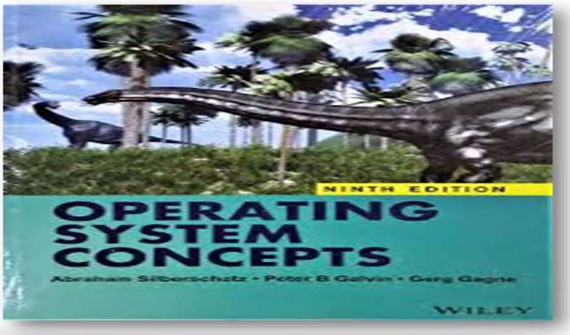
Concept of OS type of exam


NINTH EDITION
OPERATING SYSTEM CONCEPTS
Abraham Silberschatz • Peter B Galvin • Greg Gagne
WILEY

Technical modules 8

Module 1 : COSSDM = COS = SDM

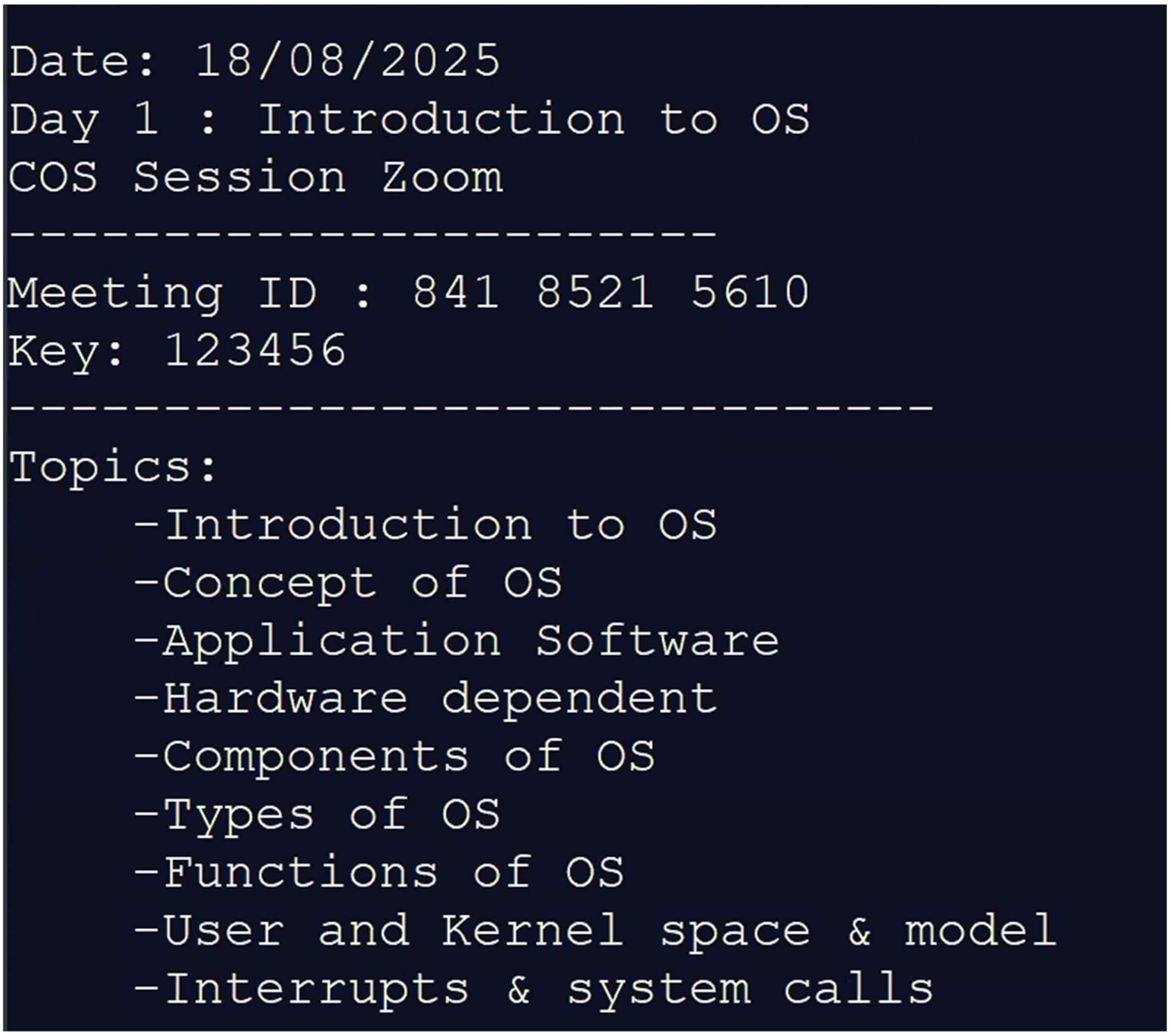
Evolution :100 Marks

Type 1 : CEE = 40marks

Type 2 : Internal exam

1 IA : MCQ Test:20 marks

2. Lab :coding : 40 marks



```
Date: 18/08/2025
Day 1 : Introduction to OS
COS Session Zoom
-----------------------
Meeting ID : 841 8521 5610
Key: 123456
------------------------------
Topics:
    -Introduction to OS
    -Concept of OS
    -Application Software
    -Hardware dependent
    -Components of OS
    -Types of OS
    -Functions of OS
    -User and Kernel space & model
    -Interrupts & system calls
```

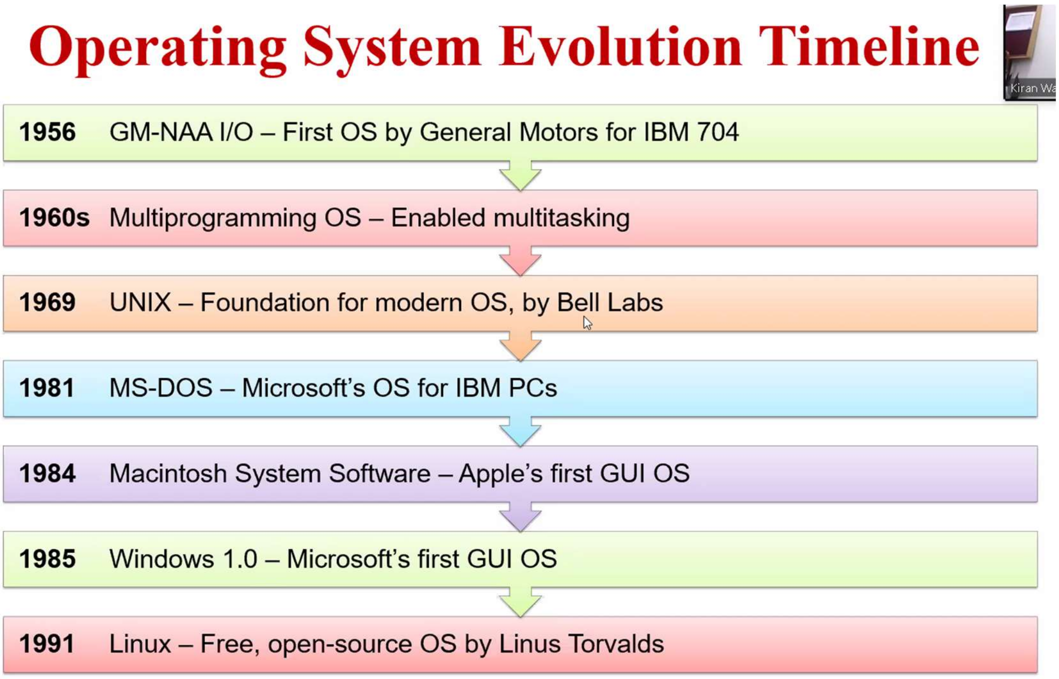
Notes - for concept of operating system

Module – 1

Introduction of OS

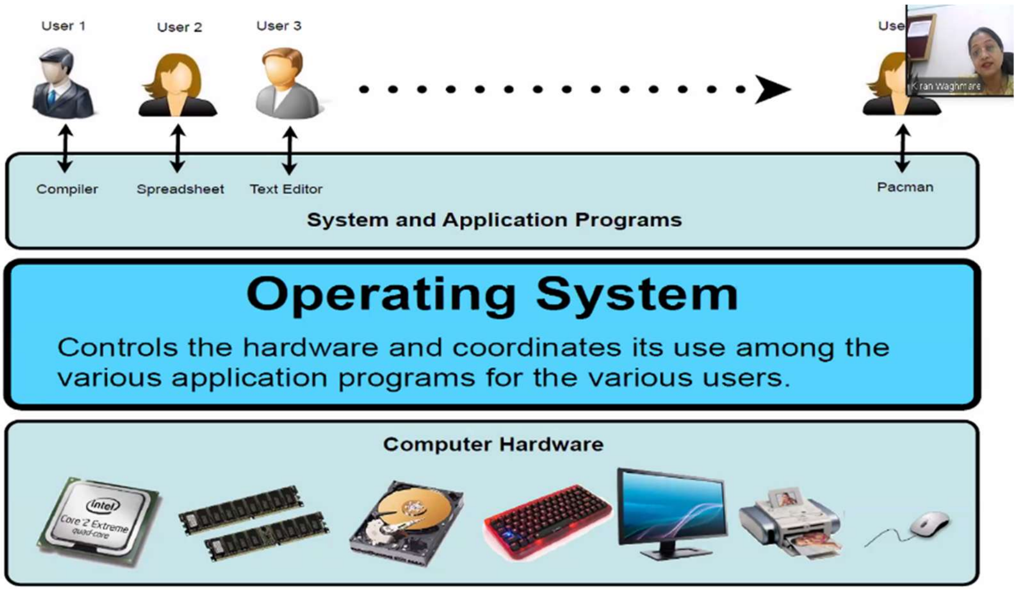
1 There are lot of Os systems that are –

Linux, Android, Ubuntu, Windows, etc



Operating System Evolution Timeline

1956	GM-NAA I/O – First OS by General Motors for IBM 704
1960s	Multiprogramming OS – Enabled multitasking
1969	UNIX – Foundation for modern OS, by Bell Labs
1981	MS-DOS – Microsoft's OS for IBM PCs
1984	Macintosh System Software – Apple's first GUI OS
1985	Windows 1.0 – Microsoft's first GUI OS
1991	Linux – Free, open-source OS by Linus Torvalds


User 1 User 2 User 3 Use

Compiler Spreadsheet Text Editor Pacman

System and Application Programs

Operating System

Controls the hardware and coordinates its use among the various application programs for the various users.

Computer Hardware

OS is program that is communicate with a user

System Software – All that apps that are automatically installs once we install OS

Eg, Notepad , Wordpad

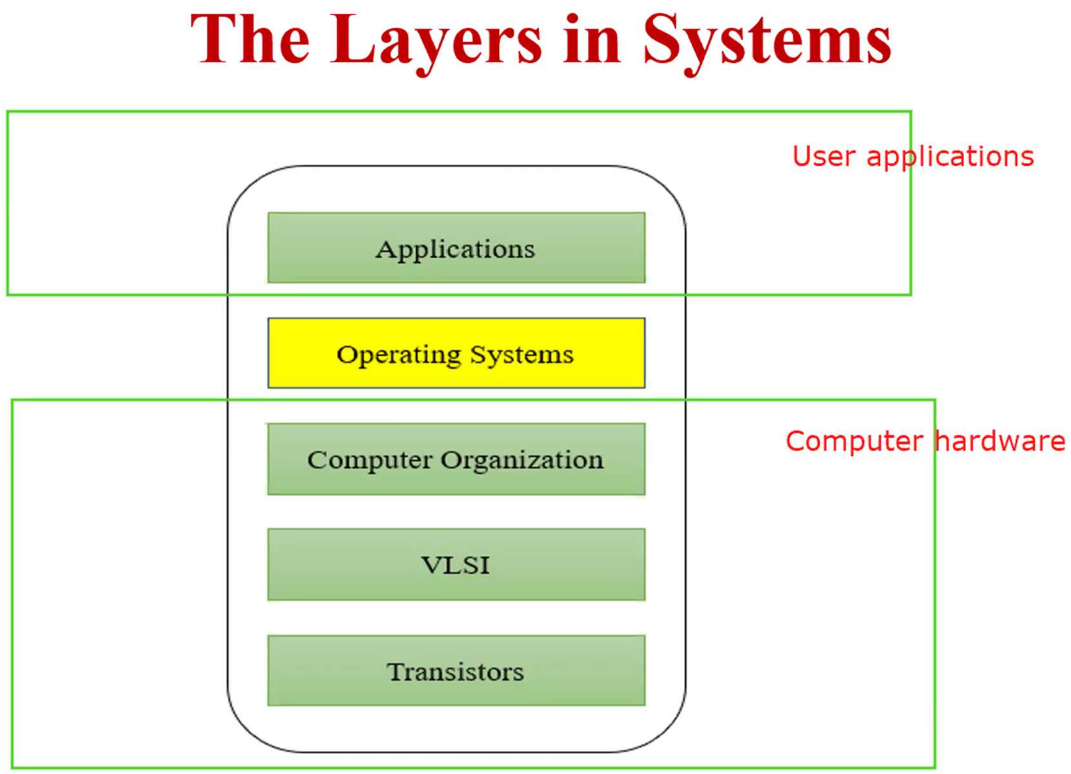
Application software – All apps that are performs users task it is called as application software.

Eg –, Chrome, Office.

Operating system definition – Is a program that manages computer hardware

It is also provides a basis for application programs and acts as a intermediary between user and computer

The Layers in system

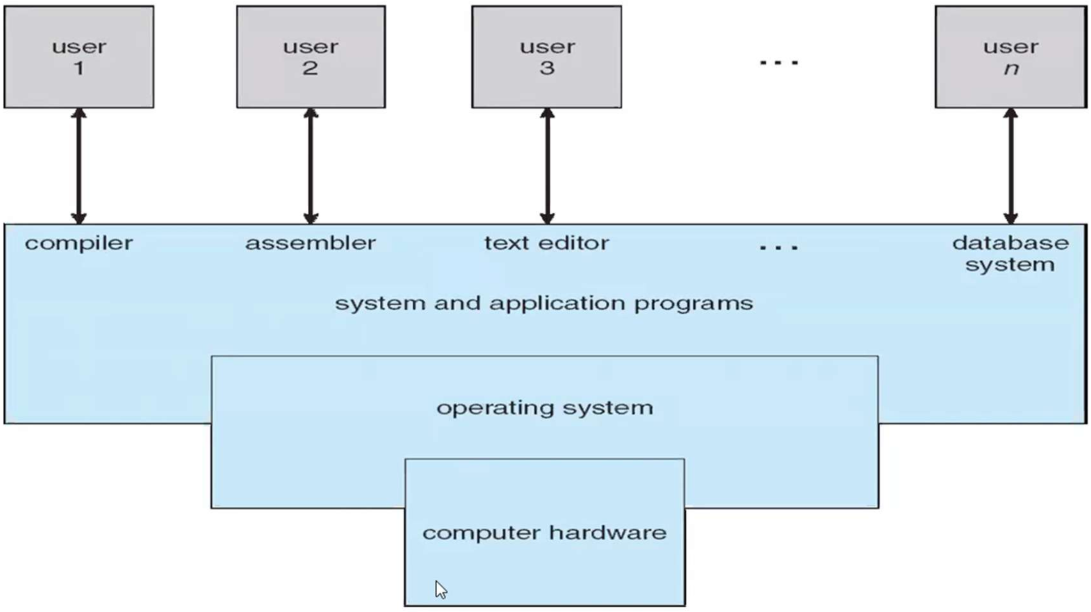


Computer abstraction – without giving any details

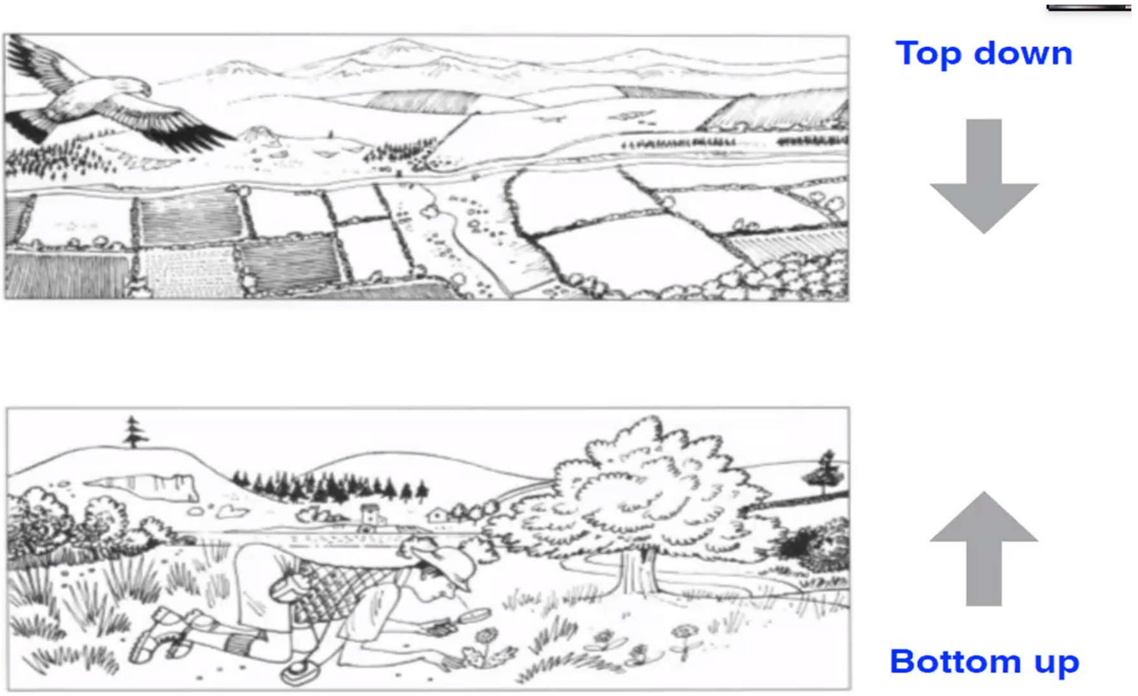
Resource management – OS automatically gives support to the application

Bus – it is a pipleline that communicate with the diff layers

- 1st layer – User layer
- 2nd layer application program
- 3rd OS
- 4th Hardware



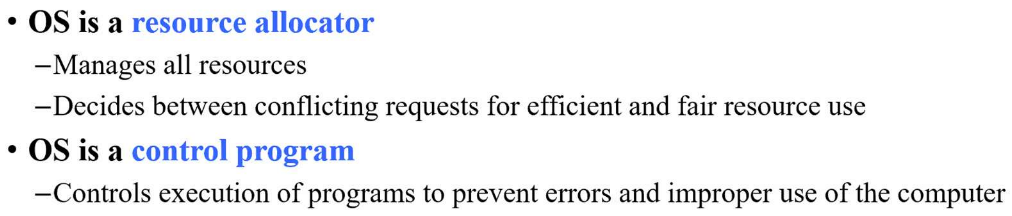
Two approaches –



Top down –

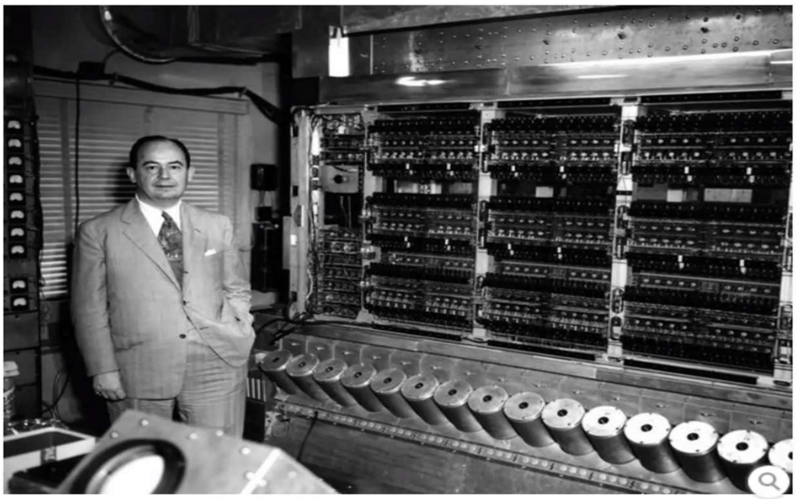
Bottom up – Gives Micro details

Kernal – one simple program which runs all the time in computer in form of (shell script)



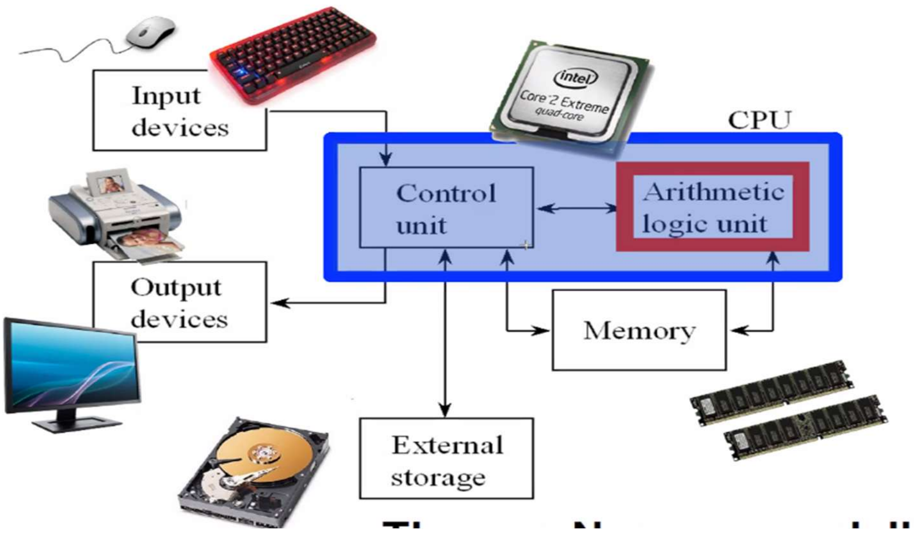
- **OS is a resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- **OS is a control program**
  - Controls execution of programs to prevent errors and improper use of the computer

First generation computer –



John von Neumann with the IAS Computer

Von Neumann persuaded IAS to expand from doing theoretical studies to building a real computer, with meteorology calculations as a key test of its scientific value. The cylinders at the bottom are the Williams–Kilburn memory tubes.

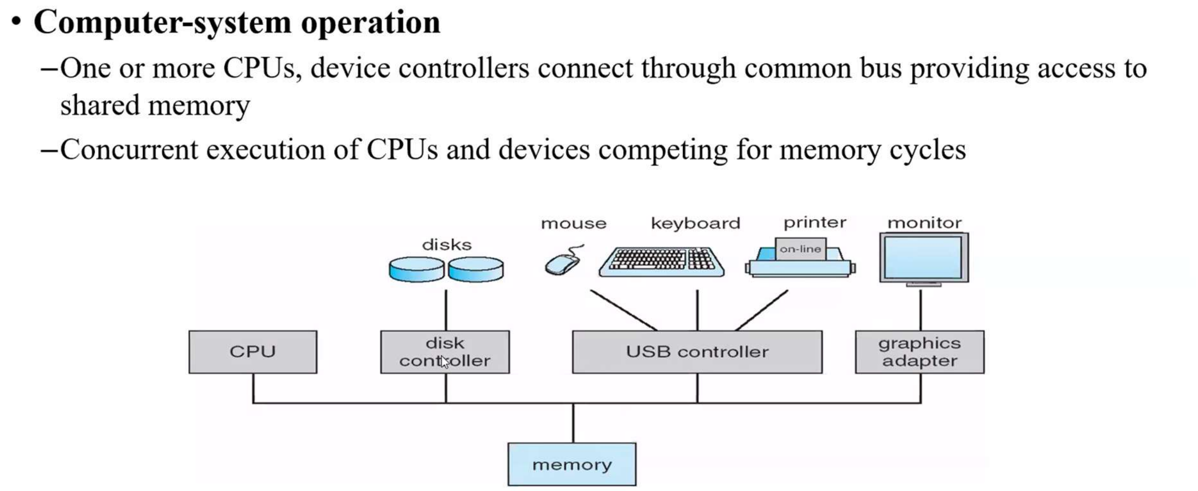


Input output bus –

CPU Bus - Processing – Mathematical computation

Ideal – No activity of CPU

Busy – CPU is doing tasks



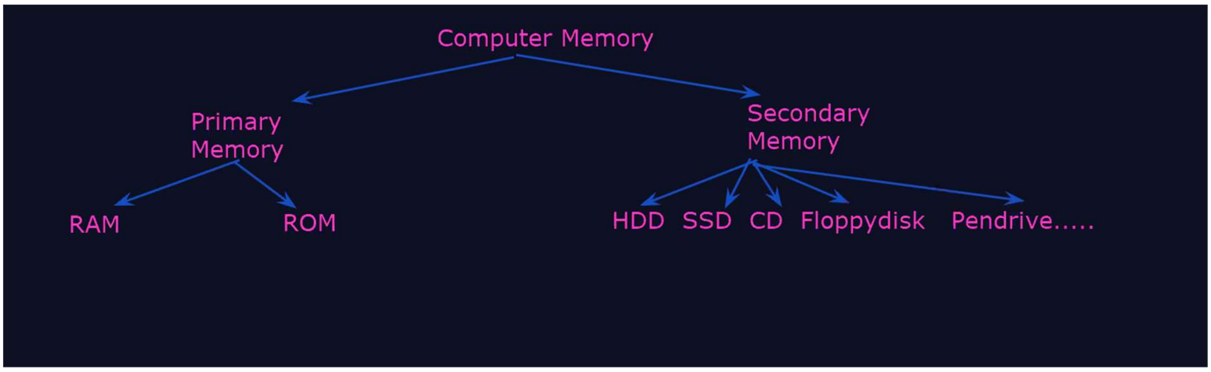
- **Computer-system operation**
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles

DMA Structure – Direct Memory Access Structure – We use it for High Speed IO actions.

Bootstap loder – Load that application into memory via shell scripting.

RAM - Volatile memory – It can change – Temporary stored

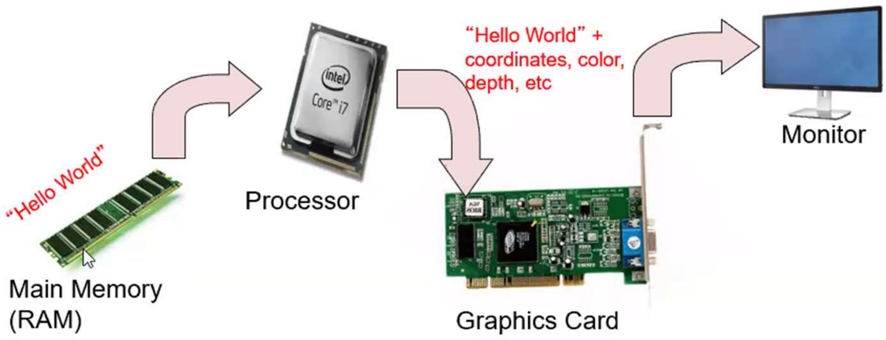
ROM – Once stored we can not edit. Permently stored memory.

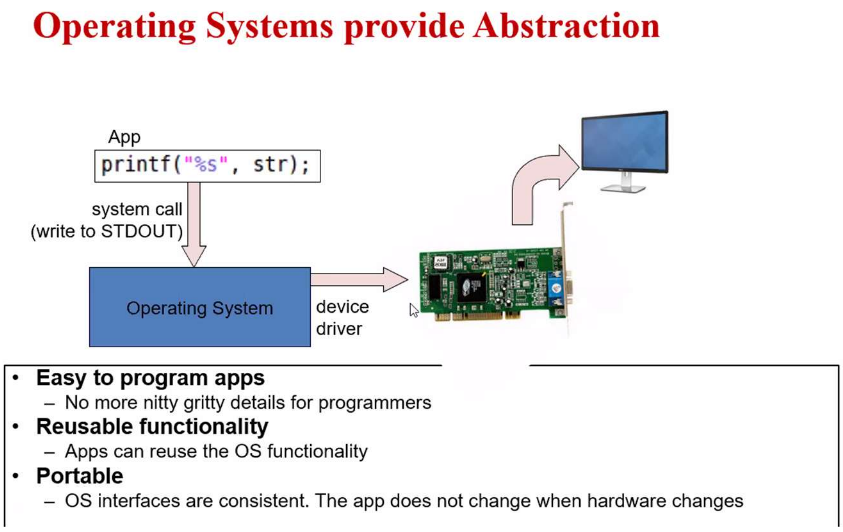


Primary – Inbuilt memory – RAM, ROM.

Secondary memory – Additional memory

Displaying on the screen!





Operating Systems provide Abstraction

App

```
printf("%s", str);
```

system call
(write to STDOUT)

Operating System

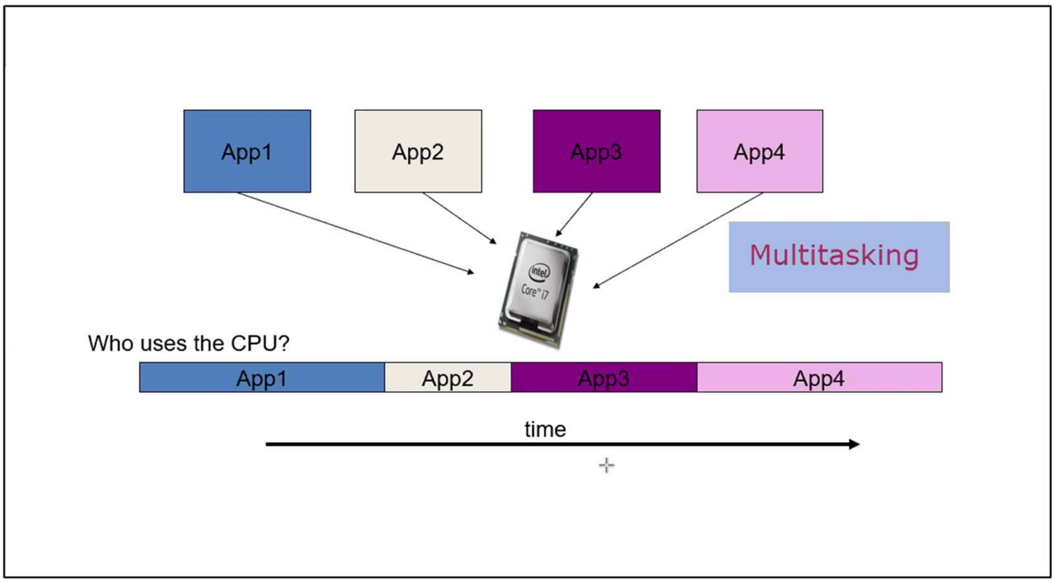
device
driver

- **Easy to program apps**
  - No more nitty gritty details for programmers
- **Reusable functionality**
  - Apps can reuse the OS functionality
- **Portable**
  - OS interfaces are consistent. The app does not change when hardware changes

System Call –

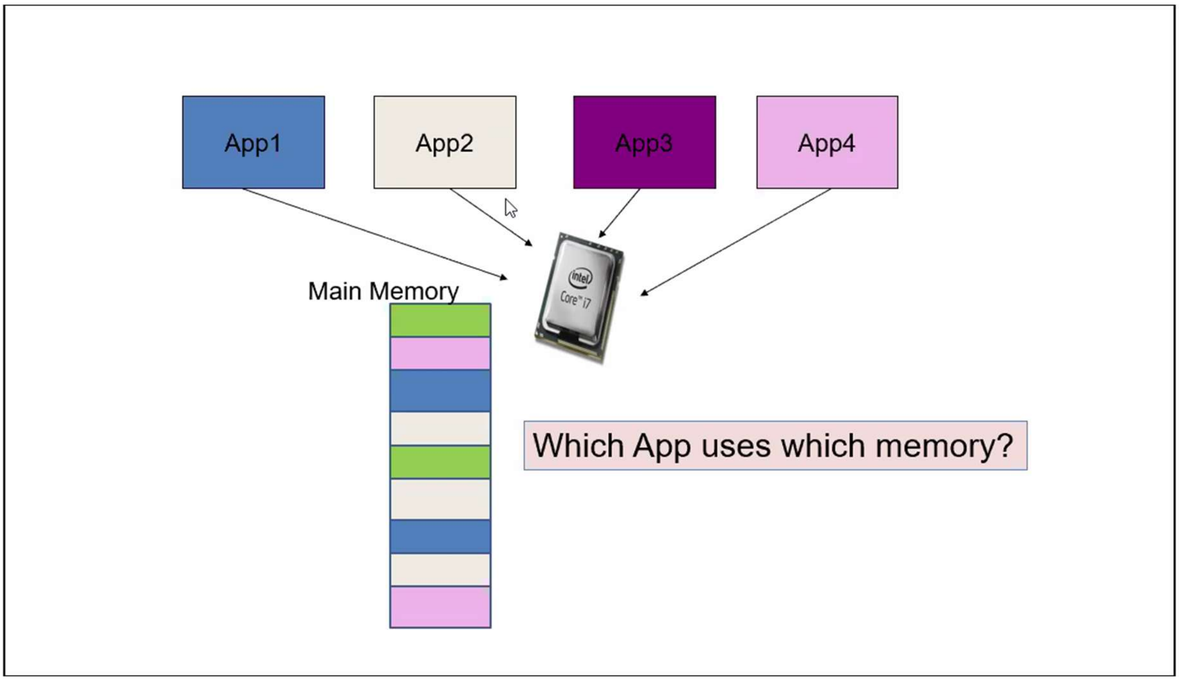
How Computer share’s the CPU

Sharing the CPU

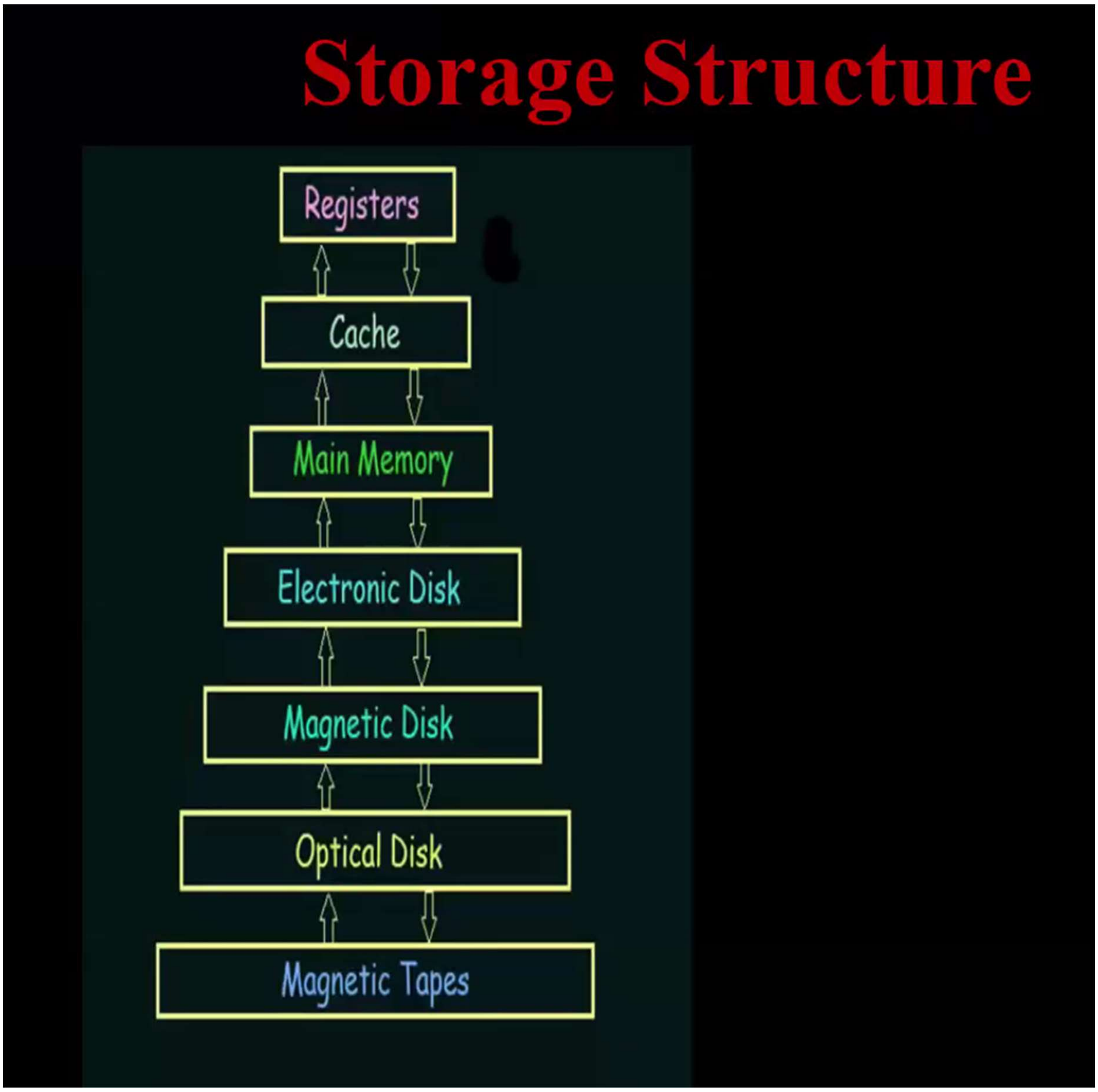


If CPU processer is busy – efficiency is good

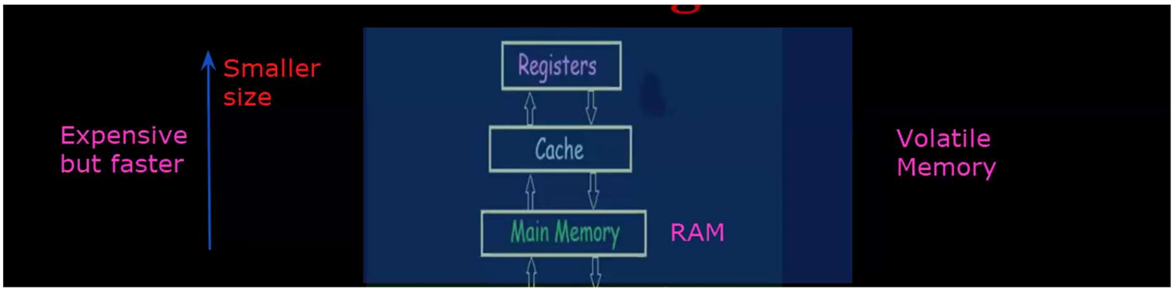
If CPU processor is Ideal – efficiency will be less

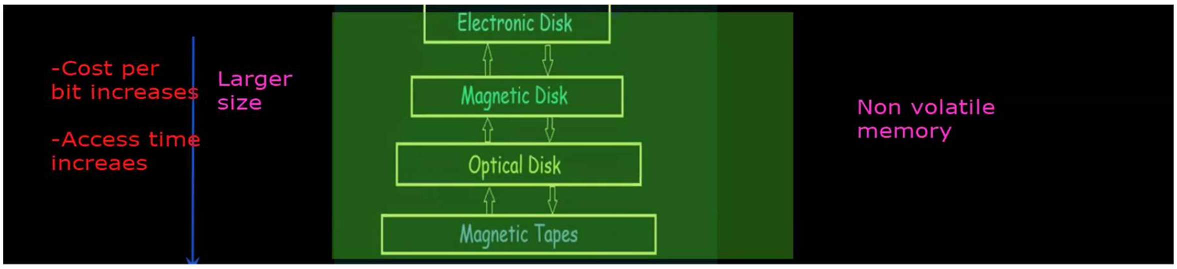


Sharing resources -We Share the same memory between multiple application.

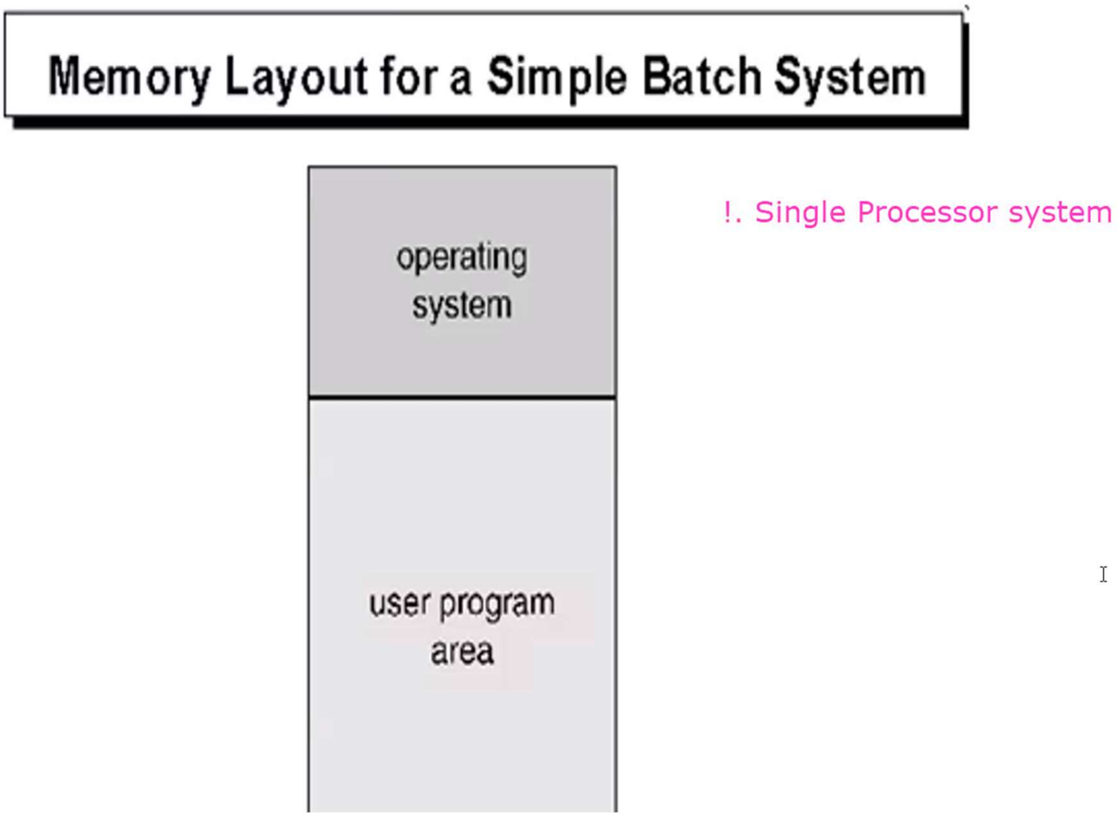


Storage Structure





Simple Batch System!

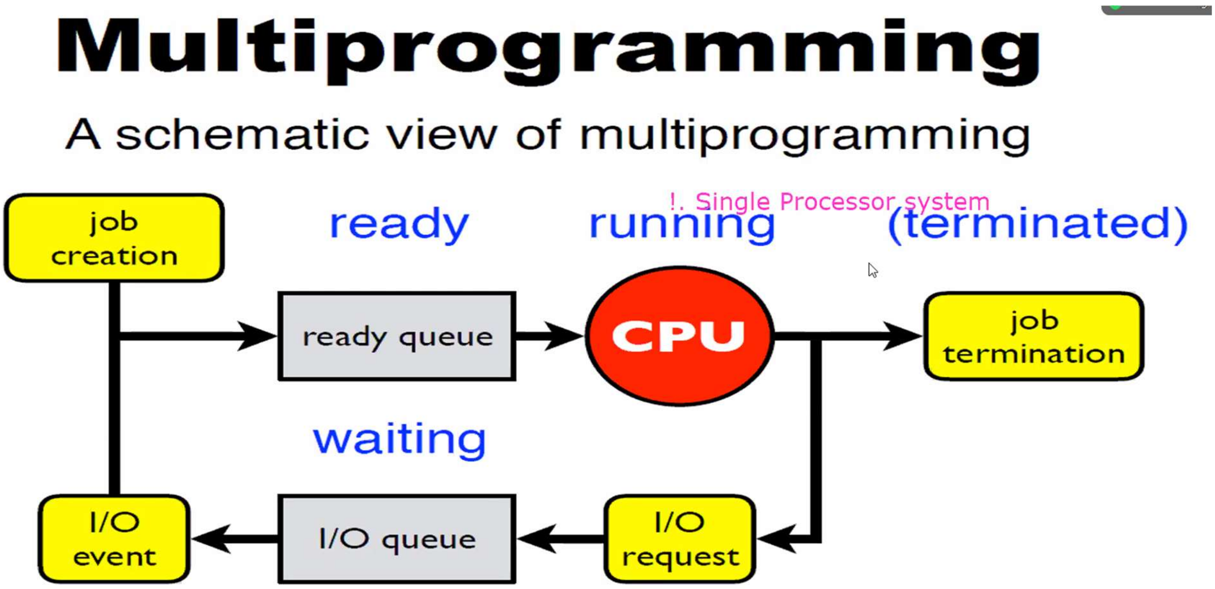


Memory Layout for a Simple Batch System

!. Single Processor system

Advantages – No Communication, between user and OS , No priority, every process is equal.

Multiple Programing!



Multiprogramming

A schematic view of multiprogramming

!. Single Processor system

ready

running

(terminated)

job creation

ready queue

CPU

job termination

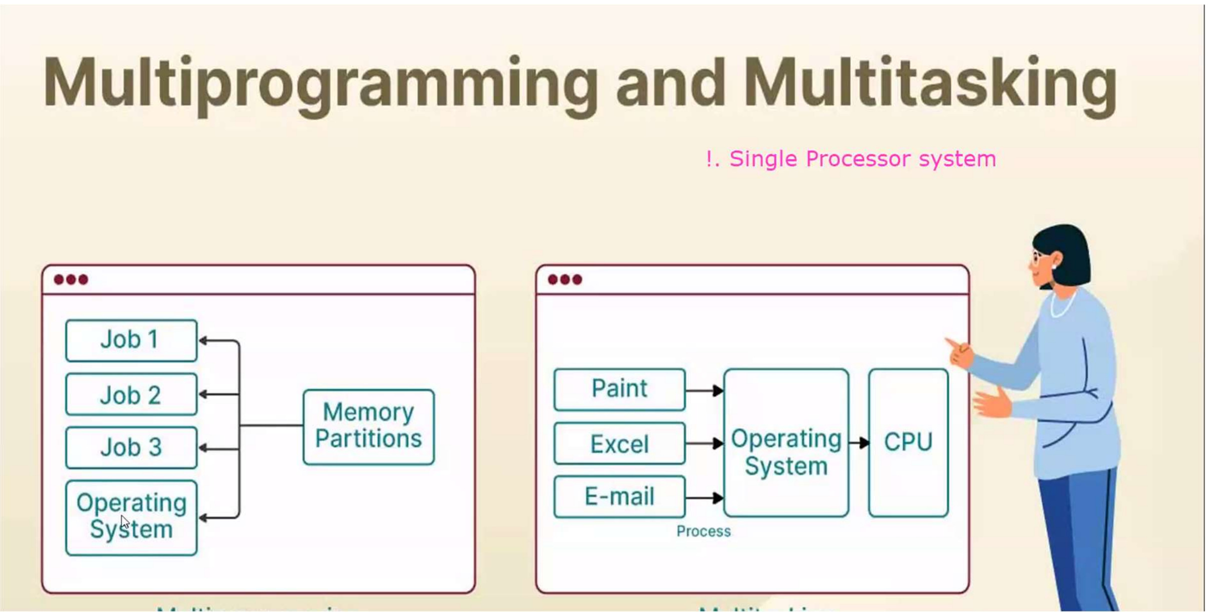
waiting

I/O event

I/O queue

I/O request

Difference between Multiprogramming and Multitasking



Multiprogramming and Multitasking

!. Single Processor system

Job 1

Job 2

Job 3

Operating System

Memory Partitions

Paint

Excel

E-mail

Process

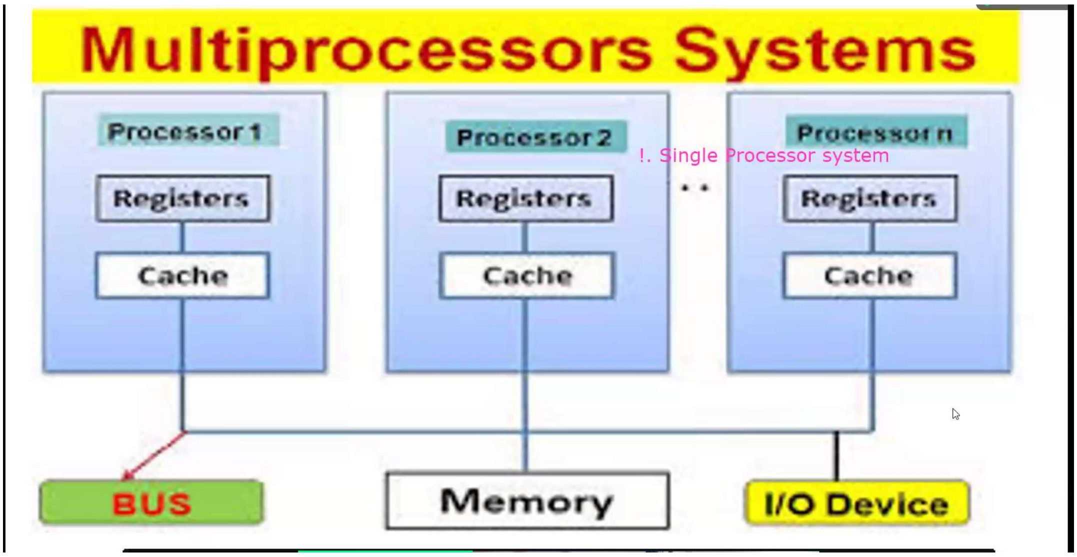
Operating System

CPU

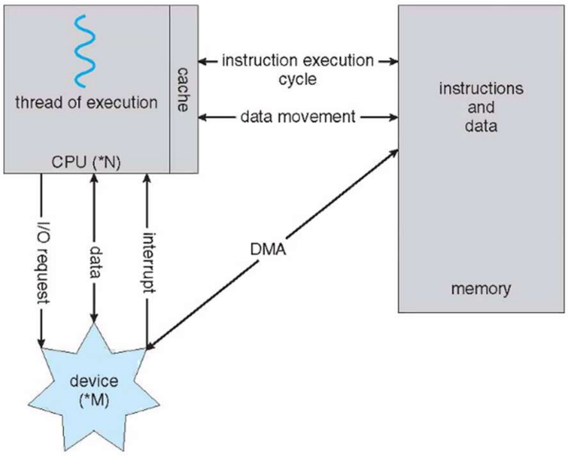
Computer-System Architecture!

Multiprocessor System –

1. Client- server architecture
2. Peer to peer architecture
3. Symmetric Multiprogramming
4. Asymmetric Multiprogramming



How a Modern Computer Works



Interrupt – It is an event usually defines as it alters the sequence of instructions executed by a processer

Halt – Stop processor

With interrupt

1. Instruction Fetch
2. Instruction execution
3. Check Interrupt

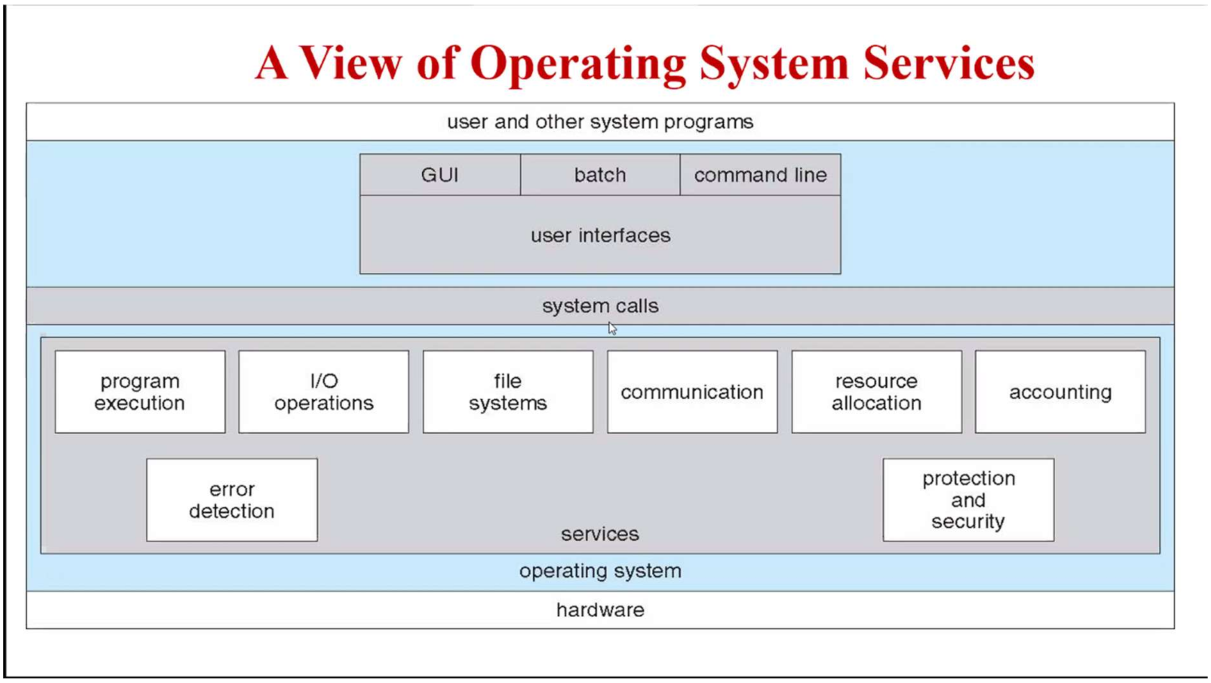
Interrupt Handling Technique

1. Polling
2. Vectored interrupted system

Kernel level interrupt –OS automatically interrupt because of priority.

User level interrupt – User it self interrupts because of priority task.

User mode – Kernal mode



A View of Operating System Services

user and other system programs

GUI

batch

command line

user interfaces

system calls

program execution

I/O operations

file systems

communication

resource allocation

accounting

error detection

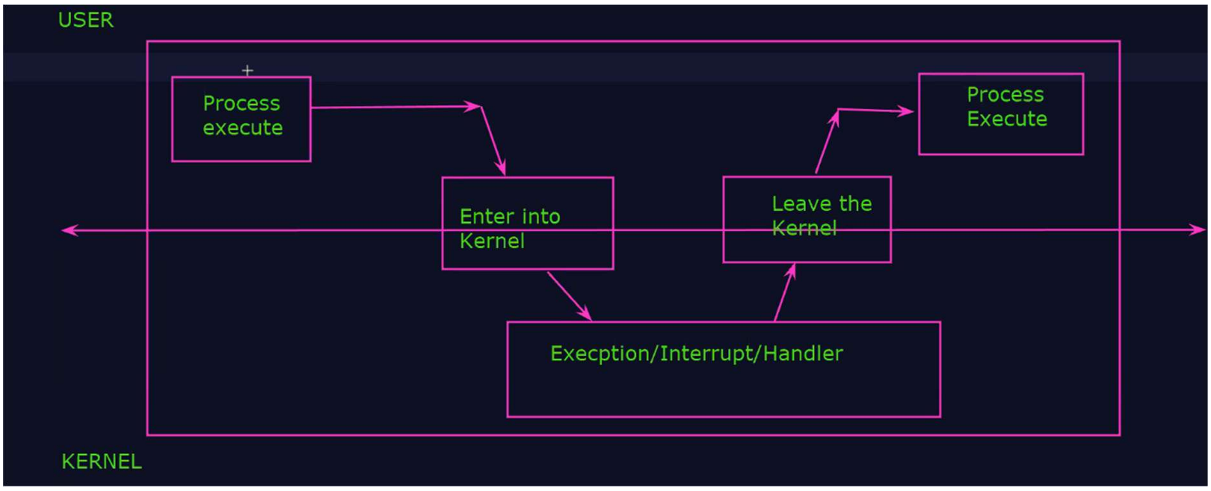
protection and security

services

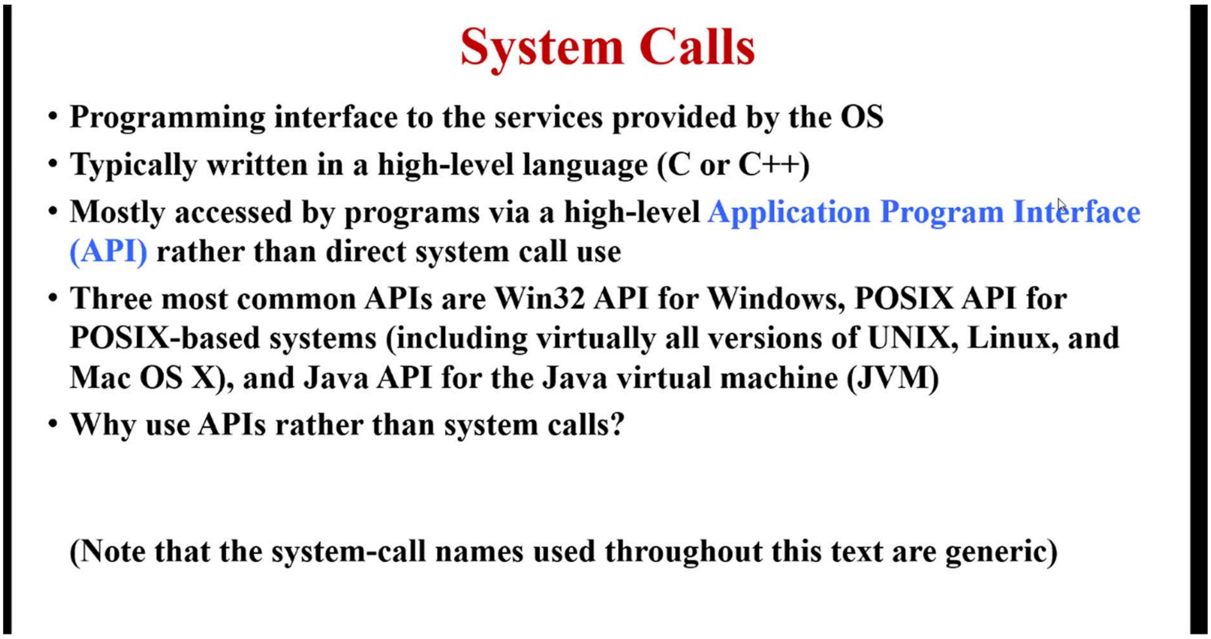
operating system

hardware

How User mode and Kernel mode works : Diagram is below.



System Calls :



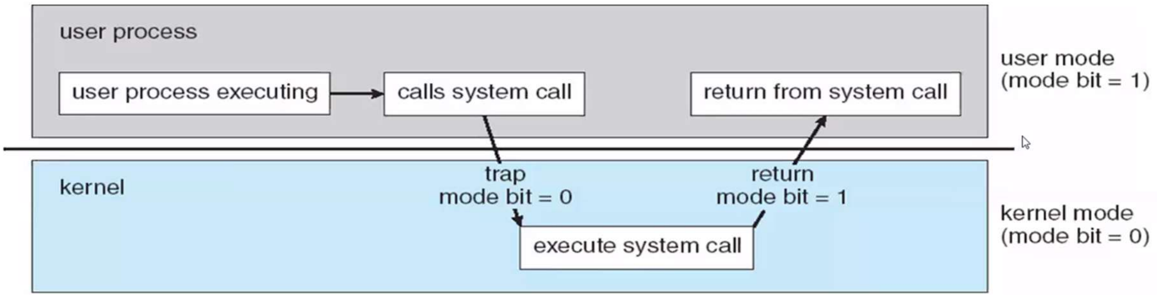
System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

(Note that the system-call names used throughout this text are generic)

Transition from User to Kernel Mode

- **Timer to prevent infinite loop / process hogging resources**
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



Day 1 LAB Session Notes

Features of all Operating System

Windows – Closed source

But more Cost

Insecure

Virus, Malware : so need Antivirus

High Hardware Cost.

Not Customizable.

Unix :

-Operating System

-1969, AT&T Bell Lab by Ken Thompson and Dennis Ritche

-Command Line Interpreter

-In was developed for the mini-computers and time shari

-UNIX was the predecessor of LINUX

Features :

-Security

-Multi-user

-Inter process comunication

-Extensive network support

-Data sent to display , files, or printer.

Linux:

Open Source

Invented by Linus Torvelds in 1991.

Linux is a variant of Unix

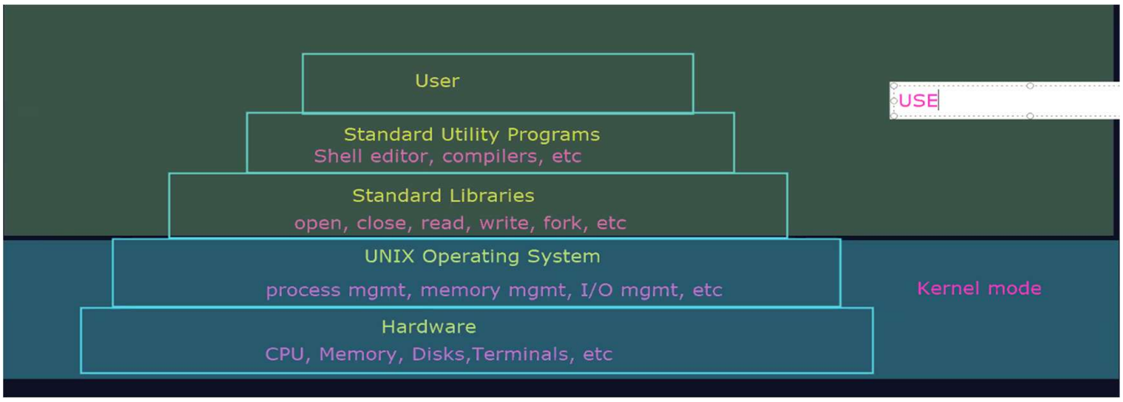
Multitasking, Multi-user, and Multiprocessor programing.

Distributions of Linux :

1. Ubuntu
2. Linux Mint
3. Redhat
4. Debian
5. Fedora
6. CentOS
7. Kali Linux

How Linux and Unix works:

Layers of linux and Unix


User
USE
Standard Utility Programs
Shell editor, compilers, etc
Standard Libraries
open, close, read, write, fork, etc
UNIX Operating System
process mgmt, memory mgmt, I/O mgmt, etc
Kernel mode
Hardware
CPU, Memory, Disks,Terminals, etc

- Kernel –

1. It is the core component of UNIX OS
2. It is responsible to execute commands
3. It is responsible to interact with hardware components.
4. It is also responsible for memory location and process allocation

- Shell –

1. It is outer layer of UNIX operating system
2. The shell is a program that sites on the interface between user and kernel.
3. It is a command interpreter and also has programming capability of it's own.

- Types of Shell –

1. Bourne Shell (Sh) – First shell by Stephen Bourne
2. C Shell (SH)
3. Korn Shell (KSH)
4. Bourne Again Shell (bash)

- Bourne Again Shell (bash) -

1. Command language interpreter
2. It is a replacement of Bourne shell (Sh)

Types of file systems in Linux –

1. Linux treats everything as a file – Including hardware devices.
2. Arranged as directory in heretical order.
3. Top level directory: Root directory (/)
4. Types of files –

1 Normal files :

These files contain data.
It can be either text file (abc.txt) or binary file (img, video)

2- Directory files :

This files represents directory
Can contain files and subdirectories

3- Device files :

In Linux every device is represented as a file.
By using these files we can communicate with that device.

The first character represents the type of file : IMP

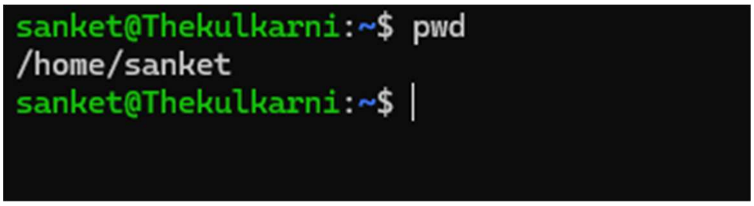
- Directory file
- Normal file
- Link file
- Character Special file
- Socket file

Common Commands :

- pwd: print working directory
- ls: List our all files and directories
- ls -A
- ls –a
- ls –r
- ls –t
- ls –F
-
- mkdir: Create directory
- cd : Change directory
- touch : To create a file

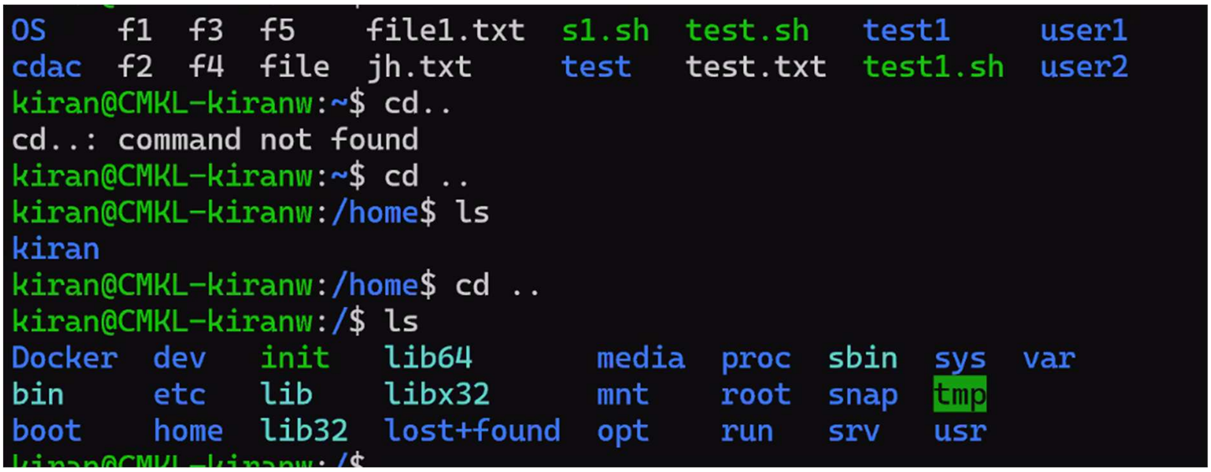
- rmdir : Remove directory
- ctrl + D for exit from txt file
-
- rm : To remove file
- cal : Display Monthly calander
- date: Display the current date and time
- help: To display list of commands
- hello : To display brief system information
- clear: To clear terminal
- exit : To logout

Practical for Day 1 :

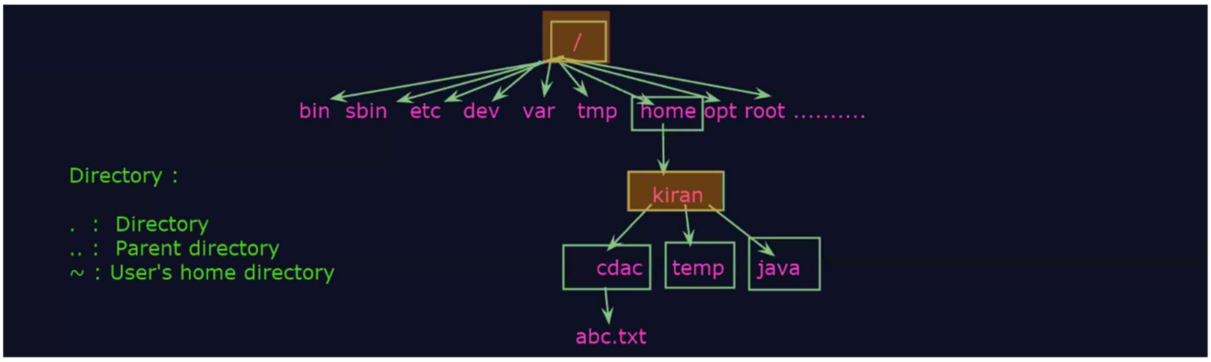


```
sanket@Thekulkarni:~$ pwd
/home/sanket
sanket@Thekulkarni:~$ |
```

Blue colour files are the Directories



```
OS    f1  f3  f5    file1.txt  s1.sh  test.sh   test1     user1
cdac  f2  f4  file  jh.txt     test   test.txt  test1.sh  user2
kiran@CMKL-kiranw:~$ cd..
cd..: command not found
kiran@CMKL-kiranw:~$ cd ..
kiran@CMKL-kiranw:/home$ ls
kiran
kiran@CMKL-kiranw:/home$ cd ..
kiran@CMKL-kiranw:/$ ls
Docker  dev  init   lib64       media  proc  sbin  sys  var
bin     etc  lib    libx32      mnt    root  snap  tmp
boot    home lib32  lost+found  opt    run   srv   usr
kiran@CMKL-kiranw:/$
```



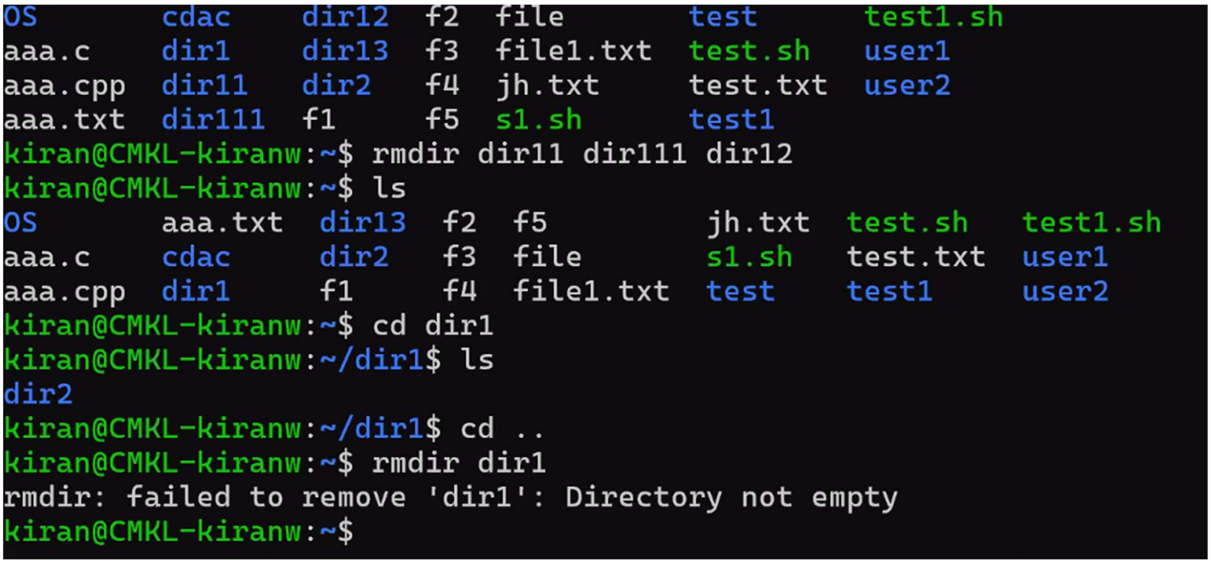
Bin – Unix utility related files saved here

Dev Device related, hardware related files saved here

Etc – login, username password will be saved here

Temp – for temporary files

Sbin - for saving device binary files



```
OS       cdac    dir12  f2  file       test      test1.sh
aaa.c    dir1    dir13  f3  file1.txt  test.sh   user1
aaa.cpp  dir11   dir2   f4  jh.txt     test.txt  user2
aaa.txt  dir111  f1     f5  s1.sh      test1
kiran@CMKL-kiranw:~$ rmdir dir11 dir111 dir12
kiran@CMKL-kiranw:~$ ls
OS       aaa.txt  dir13  f2  f5         jh.txt  test.sh   test1.sh
aaa.c    cdac     dir2   f3  file       s1.sh   test.txt  user1
aaa.cpp  dir1     f1     f4  file1.txt  test    test1     user2
kiran@CMKL-kiranw:~$ cd dir1
kiran@CMKL-kiranw:~/dir1$ ls
dir2
kiran@CMKL-kiranw:~/dir1$ cd ..
kiran@CMKL-kiranw:~$ rmdir dir1
rmdir: failed to remove 'dir1': Directory not empty
kiran@CMKL-kiranw:~$
```