Operating Systems

Quiz

Q. An OS is a __________

A. system software
B. resource manager
C. resource allocator
D. all of the above

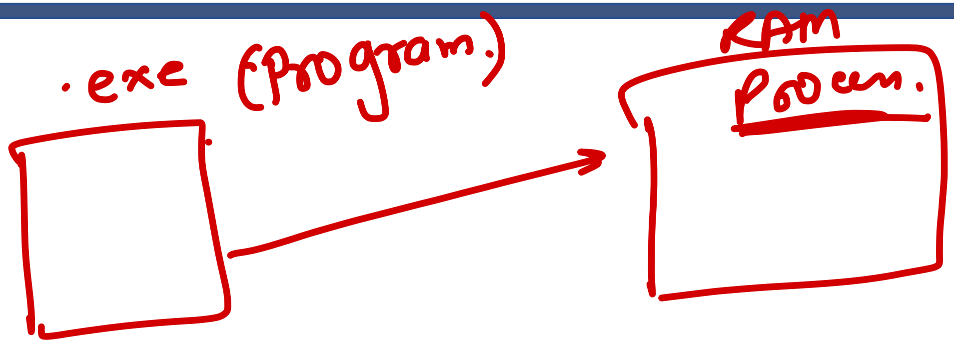
Q. Which of the following is a system program?

A. Compiler
B. Linker
C. loader
D. Assembler
E. all of the above
F. none of the above

Quiz

- Which of the following is a process?

A. program.i
B. program.o
C. program.s
D. program.out
E. None of the above
F. All of the above



- Which of the following is a program?

A. program.i
B. program.o
C. program.s
D. program.out .exe .
E. None of the above
F. All of the above

Quiz

Q. Which of the following program provides graphical user interface in Windows Operating System?

a. cmd.exe → CUI
b. explorer.exe
c. command.com → MSDOS - CUI

GUI → Window → explorer.exe

Linux → GNOME/KDE

d. all of the above
e. none of the above

Functions of an OS:

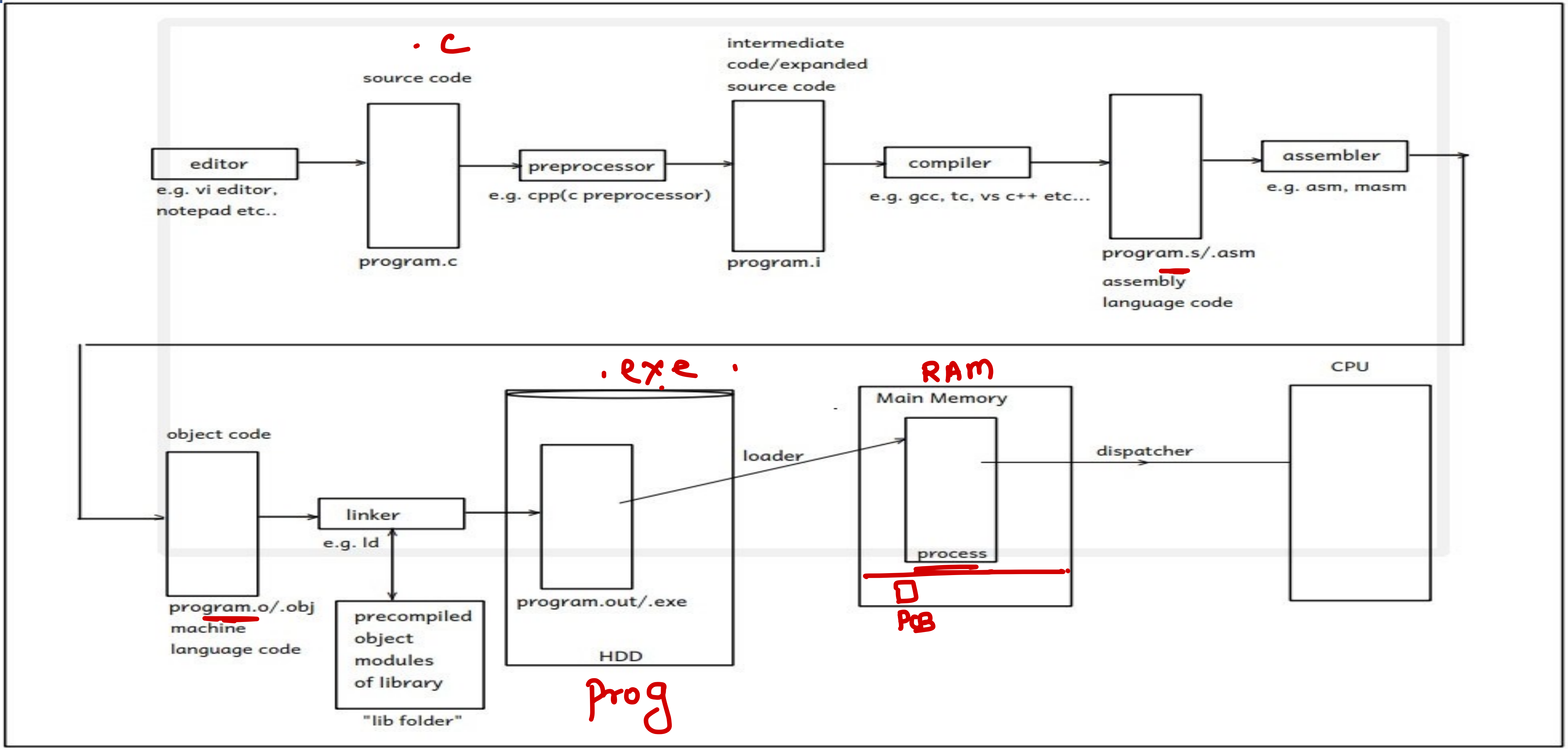
Basic minimal functionalities/Kernel functionalities:

1. Process Management
2. Memory Management
3. Hardware Abstraction
4. CPU Scheduling
5. File & IO Management

Extra utility functionalities/optional:

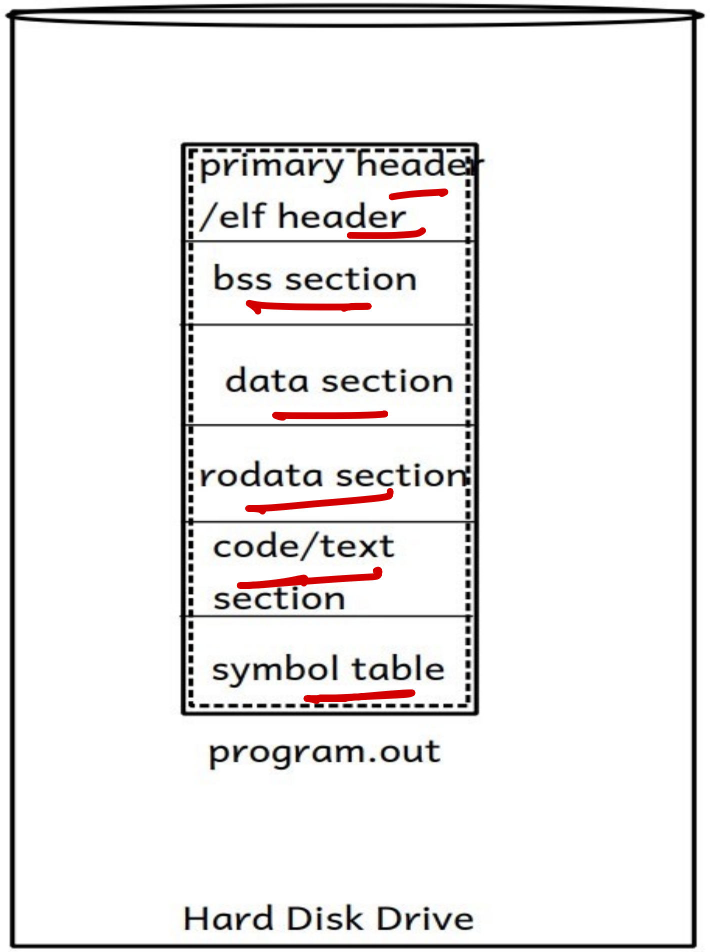
6. Protection & Security
7. User Interfacing
8. Networking

Operating System Concepts



Structure of an executable file

ELF file format in Linux



1. primary header/exe header: it contains information which is required to starts an execution of the program.

e.g. - addr of an entry point function --> main() function

- **magic number:** it is constant number generated by the compiler which is file format specific.
  - magic number in Linux starts with ELF in its eq **hexadecimal format**.
  - info about remaining sections.

2. bss(block started by symbol) section: it contains uninitialized global & static vars

3. data section: it contains initialized global & static vars

4. rodata (readonly data) section: it contains string literals and constants.

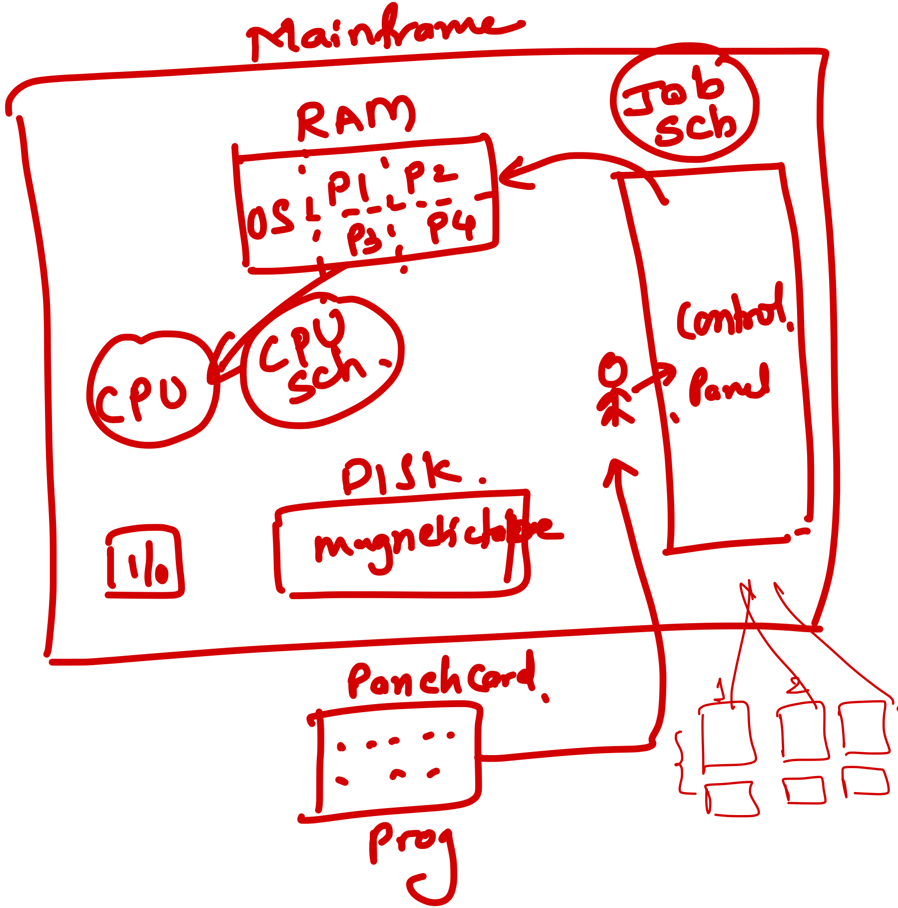
5. code/text section: it contains an executable instructions

6. symbol table: it contains info about functions and its vars in a tabular format.

Operating System Concepts

- **History of Operating System**

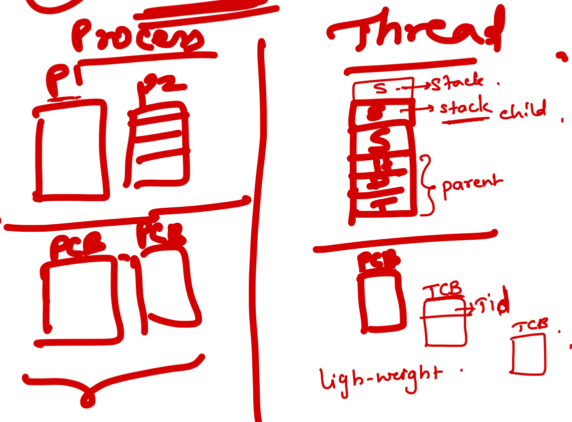
1. **Resident monitor**
2. **Batch System**
   - The batch/group of similar programs is loaded in the computer, from which OS loads one program in the memory and execute it. The programs are executed one after another.
   - In this case, if any process is performing IO, CPU will wait for that process and hence not utilized efficiently.
3. **Multi-programming**
   - Better utilization of CPU
   - Loading multiple Programs in memory
   - Mixed program(CPU bound + IO bound)
4. **Time-sharing/Multitasking**
   - Sharing CPU time among multiple process/task present in main memory and ready for execution
   - Any process should have response time should be less then 1sec
   - Multi-tasking is divided into two types
     - Process based multitasking
     - Thread based multitasking



Prog → CPU + IO
time time.

① Process

② Thread



- Process based multitasking: Multiple independent processes are executing concurrently. Processes running on multiple processors called as "multi-processing".
- Thread based multi-tasking OR multi-threading: Multiple parts/functions in a process are executing concurrently.

Thread is a light weight process

- When new thread is created a new stack and new TCB is created.
- Thread Share text, data, heap sections with the parent process

Process vs thread

- In modern OS, process is a container holding resources required for execution, while thread is unit of execution/scheduling.
  Process holds resources like memory, open files, IPC (e.g. signal table, shared memory, pipe, etc.). PCB contains resources information like pid, exit status, open files, signals/ipc, memory info, etc.
- CPU time is allocated to the threads. Thread is unit of execution.
- TCB contains execution information like tid, scheduling info (priority, sched algo, time left, ...), Execution context, Kernel stack, etc.
- For each process one thread is created by default it is called as main thread.

5. Multi-user system

Multiple users runs multiple programs concurrently

6. **Multi-processor/Mutli-core system**

System can run on a machine in which more than one CPU's are connected in a closed circuit.

Multiprocessing Advantage is it increased throughput (amount of work done in unit time)

- There are two types of multiprocessor systems:
- Asymmetric Multi-processing Symmetric Multi-processing

- **Asymmetric Multi-processing**

OS treats one of the processor as master processor and schedule task for it. The task is in turn divided into smaller tasks and get them done from other processors.

- **Symmetric Multi-processing**

OS considers all processors at same level and schedule tasks on each processor individually. All modern desktop systems are SMP.

Process life Cycle

PCB.
↳ Sched info
↳ (1) State
(2) Priority.
(3) Sched alog

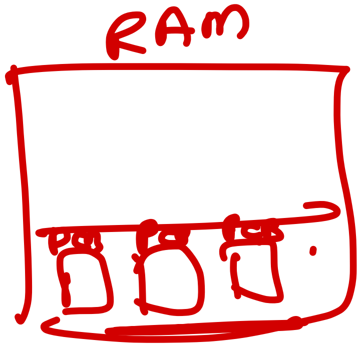
- **Process States:**

To keep track on all running programs, an OS maintains few **data structures** referred as **OS data Structure**

1. **Job queue:** it contains list of all the processes(PCB).

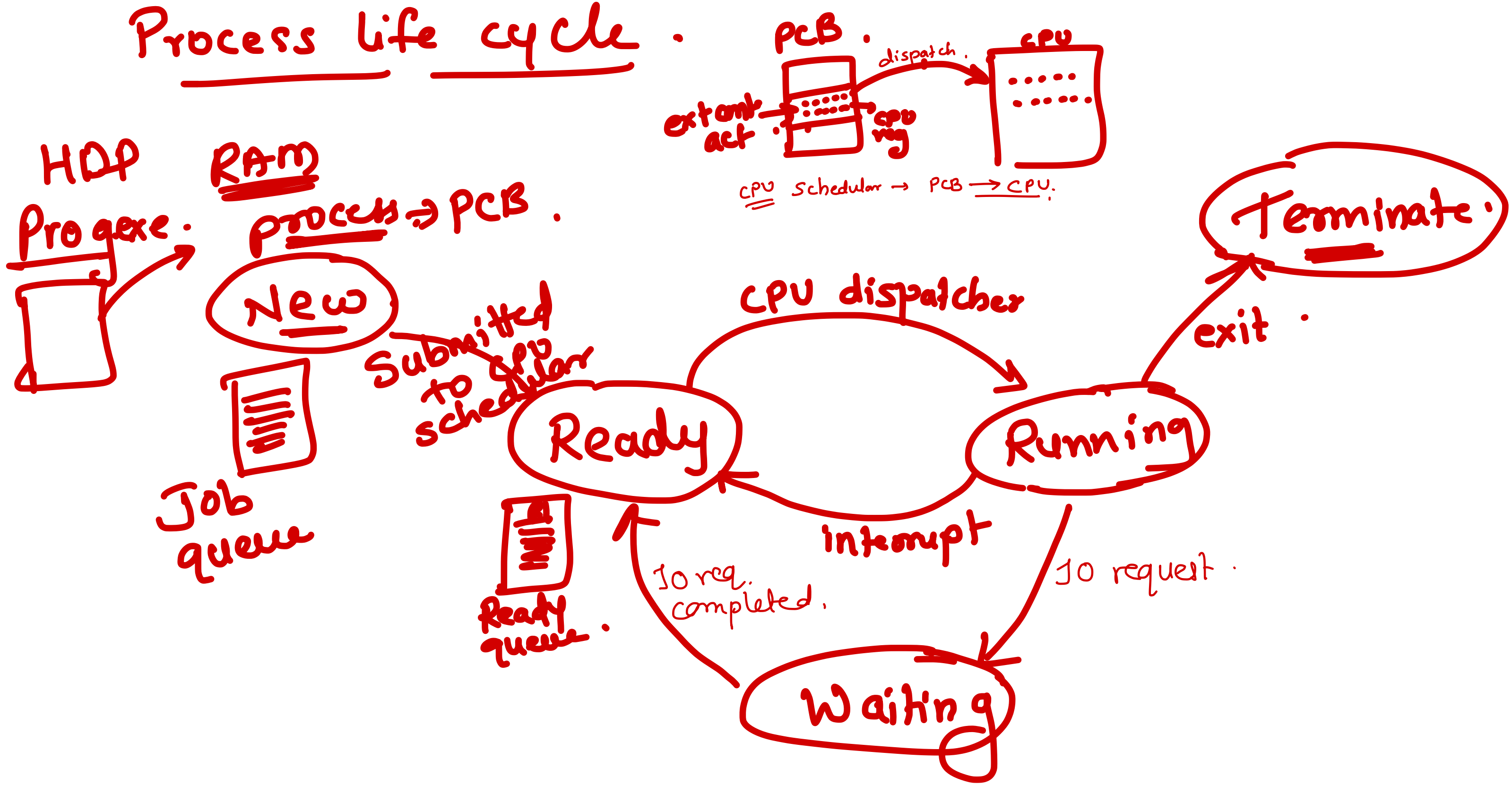
2. **Ready queue:** it contains list of PCB's of processes which are ready to run on CPU.

3. **Waiting queue:** it contains list of PCB's of processes waiting for io device or for synchronization.

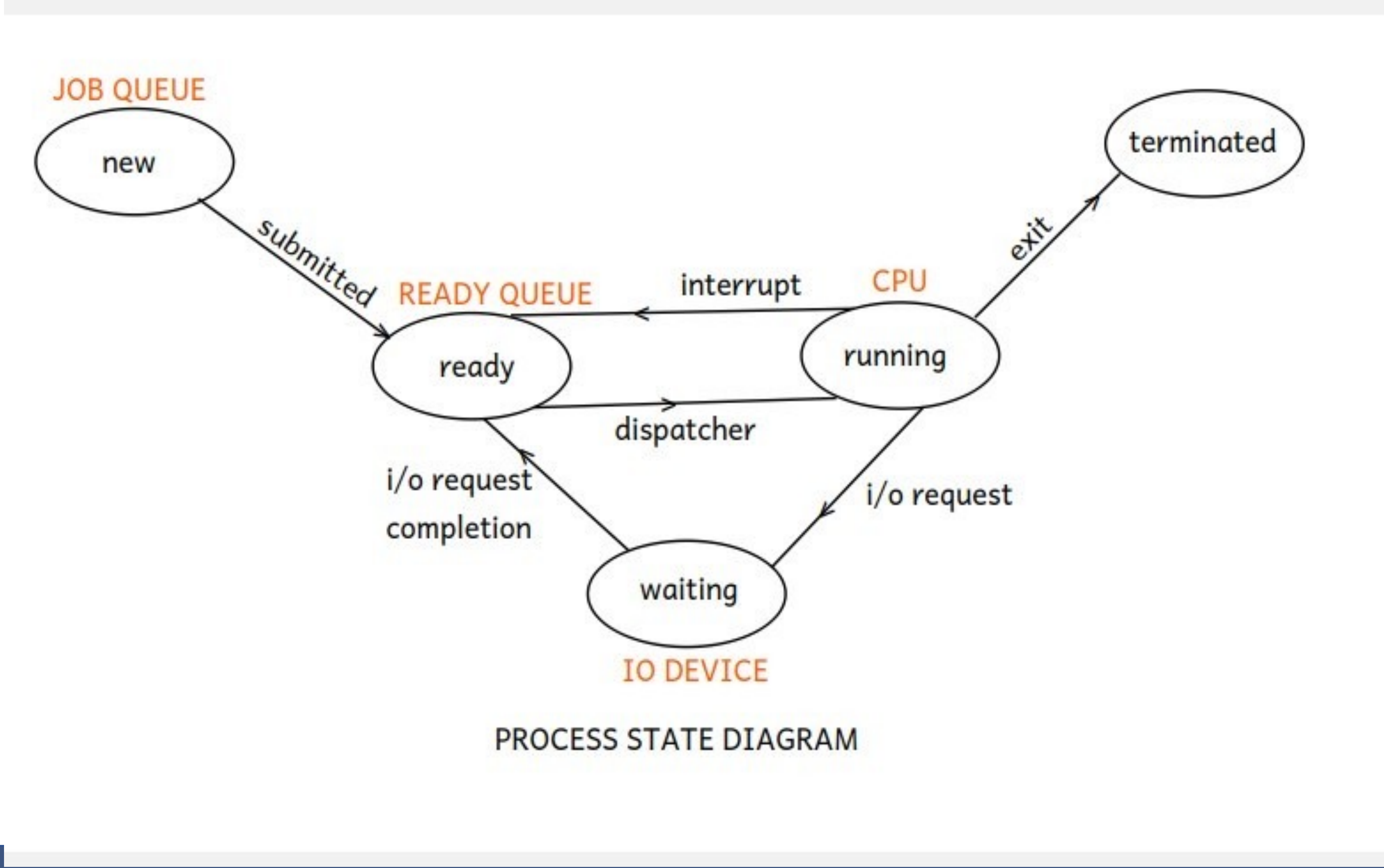


Sunbeam Infotech

Process life cycle.



Process State Diagram



Sunbeam Infotech

Process States

Throughout execution, process goes through different states out of which at a time it can be only in a one state.

-States of the process:

1. **New**: New process PCB is created and added into job queue. PCB is initialized and process get ready for execution.

2. **Ready**: The ready process is added into the ready queue. Scheduler pick a process for scheduling from ready queue and dispatch it on CPU.

3. **Running**: The process runs on CPU. If process keeps running on CPU, the timer interrupt is used to forcibly put it into ready state and allocate CPU time to other process.

4. **Waiting**: If running process request for IO device, the process waits for completion of the IO. The waiting state is also called as sleeping or blocked state.

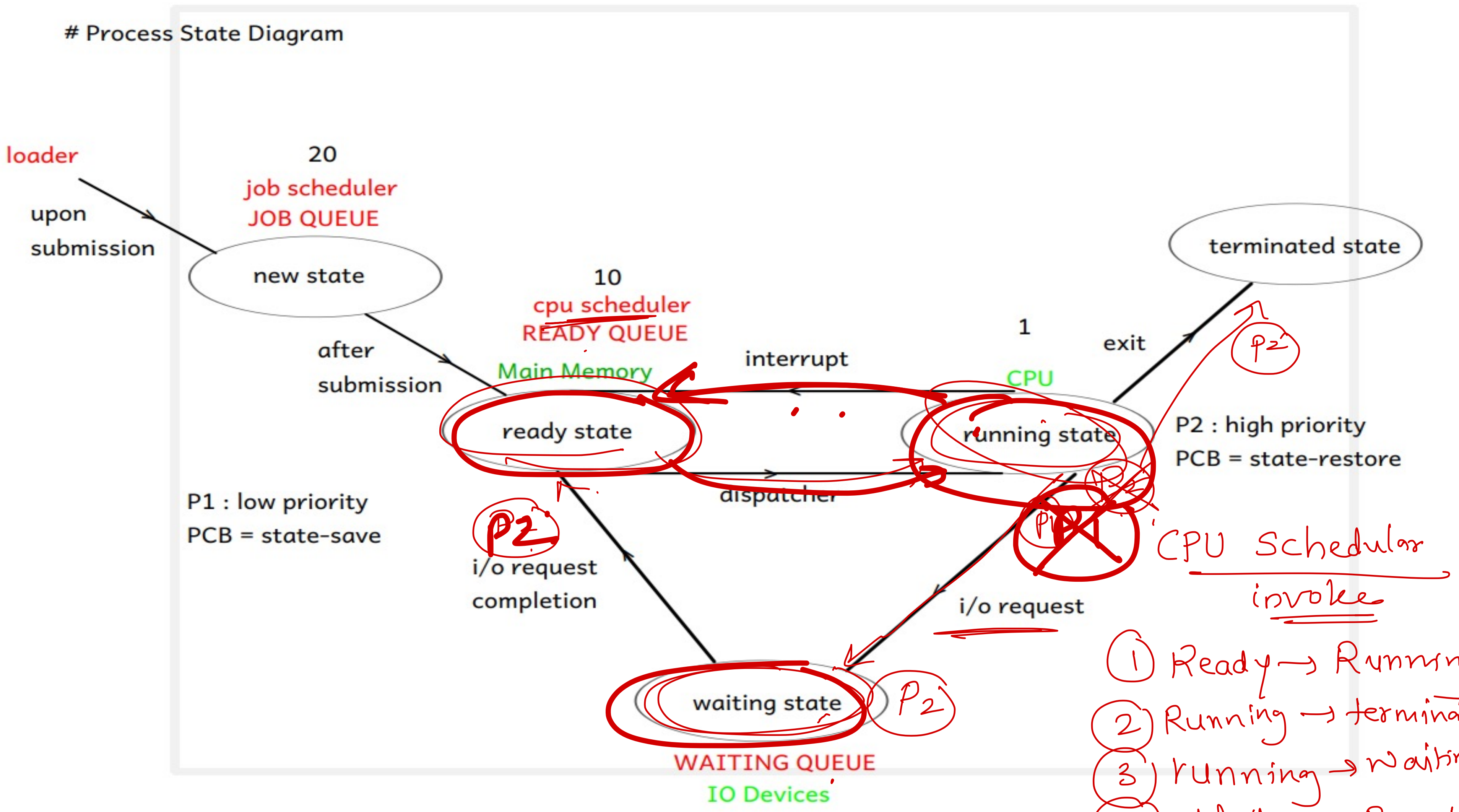
5. **Terminated**: If running process exits, it is terminated.

Sunbeam Infotech

Schedulers

- **Job Scheduler/long term schedulers**
  - Job scheduler load the programs into main memory. Used in older mainframe systems.
- **CPU Scheduler/Short term schedulers**
  - CPU scheduler pick the process to be executed on CPU from ready processes.
  - selects which process should be executed next and allocates CPU
- **CPU Dispatcher**
  - It is a system program that loads a process onto the CPU that is scheduled by the CPU scheduler.
  - Time required for the dispatcher to stops execution and one process and starts execution of another process is called as **"dispatcher latency".**

Process State Diagram



CPU Schedular
invoke

1) Ready → Running
2) Running → terminated
3) running → waiting
4) Waiting → Ready

CPU Management

- **CPU scheduler is invoked**

1. Running -> Terminated
2. Running -> Waiting
3. Running -> Ready
4. Waiting -> Ready

non-pre-emptive.

pre-emptive.

CPU Management

- **Types of Scheduling**
- **Non-preemptive**
  - The current process gives up CPU volunteerily (for IO, terminate or yield).
  - Then CPU scheduler picks next process for the execution.
  - If each process yields CPU so that other process can get CPU for the execution, it is referred as "Co-operative scheduling". e.g. Windows 3.x, etc.
- **Pre-emptive**
  - The current process may give up CPU volunteerily or paused forcibly (for high priority process or upon completion of its time quantum)

CPU Scheduling algorithms

- Schedular decides which next process to execute depending on some Scheduling Algorithm

1. FCFS : First Come First Served
2. SJF: Shortest Job First
3. Priority Scheduling
4. Round Robin
5. Multi-level Queue
6. Multi-level Feedback Queue

Turn-around time.

CPU waiting time + CPU time + IO waiting + IO time.

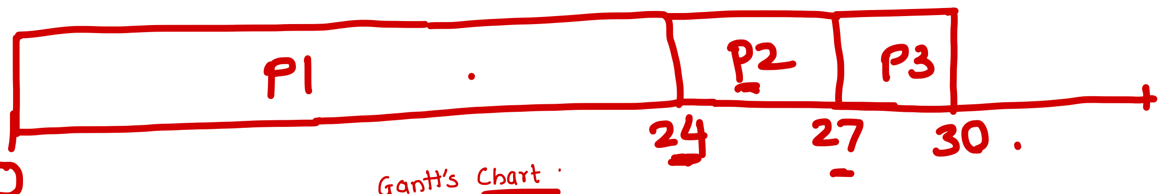
CPU Scheduling Criteries

Scheduling criteria's

- **CPU utilization**: Ideal - max
  - On server systems, CPU utilization should be more than 90%.
  - On desktop systems, CPU utilization should around 70%.
- **Throughput**: Ideal - max
  - The amount of work done in unit time.
- **Waiting time**: Ideal - min
  - Time spent by the process in the ready queue to get scheduled on the CPU.
  - If waiting time is more (not getting CPU time for execution) -- Starvation.
- **Turn-around time**: Ideal - CPU burst + IO burst
  - Time from arrival of the process till completion of the process.
  - CPU burst + IO burst + (CPU) Waiting time + IO Waiting time
- **Response time**: Ideal - min
  - Time from arrival of process (in ready queue) till allocated CPU for first time.

FCFS → First Come First serve.

Process	Arrival time	CPU Burst time	Waiting time	turn around time
P1	0 →	24	0 ⇒	24
P2	0 →	3	24 ⇒	27
P3	0 →	3	27 ⇒	30



Gantt's Chart

→ P1
{ P2
P3

avg waiting time $= \frac{0+24+27}{3} = \frac{51}{3} = 17$

non-premptive.

avg turnaround time $= \frac{24+27+30}{3} = 27$

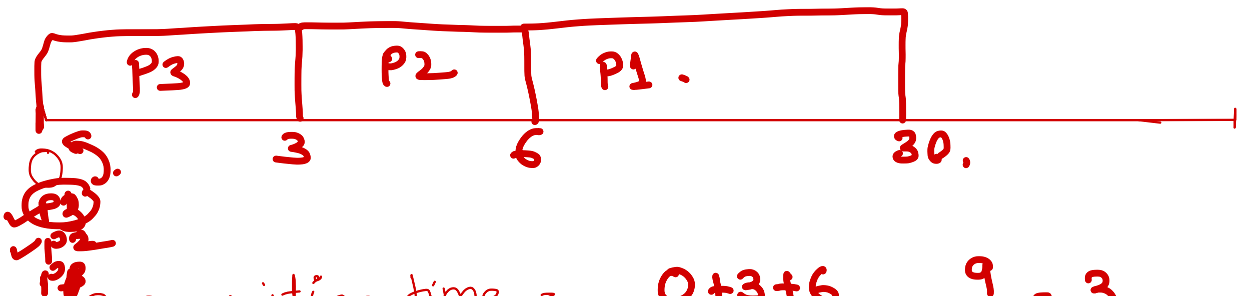
Response time P2 = 24
P3 = 27

Response P1 = 0

FCFS

Convoy effect.

Process	Arrival	CPU Burst	Waiting time.
P3	0	3	0
P2	0	3	3
P1	0	24	6

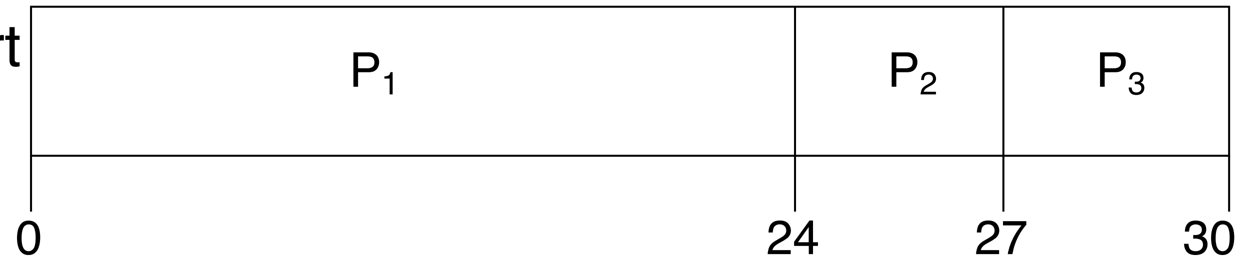


avg waiting time $= \dfrac{0+3+6}{3} = \dfrac{9}{3} = 3$

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
- The Gantt Chart



Waiting time for P_1 = 0; P_2 = 24; P_3 = 27

- Average waiting time: (0 + 24 + 27)/3 = 17

- *Convoy effect -> If bigger processes arrive first, average waiting time increases*

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

P_2 , P_3 , P_1 .

The Gantt chart for the schedule is:

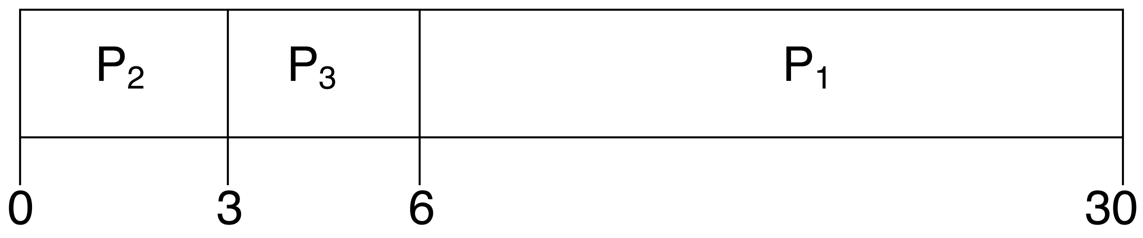
CPU burst time -> p3 = 3, p2 = 3 , p1 = 24

Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$

Average waiting time: (6 + 0 + 3)/3 = 3

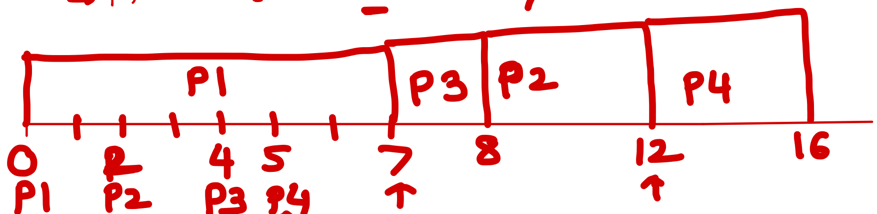
Much better than previous case.

Convoy effect short process behind long process



SJF → Shortest Job first → non-pre-emphive

Process	arrival	CPU Brust	avg waiting time
P1	0	7	0
P2	2	4	6
P3	4	1	3
P4	5	4	7

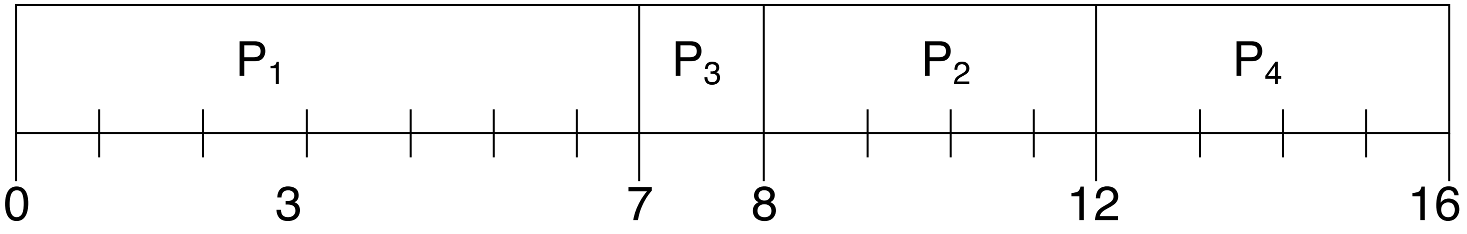


avg waiting time $= \frac{0+6+3+7}{4} = \underline{\underline{4}}$

Example of SJF

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- Average waiting time = (0 + 6 + 3 + 7)/4 = 4
  - *P1 waiting time = 0*
  - *P2 waiting time = 6 (8-2)*
  - *P3 waiting time = 3(7-4)*
  - *P4 waiting time = 7(12-5)*



SJF → Shortest Remaining Time first. → pre-emptive.

Process	arrival	CPU Burst	remain time	waiting time
P1	0	7	→ 7-2=5 ✓	9
P2	2	4	→ 4-2= 2 ✗	1
P3	4	1	→ 1-1=0	0
P4	5	4	→ 4 ✓	2

P1	P2	P3	P2	P4	P1
0 – 2	2 – 4	4 – 5	5 – 7	7 – 11	11 – 16

0 1 2 3 4 5 7 11 16

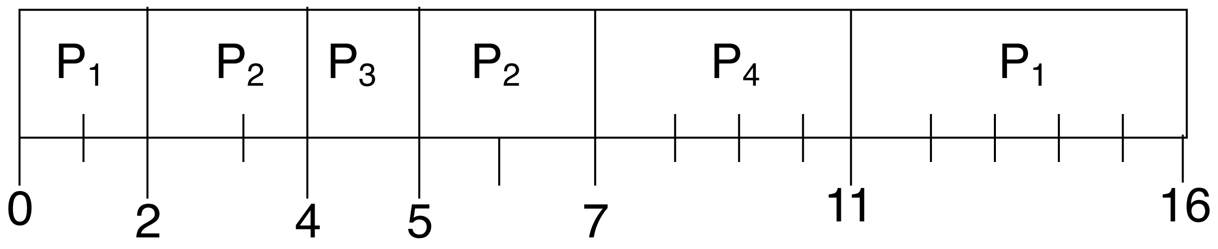
P1 (0) P2 (2) P3 (4) P4 (5)

avg waiting time = $\frac{9+1+0+2}{4}$ =

Example of Preemptive SJF (Shortest Remaining Time First [SRTF])

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Remaining time = p1 = 5, p2 = 2, p3 =1, p4 = 4
Waiting time = p1 = 9; p2 =1 , p3=0 , p4 = 2
Average waiting time = (9 + 1 + 0 +2)/4 = 3



Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - Non-preemptive
- **Starvation** – A process is not enough CPU time for its execution.(waiting in the ready queue)
- Solution of starvation is Aging – The process spending long time in ready queue, increase its priority dynamically.

Example of Priority Scheduling

→ non-pre-emptive

Process	arrival	CPU Burst Time	Priority	Waiting time
→ P_1	0	10	3 ✓	6
→ P_2	0	1 (5)	1(high priority) ✓	0
P_3	0	2	4(low priority)	16
→ $P_{\cancel{5}}$ 4	0	5	2 ✓	1

P2→P~~5~~4 →P1→ p3
Waiting time (p1 = 6, p2 =0 , p3=16 , p5= 1)

avg waiting time = $\frac{6+0+16+1}{4}$ = 5.75

Average waiting time =(6 +0+ 16 + 1) /3= 5.75

P1.
P2
P3
P4

Priority Sch

→ P1 (8)

→ P2 (10) ✓

→ P3 (3)

→ P4 (5)

P5 (7)

P6 (9)

✓ P7 (2) ✓

Starvation

↓

Aging.

Running Process.

P1

P3.

P4

P5

P6

P7

P2