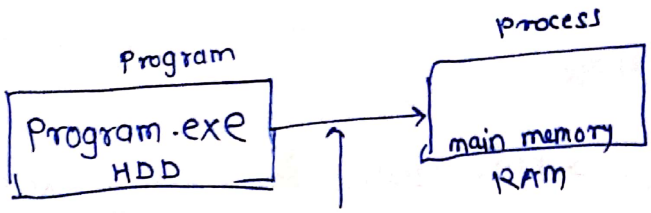
Operating System

- Interface between User & Hardware
- Fn - Resource Allocatn, Controls program, Resource manager



Job is done by LOADER (Built in program of OS)

- **Loader** - Loads program from HDD to RAM
- **Dispatcher** - It is system program (OS)
  - Loads data & instrn of program from RAM to CPU

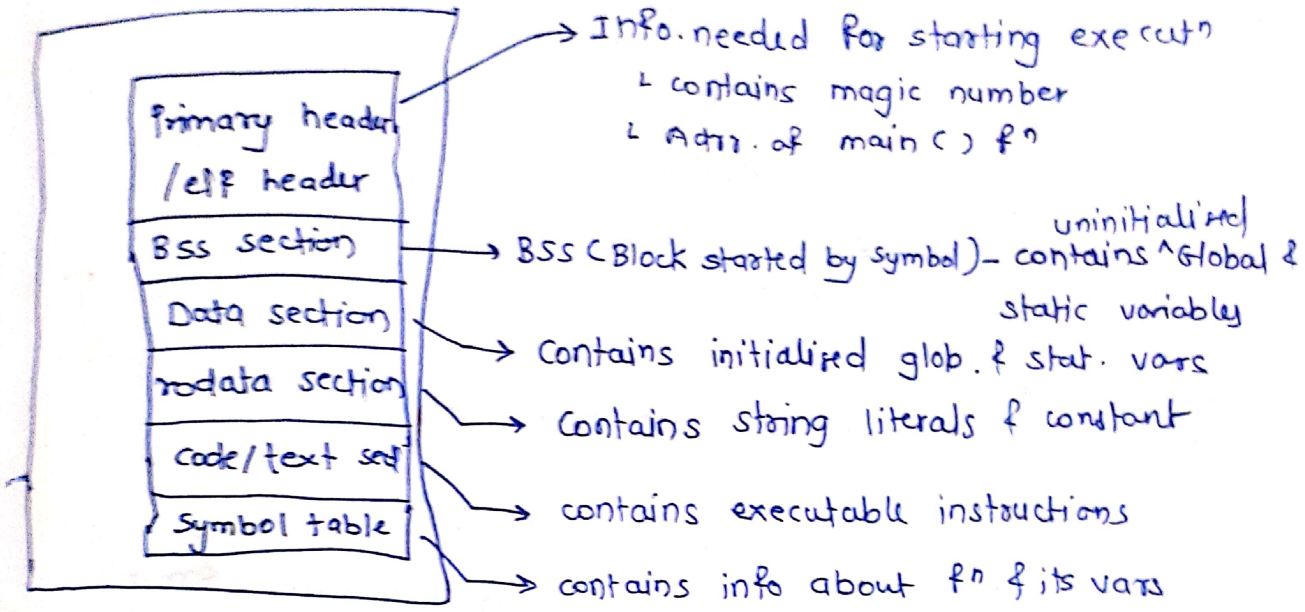
- OS controls execution of all programs & I/O devices
- C program is platform independent
- Compiled file in one m/c cant be executed in other m/c
- File format
  - linux - ELF - Executable & linkable
  - win - PE - Portable executable

- Loader - It checks file format & then checks magic number

- "magic number" - It is constant number generated by compiler i.e. file format spacific (△ OS)

- structure of an executable file

ELF file format in Linux



- OS provides two kind of interfaces for user in program forms

1. "CUI" - Command User Interface / CLI : command line inter.
   - User can enter command in text format & interact with used
   - Ex: Command prompt - "cmd.exe"
     - Linux - "shell" / "terminal"
     - MSDOS - "command.com"

2. "GUI" - Graphical user interface
   - Clicking - for minimise, rt, lt, double click
   - windows : "explorer.exe" ; Linux : Genome / KDE

- OS is simply collection of program in binary format
- 3 main components - 1. Kernel - Core program of OS, does basic minimal functionalities
  Eg: Linux - vmlinuz, win - ntoskrl.exe
  2. Utility programs
     └ Disk manager, firewall, Antivirus
  3. Applicatn softwares
     └ Apps

Fn of OS

1. Process mgt
2. memory mgt
3. CPU sched.

(Kernel Basic)

4. Hardw. Abstractn
5. File & IO mgt

Addn

6. Network
7. security
8. Interfacing

- Process to load kernel from HDD into RAM is called 'Booting'

BOOTING

└ loading of kernel from HDD to RAM by "bootstrap program"

- 2 steps of booting - 1. machine boot
  2. system boot

- devices connected through ports to motherboard are peripherals

1. BIOS - Basic ip/outp system (present inside ROM)
   ↓
   - performs POST (Power on self Test)
   ↓
   - Then BIOS executes Bootstrap loader program
   ↓
   - selects bootable device — contains bootable program in its first sector

2. System
- After selection of bootable device (Bootloader program) executes & names installed number of OS
- select any one OS
- Then Bootstrap pro. of that OS revokes
- searches for kernel
- Loads it on RAM.

[interrupt 10]

OS

1. UNIX : Uniplexed Info & Computing Services/System
   - 1st multi-user, multi-programming & multi-tasking OS
   - mother of all OSs.
   - By Ken, Denies in 1970 at AT&T bell labor.

- File - whatever that can be stored

~~Process~~

- All devices are file & divided into two categories
  1. character device - Data gets transfered character by char.
     - keyboard, mouse, printer
  2. Block device - Data get transfered block by block
     - storage devices
     - Buffer cache - To achieve max reboot in min movement

- major subsystems of OS
  - File (1)
  - Process control (2)

System calls (AKA software interupts / traps)

- Functions defined C, C++ & languages
- provides interface of services to users

- Unix - 64, Linux - 300, Win - 3000
  (To use kernel function)

- 6 categories of system calls
  1. File operatn - open, close, read, write
  2. Process control - fork, exit, wait
  3. Device control - same as file (as all device = file)
  4. Accounting info - getpid
  5. Interprocess commn - pipe, signal, kill
  6. Protn & security - chmod(), chown()

Dual mode operation

- ~~Kernal~~ mode & user mode

1) system mode (privileged mode) - CPU executes system defined code instruction

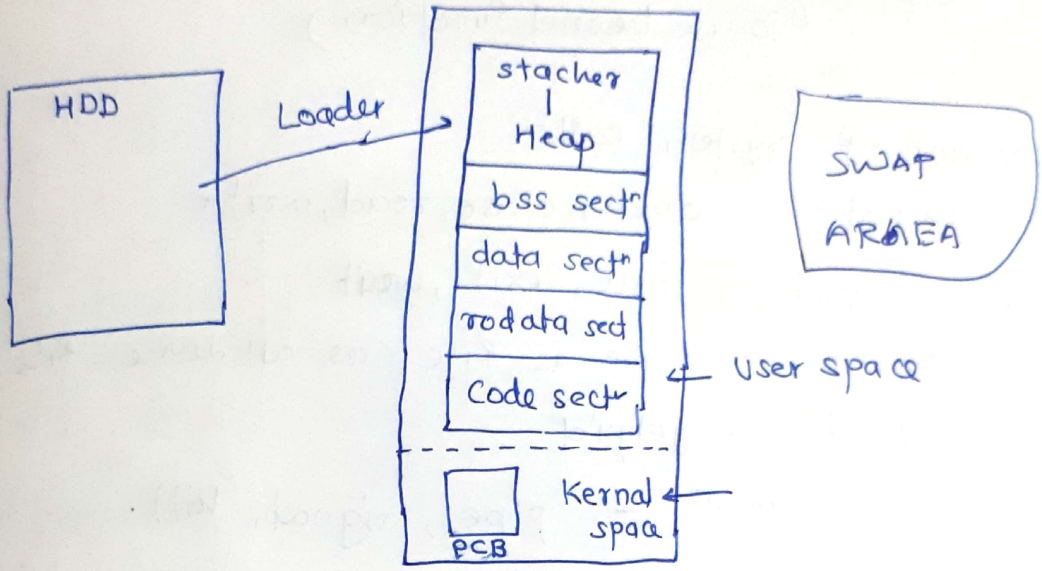
2) User mode (unpriv. mode) - CPU executes user defined code instruction

- **mode bit** (maintained by OS)
  - To identify ① / ②
  - Kernel mode - 0
  - User mode - 1

User - 1 - on
System - 0

- Program - Set of Instruction given to m/c to perform specific task

- process - Program in execution is process
  - Running program is process



- PCB → Process control Block / Process description
  - ↳ All control & execution info can be kept here
  - ↳ PCB is soul of process

- Swap Area - Portion of HDD used by OS as an extension of main memory (here inactive programs are kept for time being)

- Active running program - In PCB + User space

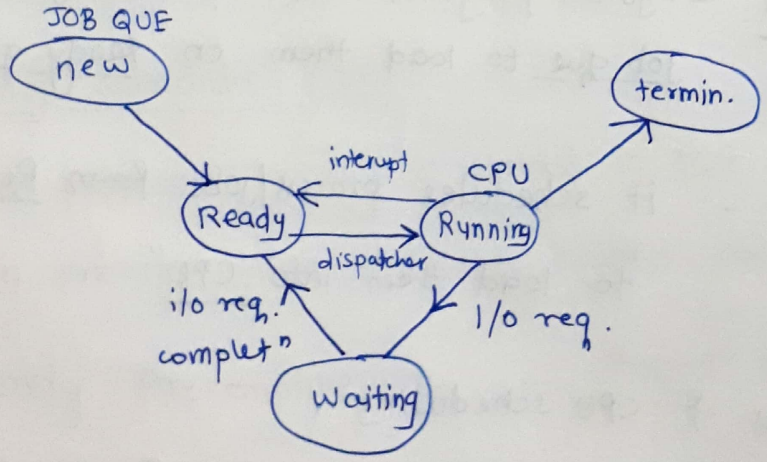
Inactive ——u—— - In Swap Area + ~~PCB~~

- Loader → loader verification (file format, magic number)
  ↳ Then execution starts
- when loader loads program into RAM then structure created inside kernel space
- For execn & control i.e. in PCB
- Per process PCB is created
- stack section → contains fn actn records of called fn
  Heap section → dynamically allocated memory

States of Process *

1. New 2. Ready 3. Running 4. Waiting 5. Terminated

1. New - Upon submission / when PCB is created
2. Ready - IF process in in RAM & waiting for CPU time
3. Running - CPU executing
4. Waiting - Requesting for i/o devices
5. Terminated - Upon exit process - PCB gets destroyed



- Kernel DS - To keep track on all running programs
  OS maintains data structures

DS (Kernal DS)

1. Job que - Contains list of PCB of all submitted process
2. Ready que - list of PCB's of processes in RAM & waiting for CPU time
3. waiting que/ device que - Waiting que for each device / list of PCB requesting for particular device

- Firstly PCB is created & adds into Job que in kernal space.

Features of OS

- multitasking & multiprogramming (time sharing) - multi-threading
  - multi-processor
- multi-user

↳ multithreading → Thread is smallest execution unit of a process
  ↳ It is lightweight process
  ↳ smallest indivisible part

- CPU can execute only thread of any one process at a time.

↳ Multi-user → multiple users can be logged in at a time

- Job scheduler (AKA Long term scheduler) - system program that schedules jobs from job que to load them on Ready que

- CPU scheduler (AKA short term scheduler) - it schedules process/jobs from Ready que to load them into CPU

  ↳ 2 types of CPU scheduling

  1. "Non-preemptive" - control of CPU released voluntarily
  2. "Preemptive" - control of CPU released by process forcefully

- Criterias to select algorithms of scheduling
  1. CPU utilisatn
  2. throughput (total WD per unit time)
  3. waiting time
  4. Response time
  5. Turn-around time
  6.

- CPU burst time - Total number of CPU cycles for execution

4 Algorithms (CPU scheduling)

1) FCFS (First come first serve)

↳ non-preemptive (convoy effect - Long processes first in que & shorter later but Turn-around time ↑)

2) SJF (shortest job first)

↳ Process with minm CPU Burst time, will get first executed

- Non-preemptive SJF → SNTF (Shortest-Next-Time-First)
- Preemptive SJF → SRTF (Shortest-Remaining-Time-First)

3) Priority scheduling (PS)

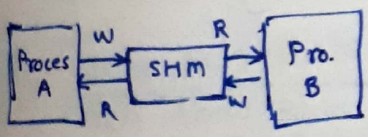
↳ Each process's priority is stored in PCB

- min. priority value indicates highest priority (1 > 2 > 3)
- Purely Pre-emptive - Starvatn in (② SJF, ③ PS)
- Ageing - Priority increament technique of Blocked process used by OS to get CPU time

4) Round robin sch. Algo

- Fixed time slice (time quantum decided in advance
- Purely preemptive
- Ensures minimum response time
- Combinatn of 4 algorithms for CPU scheduler

- Types of Process < Independent (No data sharing)
  Co-operative (Data sharing)
- Need of communicatn in 2 co-operative processes
- **Inter process Commⁿ (IPC)**
  - OS provides medium of commⁿ
  - IPC is service by the kernel for the processes commⁿ
  - 2 IPC models / method
    - ① **shared memory model** → 2 processes communicates via Reading & writing data into shared memory region i.e. provided by OS



    - ② **Message passing model** - By sending messages
      - message que system (By OS)
      - shared memory model is faster
      - Types - 1. "pipe" - Pipe mechanism - Unidirectⁿ commⁿ
        - 1 way pathway
      - 2. "message que" - one process can send as well recieve message packet

↳ one process sends signal to another process but

↳ OS can send signal to an process but process can send signal to OS.

↳ SIGTERM, SIGKILL

4. Socket

- Process on one m/c communicate with process running on another m/c
- Socket = IP adress + Port Number

- Race Condn - 2 or more processes are trying to access same resource at same time

↳ so problem of "data inconsistency" - critical sectn problem

↳ so need of process synchronizatn/co-ordination

- can be avoided by OS

↳ ① Deciding on order of Allocatn of resources

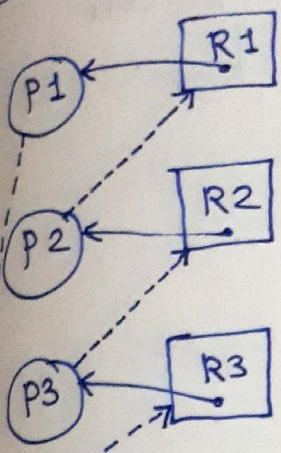
② Last process changes prevail

[Synchronisation Tools] (for process synchronisation)

1. Semaphore

2. Mutex object

Deadlock



Process	Resource Assigned	Resource request
P1	R1	R3
P2	R2	R1
P3	R3	R2

- No execution will continue further

- Four necessary & sufficient condition to occure deadlock

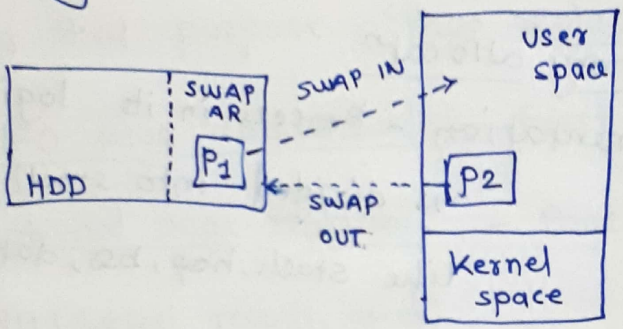
1. mutual exclusion - Resource at a time can be only acquired by one process.
2. No preemption - control of resource cant be taken away forcefully
3. Hold & wait - Hold r. & waiting for next R.
4. Circular wait

- Deadlock handling methods

1. Deadlock prevention - By discarding any 1/4 condn ↑
2. Deadlock detection & Avoidance
   - 2 Algorithms - 1) Resource allocatn graph algo.
     2) Bankers algo.
3. Deadlock recovery - 1) Process terminatn
   2) Resoure preemptn

Memory Management

- Execution of every program occurs on RAM (main memory)
- Swap area - Area in HDD which is used by RAM as a extension of main Area. To keep inactive ~~process~~ running programs are kept temporarily.
- Processes keep swapping (swap in & swap out)
- "Memory manager" - Performs swapping operations



- conventionally size of swap area = 2 × RAM memory
- Two ways of memory allocation / two methods

1) Contiguos memory allocation

- process executed when memory allocated in continguous manner

1.1) Contigous memory allocation - Fixed size partitioning

- User space of RAM divided into fixed partitions (same size)
- Process allocated free partition

1.2) Variable size partitioning - Dynamic partitions

- Problem of Fragmentation occurs in dynamic partitioning

- External Fragmentn - memory gets divided into used & free partitions,

  ↳ memory is availble but not in contigues manner

Two solutions

① Compaction

↓

shifting all used & free partitions diff. sidy. (not practical soln)

② Non-contangous memory allocatn

② Non-cont. memory allocatn

methods - i) Segmentation - Process in its logical memory is divided into small size segments like stack, heap, bss, data, rodata

ii) Paging - RAM divided into fix size of partition called as Frames

---

Virtual memory management

↳ manages main memory + swap Area

↳ So processes of size more than RAM can be dealth with

- Availabl. 4GB RAM + 8GB swap

- V.M. = Paging + swaping