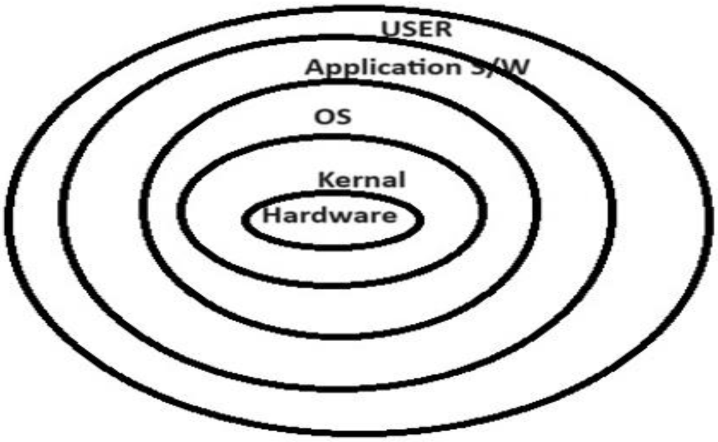
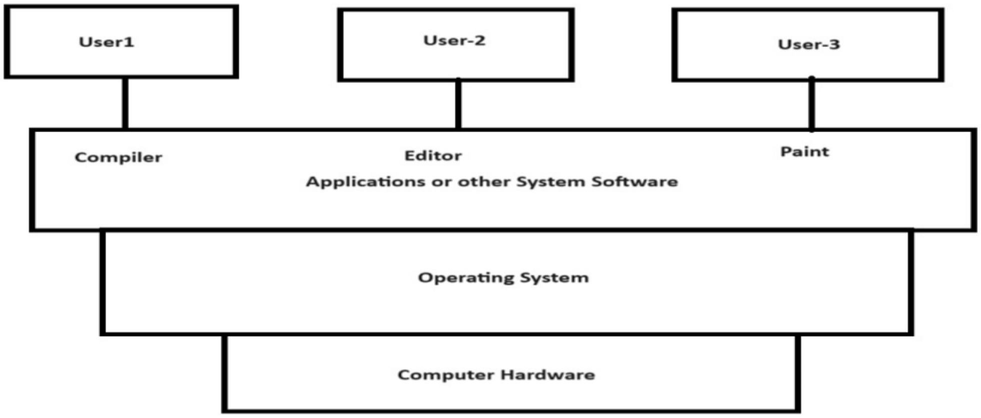
OS Notes Day-1 Date: 27-08-2024

Session-1: Introduction to Operating System

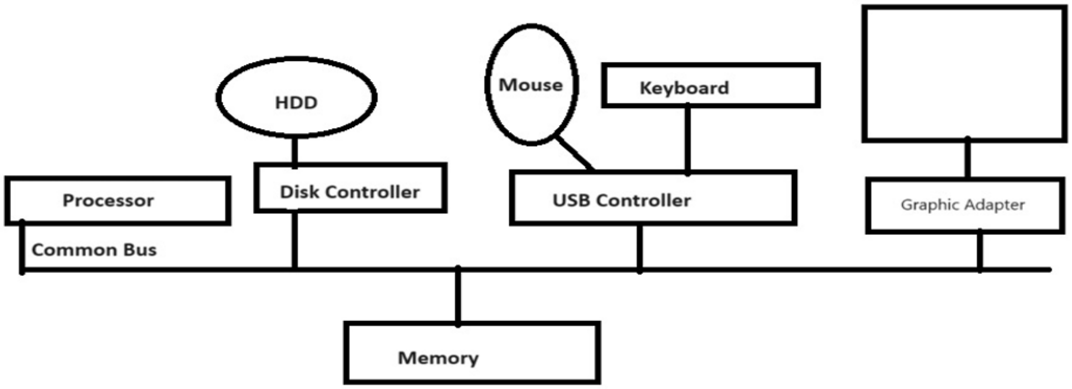
- What is OS?
  - Hardware Manager: It manage all the hardware resources or components of computer.
  - Process Manager: It supervise all the task/process/job which is being executed by processor.
- How is it different from other application software?
  - OS is installed over hard-drive.
  - Applications are also installed over hard-drive but under the layer of OS.
  - OS runs over computer system, and Applications runs over OS.
- Why is it hardware dependent?



- Different components of OS



- Basic computer organization required for OS.



- Examples of well-known OS

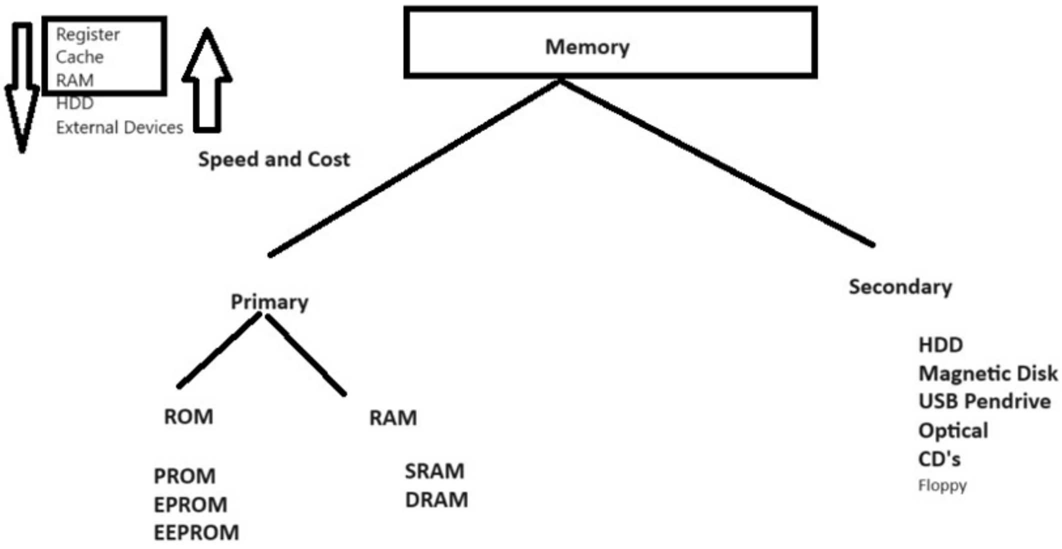
1. Mobile OS: Android, iOS, Windows
2. Embedded System OS:
3. Real Time OS: HRT, SRT
4. Desktop OS: Personal Computer
5. Server machine OS etc.

- How are these different from each other and why?

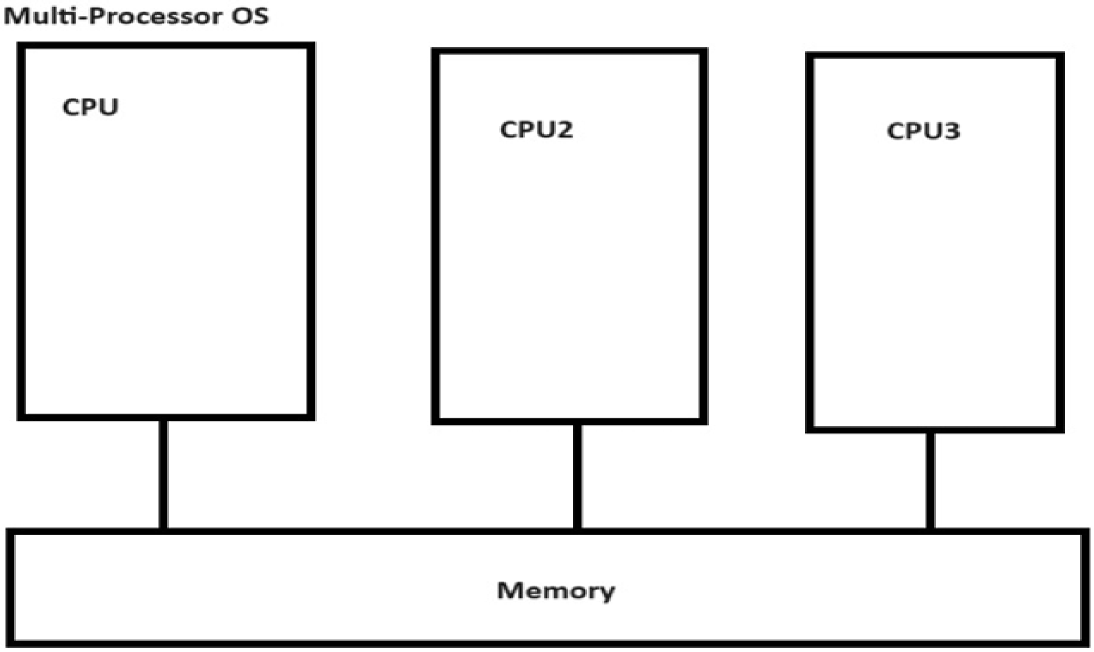
- Functions of OS
  - Process Management (Process Scheduling Algo)
  - Memory Management
  - Device Management (HDD, Printer, Monitor, Spealer, WebCam)
  - Disk Management (Disk Scheduling Algos)
  - Network Management (Network Card / Controller)
  - File Management
  - Security Management (Firewall, Anti Virus, Anti Spyware)
- User and Kernel space and mode;
- Interrupts and system calls
- Memory Hierarchy in Computer System

1. Primary Memory: RAM, ROM (PROM, EPROM, EEPROM): BIOS/Firmware/UEFI

2. Secondary Memory:HDD, SSD, Magnetic Tape, Pendrive , External HDD, CDs, Flopy



- ➢ Types of Operating System
  - Batch Operating System
  - Multi-Programming OS
- ➢ Multiprocessor OS



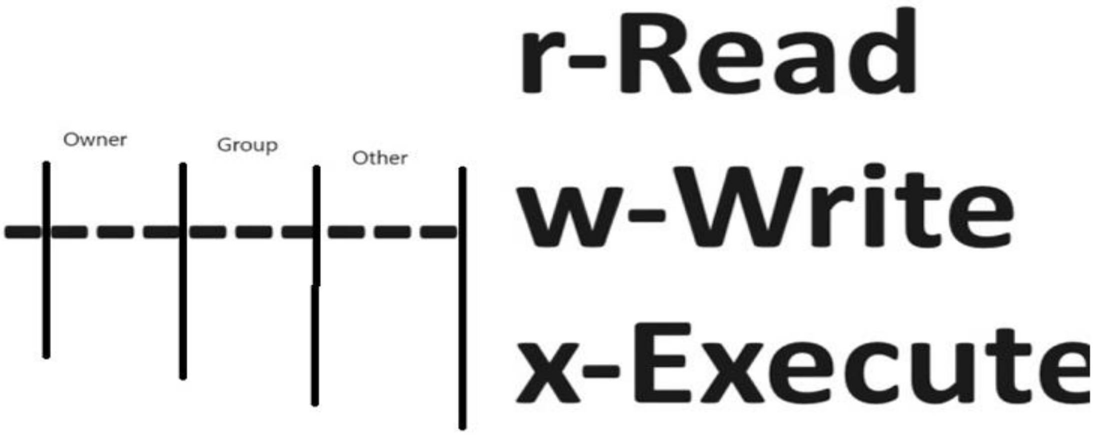
- Distributed OS
- Desktop OS
- Server OS

Session-2: Introduction to Linux

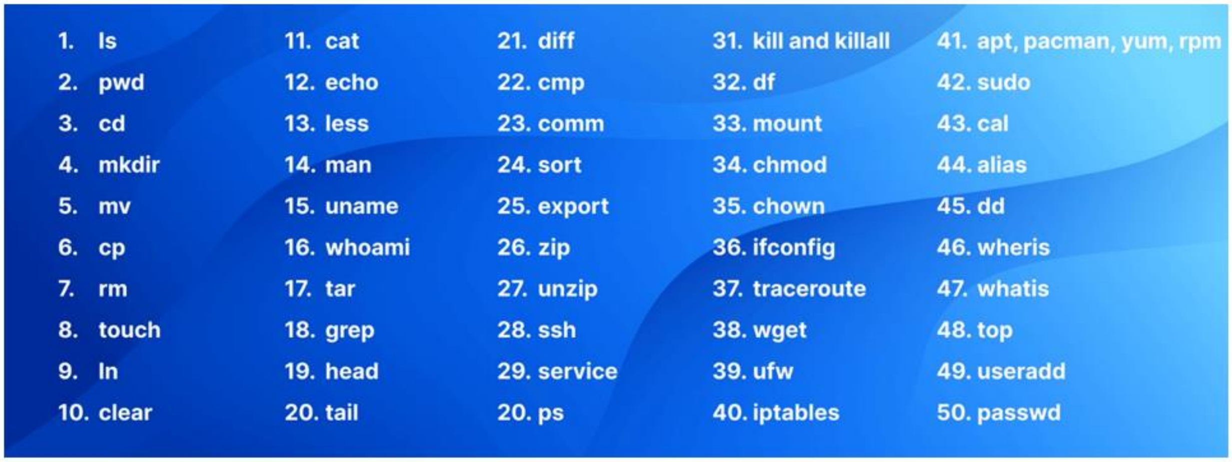
- It is an Open Source operating system. It is available free to use and user can modify it according their need.
- The founder or linux is Linun Torvards. It available since 1991.
- An Open Source Community is woking behind the updation and upgradation of the linux code.
- Feature
    1. No Cost / Low Cost
    2. Multi-Tasking
    3. Security
    4. Multi-User
    5. Stable and Scalable
    6. Networking
    7. CLI as well as GUI
    8. Better File System
    - Working with basics file system of Linux
    - / is root directory

    1. /bin: User Bineries
    2. /sbin: System Bineries
    3. /etc: Configuration Files
    4. /dev: Device Files
    5. /proc: Process Information
    6. /var: Variables Files
    7. /tmp: Temporary Files
    8. /usr: User Programs
    9. /home: Parent directory of user friendly directory
    10. /boot: Boot Loader Files
    11. /opt:Apps
    12. /lib: System Libraries
- Commands associated with files/directories
    1. pwd: Present Working Directory
    2. ls: it list out all the files and directory of current working directory
    3. nano: it actually run the nano editor and open the specified file.
    4. touch: It is used to create a new file
    5. mkdir: To create a new directory.
    6. chmod: to give and revoke the file or directory permissions
    7. rm: to remove file and recursive directory
    8. rmdir: to remove a prticuler directory
    9. cd: to change directory
- Other basic commands.
- **Ref: https://ubuntu.com/tutorials/command-line-for-beginners#1-overview**
- Operators like Redirection (>), Pipe (|)

- What are file permissions and how to set them?



- Permissions (chmod, chown, etc)
- Ref: https://help.ubuntu.com/community/FilePermissions
- chown: to change owner of the file
- su: to switch the user from current to any specified user
- cat: to display content of the file on console
- head: to display top n lines of the file on console. By default it will print first 10 lines
- tail: to display bottom n lines of the file on console. By default it will print last 10 lines
- adduser: to add new user into the system
- sudo: to give some specific privilages to the user's other than root.
- Operators like Redirection (>)
- Other basics command



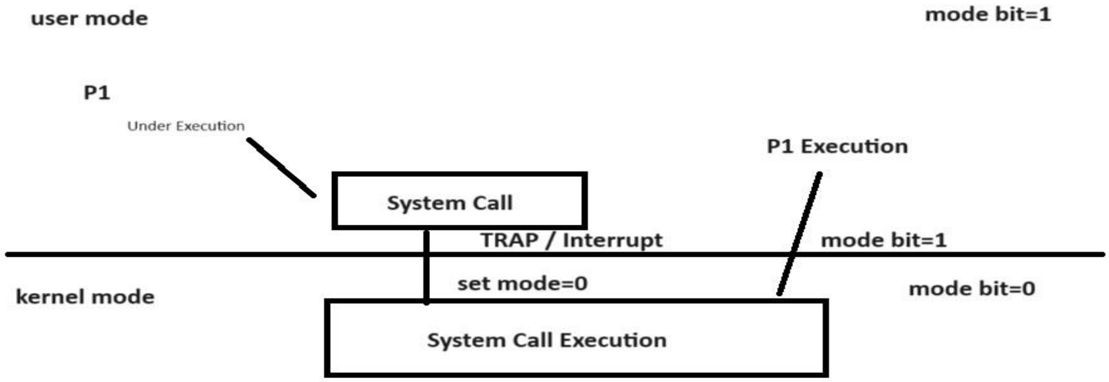
1. ls
2. pwd
3. cd
4. mkdir
5. mv
6. cp
7. rm
8. touch
9. ln
10. clear
11. cat
12. echo
13. less
14. man
15. uname
16. whoami
17. tar
18. grep
19. head
20. tail
21. diff
22. cmp
23. comm
24. sort
25. export
26. zip
27. unzip
28. ssh
29. service
20. ps
31. kill and killall
32. df
33. mount
34. chmod
35. chown
36. ifconfig
37. traceroute
38. wget
39. ufw
40. iptables
41. apt, pacman, yum, rpm
42. sudo
43. cal
44. alias
45. dd
46. wheris
47. whatis
48. top
49. useradd
50. passwd

- **Process Management**
  - User Mode and Kernal Mode (System calls)
  - System Calls types

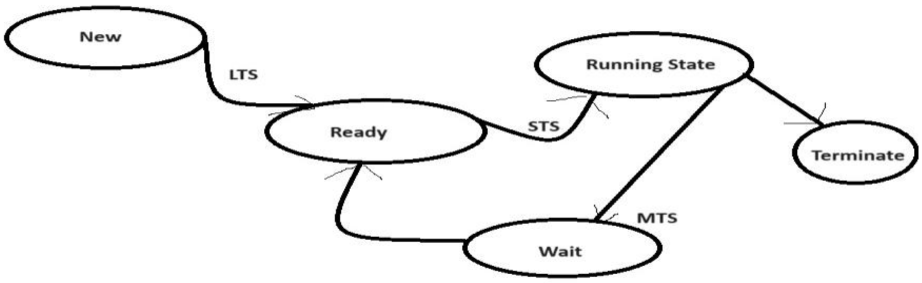
6. File releated calls: Read(), Write(), Delete(), Open(), Close(), Create()

7. Process related calls: New(), Fork(), Exit(), Wait(), Running(), etc.

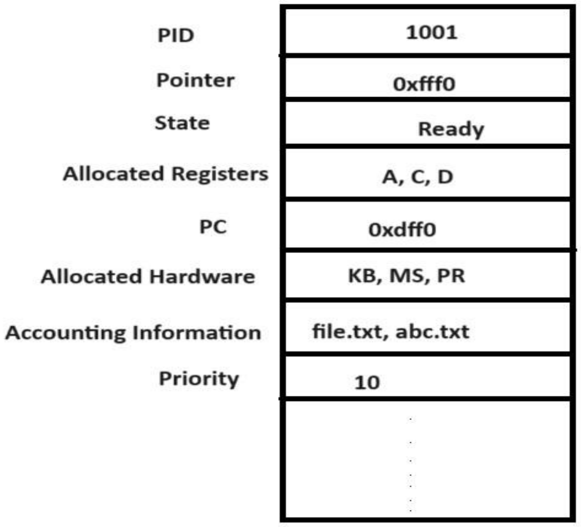
8. Device related calls: Read(), ioctl etc.
9. Information related: getpid, gettime, sysdata, etc.
10. Communucation related: wait(), signal(), status, etc.



- Process: It is information or code or data which help the processor to execute or complete the user task.
    1. Code Segment: It consist of compiled code or instructions to be executed
    2. Data Segment: It consist of data required for the execution.
    3. Information Segment: It have metadata about the system variables or system which help in the process execution.
    Memory Segment: Heap and Stack
- Process Life Cycle
    1. New State
    2. Ready State (Ready Queue)
    3. Running State- Processor is executing the process
    4. Terminated State
    5. Wait State



- Process Schedulers
  1. Long Term Schedulers (LTS)
  2. Short Term Schedulers
  3. Medium Term Schedulers
- Process Control Block (PCB) (Kindly read all attributes of PCB)



PID	1001
Pointer	0xfff0
State	Ready
Allocated Registers	A, C, D
PC	0xdff0
Allocated Hardware	KB, MS, PR
Accounting Information	file.txt, abc.txt
Priority	10

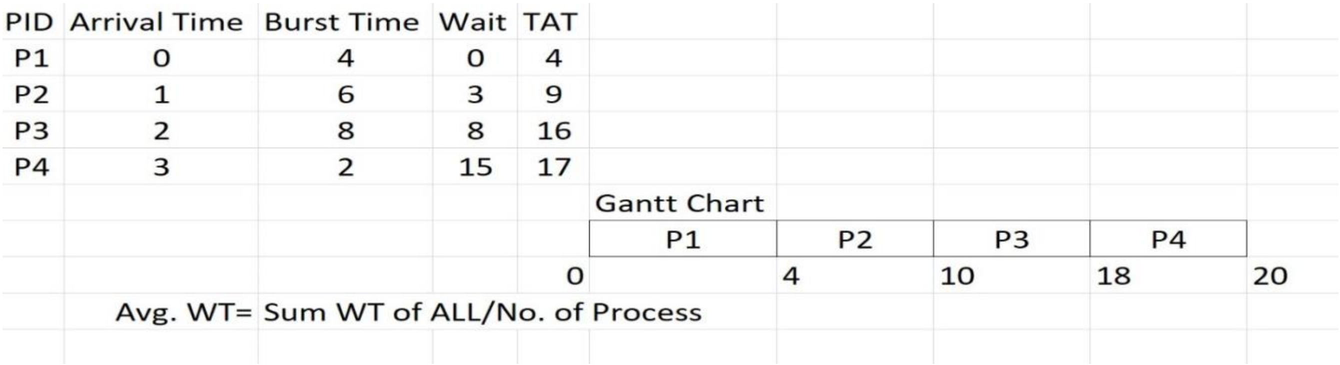
PCB P1

- Scheduling Algorithms
  1. Pre-Emptive Scheduling
  2. Non Pre-Emptive Scheduling

1. FCFS (First Cum First Serve)

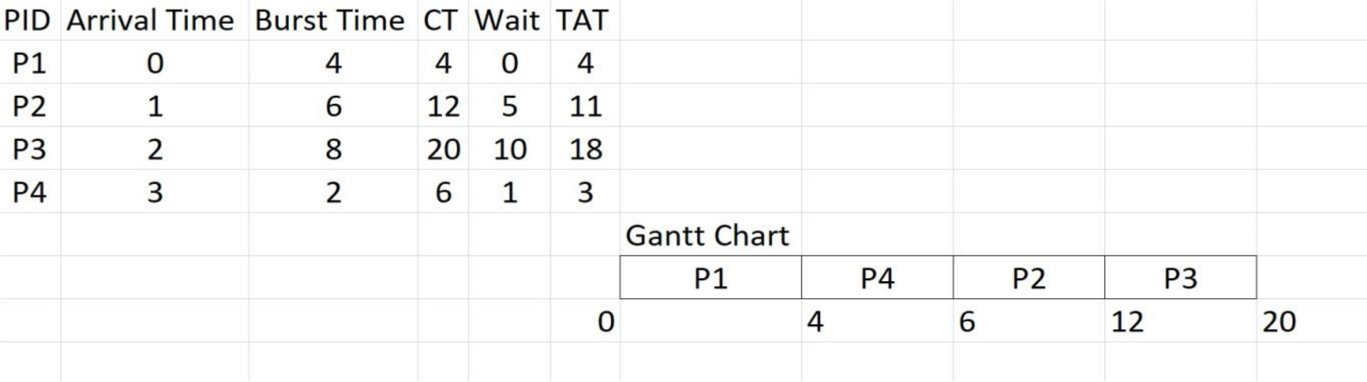
   - Waiting Time of Process= CPU Allocation-Arrival Time

   - Avg Waiting Time = Sum WT of All process / no. of processes



PID	Arrival Time	Burst Time	Wait	TAT					
P1	0	4	0	4					
P2	1	6	3	9					
P3	2	8	8	16					
P4	3	2	15	17					
					Gantt Chart				
					P1	P2	P3	P4	
				0		4	10	18	20
	Avg. WT=	Sum WT of ALL/No. of Process							

2. SJF (Shortest Job First)

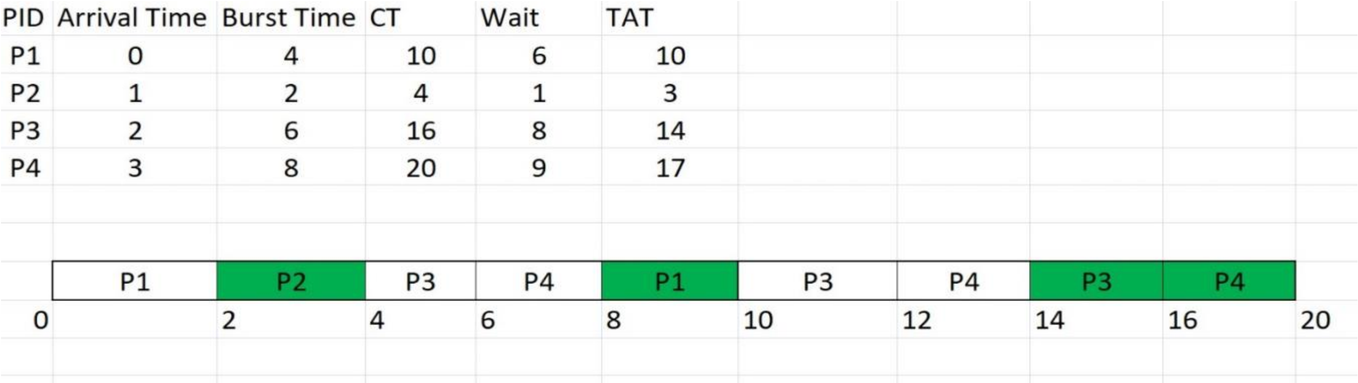


PID	Arrival Time	Burst Time	CT	Wait	TAT
P1	0	4	4	0	4
P2	1	6	12	5	11
P3	2	8	20	10	18
P4	3	2	6	1	3

Gantt Chart

	P1	P4	P2	P3	
0		4	6	12	20

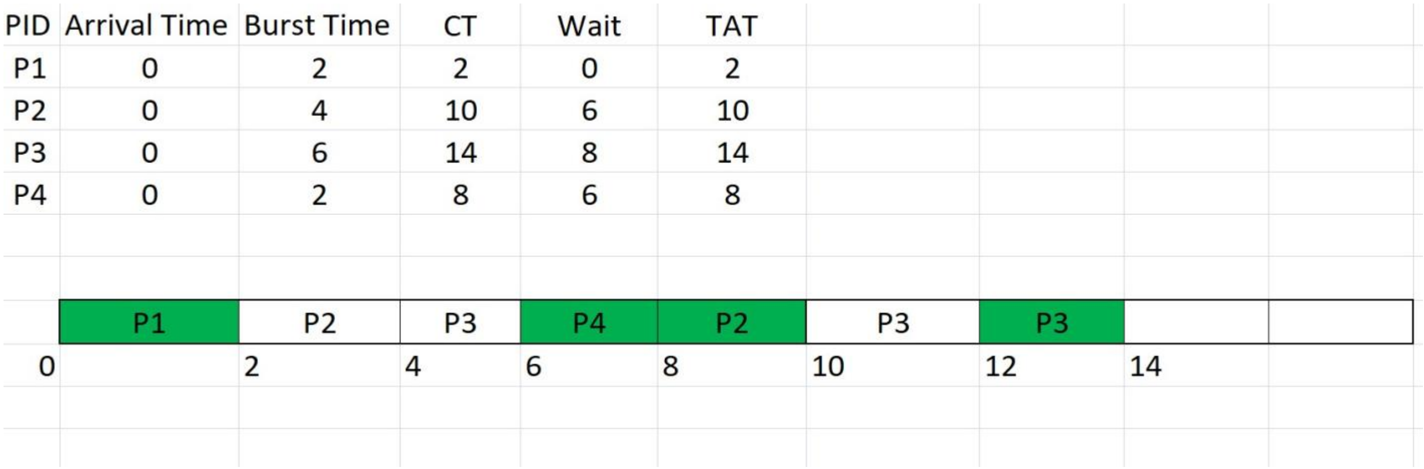
3. Round-Robin
    - In round robin algo a fixed slice of time is given to each an every process. That slice of time is know as quantum.



PID	Arrival Time	Burst Time	CT	Wait	TAT
P1	0	4	10	6	10
P2	1	2	4	1	3
P3	2	6	16	8	14
P4	3	8	20	9	17

	P1	P2	P3	P4	P1	P3	P4	P3	P4	
0	2	4	6	8	10	12	14	16	20	

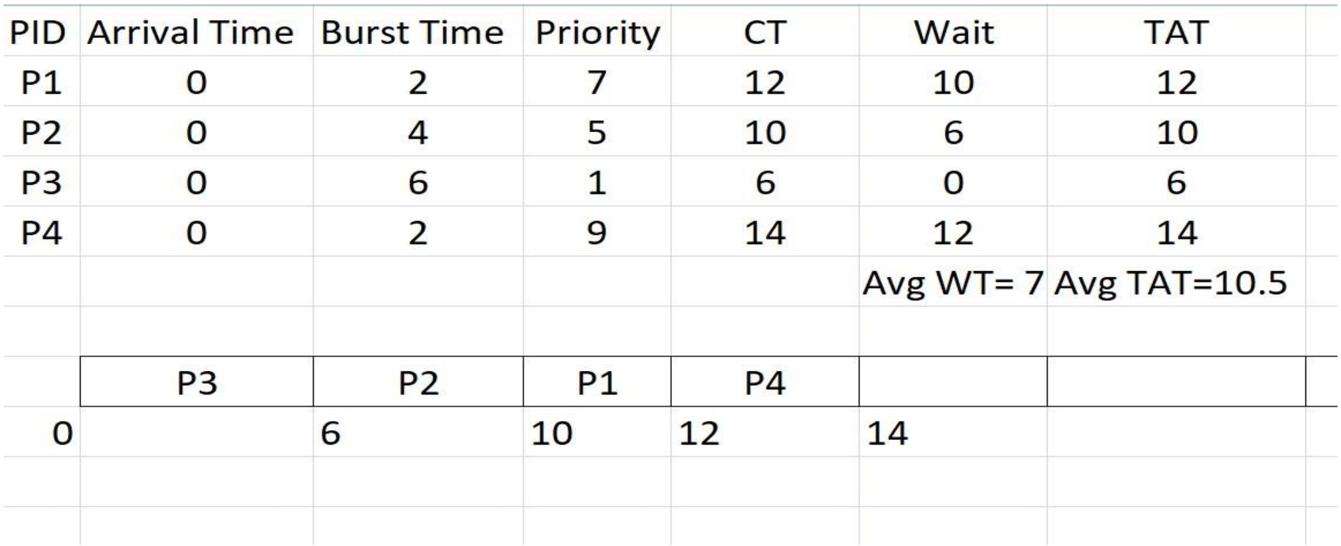
    - RR with AT 0



PID	Arrival Time	Burst Time	CT	Wait	TAT
P1	0	2	2	0	2
P2	0	4	10	6	10
P3	0	6	14	8	14
P4	0	2	8	6	8

	P1	P2	P3	P4	P2	P3	P3		
0		2	4	6	8	10	12	14	

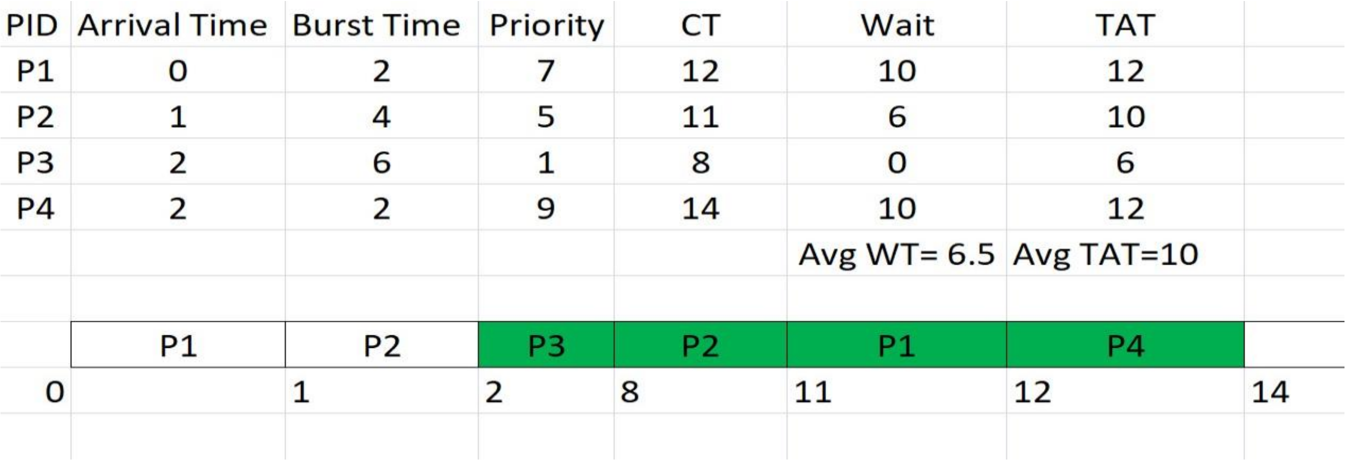
4. Priority



PID	Arrival Time	Burst Time	Priority	CT	Wait	TAT
P1	0	2	7	12	10	12
P2	0	4	5	10	6	10
P3	0	6	1	6	0	6
P4	0	2	9	14	12	14
					Avg WT= 7	Avg TAT=10.5

	P3	P2	P1	P4		
0		6	10	12	14	

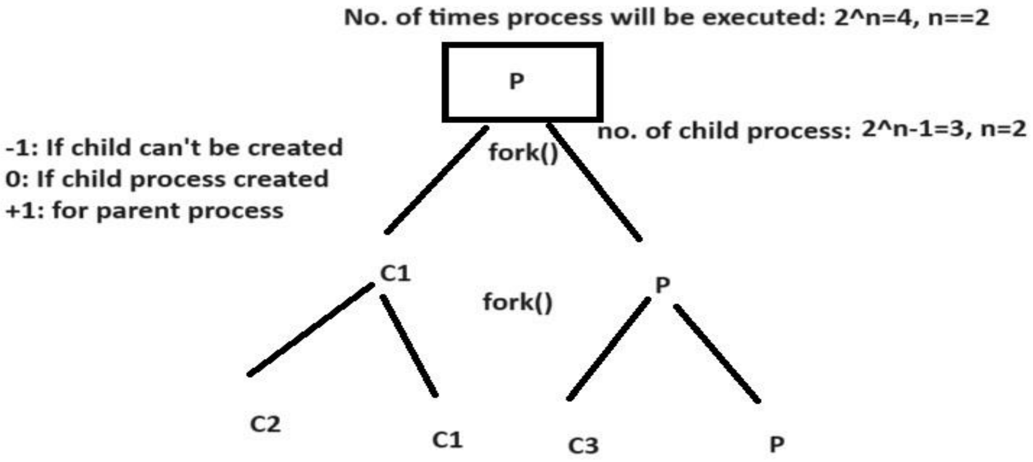
- PR with some AT



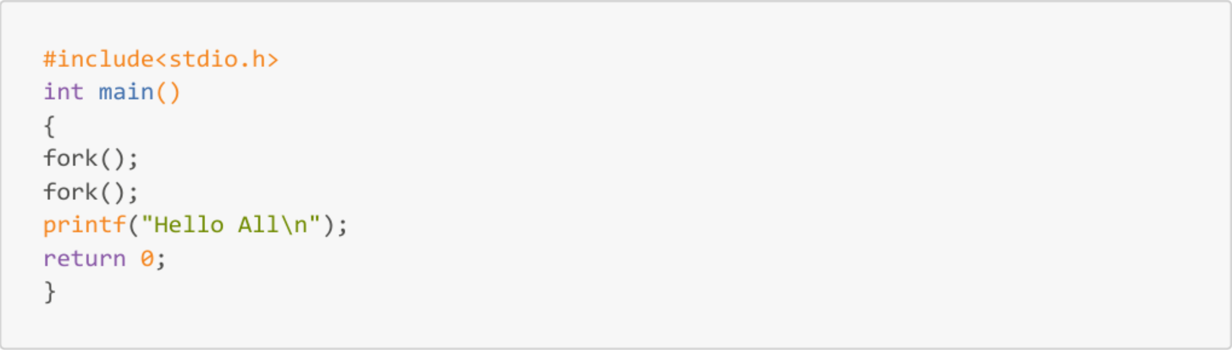
PID	Arrival Time	Burst Time	Priority	CT	Wait	TAT	
P1	0	2	7	12	10	12	
P2	1	4	5	11	6	10	
P3	2	6	1	8	0	6	
P4	2	2	9	14	10	12	
					Avg WT= 6.5	Avg TAT=10	
	P1	P2	P3	P2	P1	P4	
0		1	2	8	11	12	14

- **Process creation using fork**
  - fork(): It is method / system call which is used to create child process of the current process being executed.
  - What a child process does: It does exactly the same for which you have design the parent process.

- waitpid and exec system calls (fork-exec-wait)


No. of times process will be executed: 2^n=4, n==2
P
no. of child process: 2^n-1=3, n=2
fork()
-1: If child can't be created
0: If child process created
+1: for parent process
C1
fork()
P
C2
C1
C3
P

- Examples on process creation

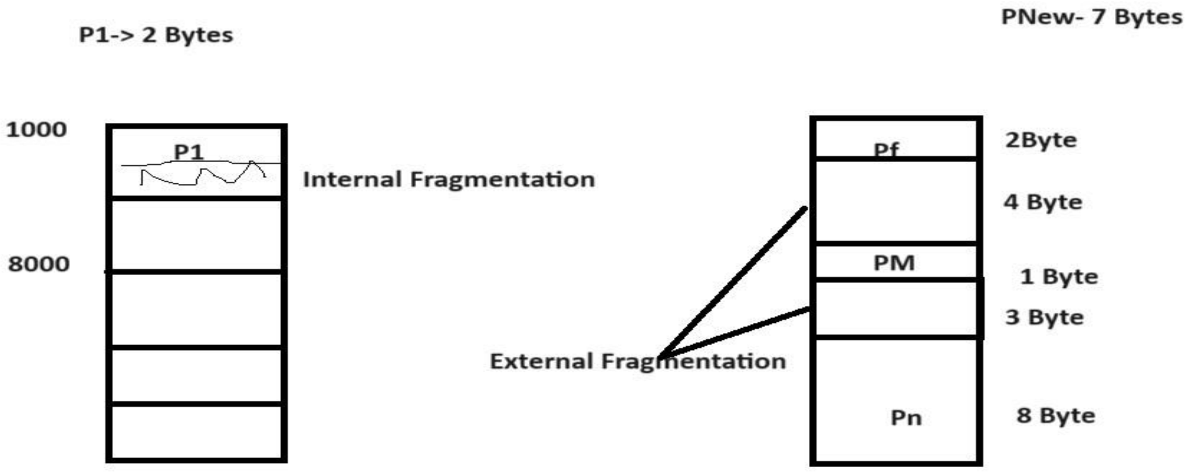


```c
#include<stdio.h>
int main()
{
fork();
fork();
printf("Hello All\n");
return 0;
}
```

- Parent and child processes
- Orphan and zombie processes

Memory Management (RAM)

- The memory of computer systems can be divided into block of fized size or variable size.
- This process is known as Fixed Size Partioning and Varibale Size Partioning respectively.
- Fixed Size Partioning: Here the memory is divied into fixed size blocks where all the blocks are of size like either 2 Bytes, 4 Bytes , etc.
    - The Process which will get the memroy may be small or equal to the partitioning block size.
    - As the process size may be smaller than block, it give rise to Internal Fragmentation
- Variable Size Partioning: Here the memory is divided into variable size blocks in which blocks may have size 2 Bytes, 3 Bytes or any size.
    - In varibale size partioning during contigous memory allocation some time total memory available can not given to the new process which leads to External Fragmentation.



Fixed Size Partioning

Variable Size Partitioning

Compaction: Process of moving empty space of memory to one side of the memory to get large block of the free memory.