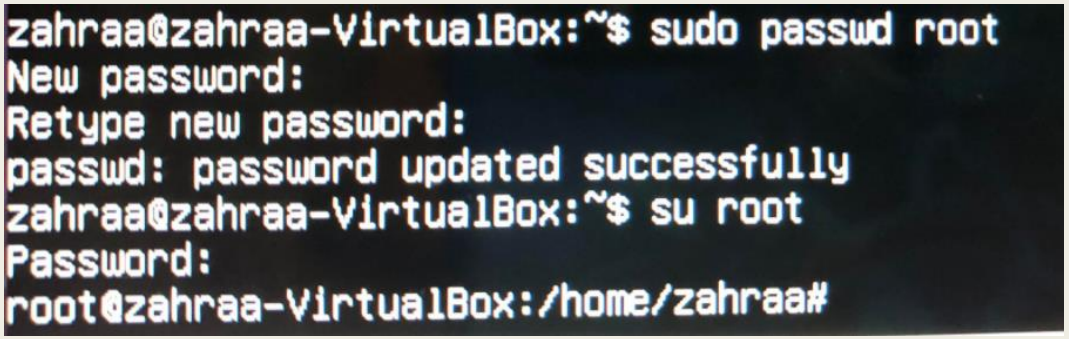
To see all users you have created: cat /etc/passwd

To display you ID: id

To display any user ID: id username

To create a root user

- Root creation



- To switch back to a regular user ➔Su username

su zahraa

System Files ‘linux’

/ → home →…

→ var

→etc

→dev

→tmp

→…

System Files ‘linux’

Home → contains home directories

i.e /home/saja

etc → contains all machine configurations

i.e your IP address for internet connection

Dev → each hardware component treated as a file, these files saved on Dev

i.e mouse, keyboard,…

Tmp → contains temporary files

i.e streaming files

System Files ‘linux’

usr → contains shared files between users

i.e background options, help options, …

var → contains variable files

i.e time you locked in, your emails,…..

bin, sbin→ contains administrative commands.

i.e mouse, keyboard,…

root → contains files for root user

i.e password

System Files ‘linux’

proc → contains all information about your hardware and software systems

i.e RAM size, processor status network info.,…

Opt →contains all files related to installed programs

i.e matlab, python

Media → for removable devices

i.e flash memory

Boot →contains kernel information

Lab

- To display the root files

  ls /

  It displays : home lib dev boot …..

- To display what is inside any of these files

  ls /home

  It displays : linux

- To display what is inside linux

  ls /home/zahraa/

Linux Commands

mkdir command

To create a new directory .

mkdir_directorymane to create called directory name

mkdir –p dir1/dir2/dir3 it will create the directory tree. Dir3 is created under dir2 is created under dir1

Cd ! It will point the location of the last created dir

Lab

- To create a file

Touch file1

- To create a file

Touch file2 file3