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? → OS

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] → Linux .exe

E. None of the above

F. All of the above

Quiz

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

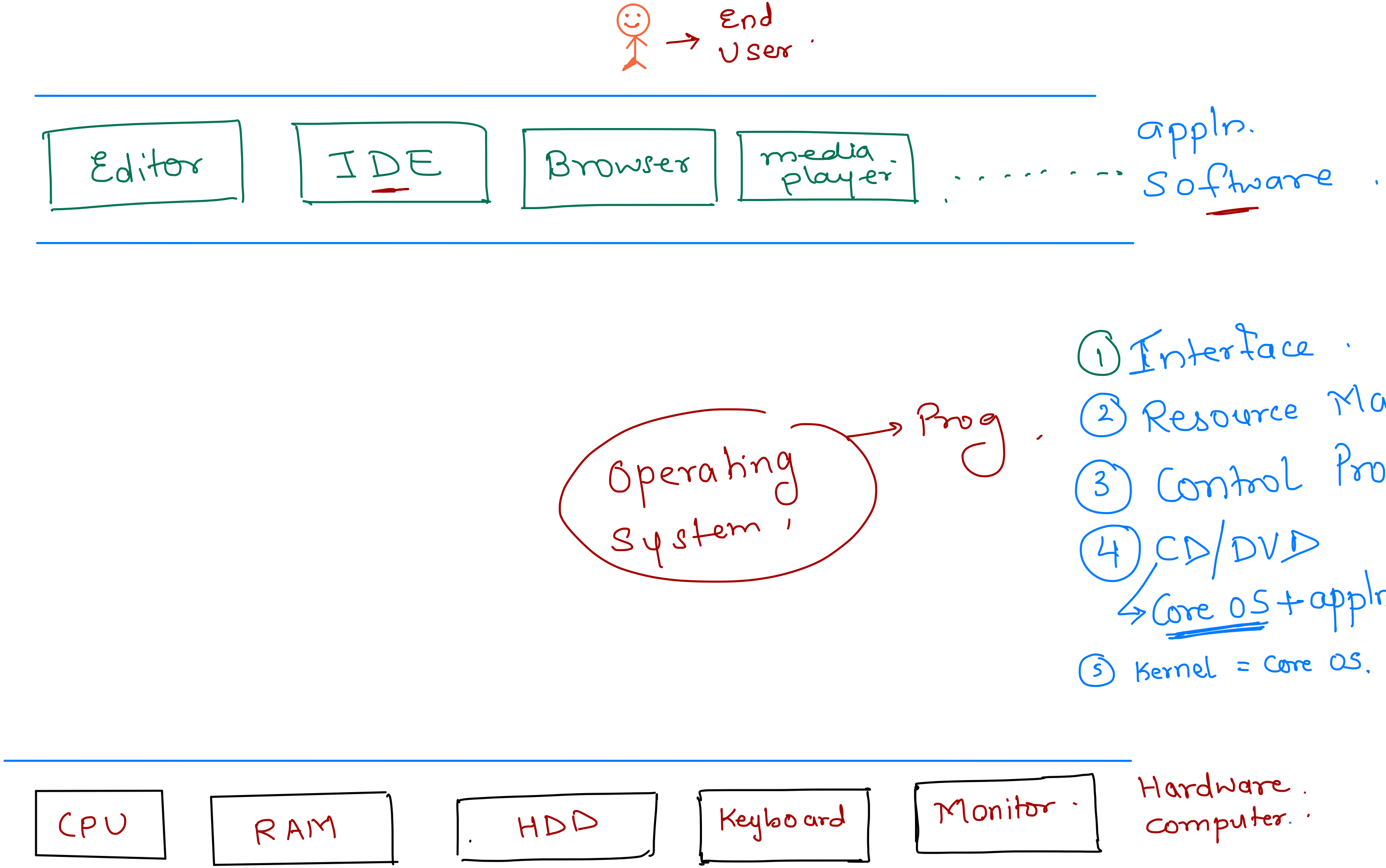
a. cmd.exe

[✓ b. explorer.exe]

c. command.com

d. all of the above

e. none of the above



1. Interface .
2. Resource Manger .
3. Control Program
4. CD/DVD
   → Core OS + appln prog + Utilities
5. Kernel = Core OS.

Operating System Concepts

# Functions of an OS:

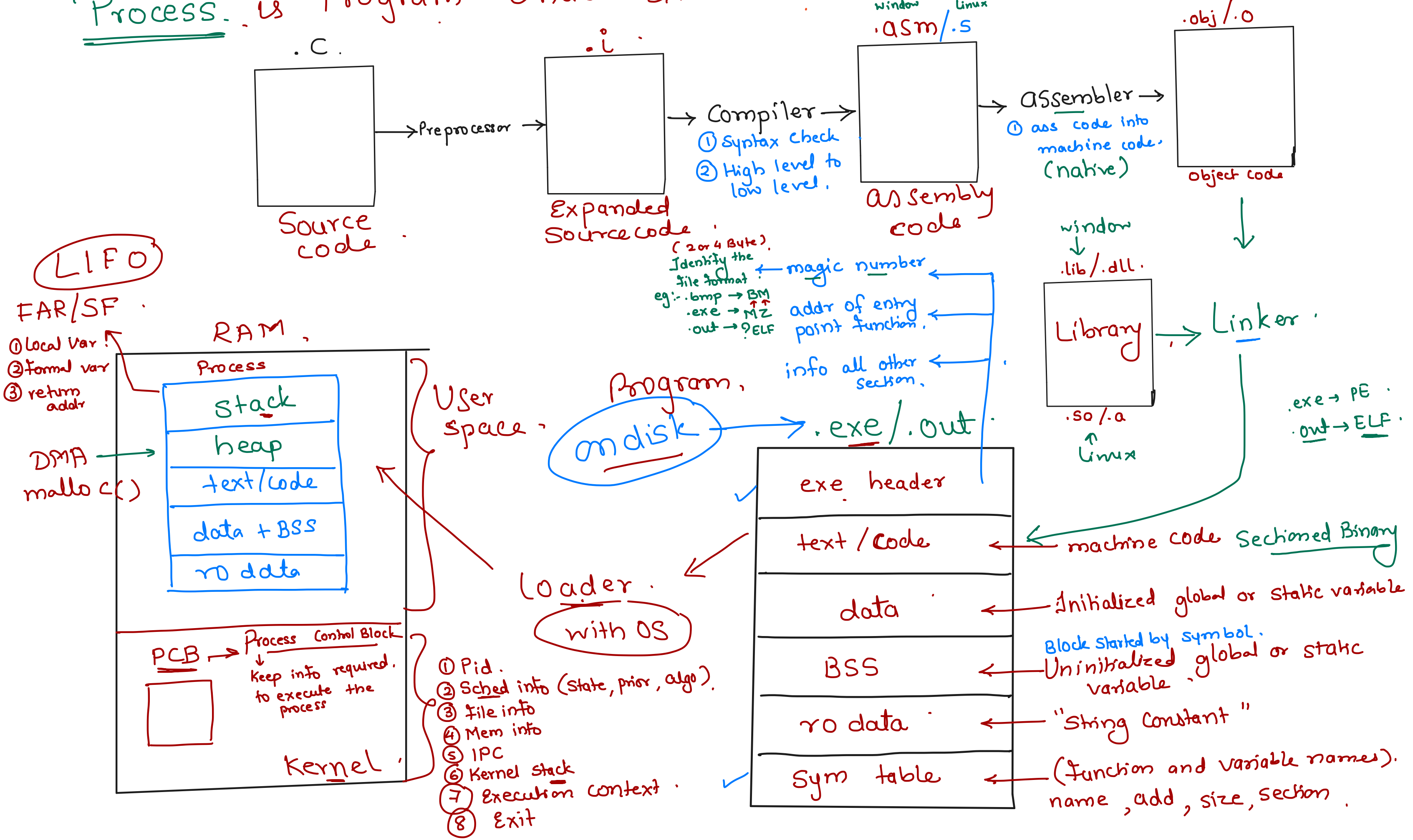
Basic minimal functionalities/Kernel functionalities:

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

Process is Program Under Execution.



.c
Source code
→ Preprocessor →
.i
Expanded Source code.
→ Compiler →
① Syntax check
② High level to low level.

window .asm / .s Linux
assembly code
→ assembler →
① ass code into machine code. (native)

.obj / .o
object code

window
.lib / .dll.
Library
.so / .a
Linux
→ Linker.

.exe → PE.
.out → ELF.

LIFO
FAR/SF.
① Local var!
② formal var
③ return addr

DMA
malloc()

RAM.
Process
stack
heap
text/code
data + BSS
ro data

User space.

PCB → Process Control Block.
Keep info required to execute the process
Kernel.

① Pid.
② Sched info (State, prior, algo).
③ File info
④ Mem info
⑤ IPC
⑥ Kernel stack
⑦ Execution context.
⑧ Exit

Loader.
with OS

Program,
on disk → .exe / .out.

(2 or 4 Byte).
Identify the file format. ← magic number
eg:- .bmp → BM
.exe → MZ
.out → ?ELF

addr of entry point function.
info all other section.

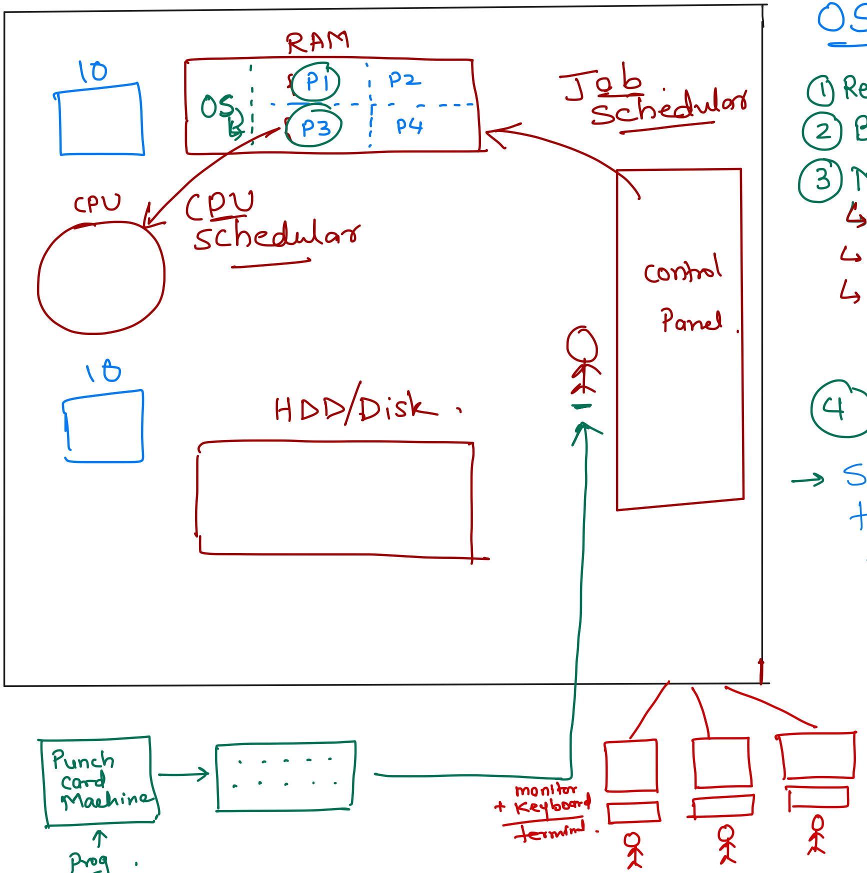
exe header
text / code ← machine code Sectioned Binary
data ← Initialized global or Static variable
Block Started by symbol.
BSS ← Uninitialized global or static variable.
ro data ← "String Constant"
Sym table ← (function and variable names).
name, add, size, section.

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

Old Computers → Mainframe computers.



Prog → CPU Burst time. + IO ~~Burst~~ Burst time

OS history / Evolution.

1. Resident Monitor.
2. Batch System.
3. Multi_Programming
   ↳ multiple prog load RAM.
   ↳ better utilization of CPU.
   ↳ mixed prog.
   CPU + IO.
4. time sharing / multi-tasking

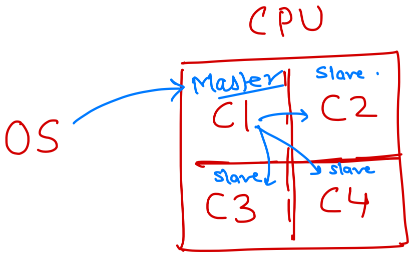
→ Sharing CPU time among multiple tasks present in mem & ready for execution.

Response < 1sec.

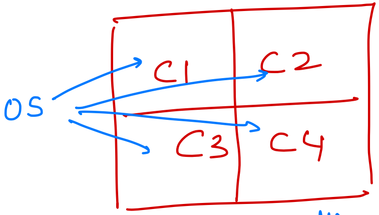
multi_tasking
- process Based
- thread Based (multi-threading)

5. Multi-user system.
6. Multi-processor/Multi-core.

Multi-core system/ Multi- process.



Asymmetric multi-core system.



symmetric multi-core system.

eg:- Vista 7, 8, 10, 11

- 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.

Operating System Concepts

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

Process → PCB

↓

sched info

① state ② Priority ③ algo . . . . . .

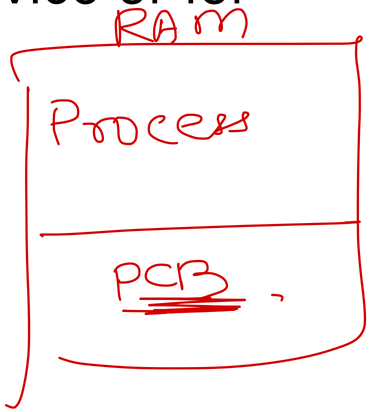
- **Process States:**

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.

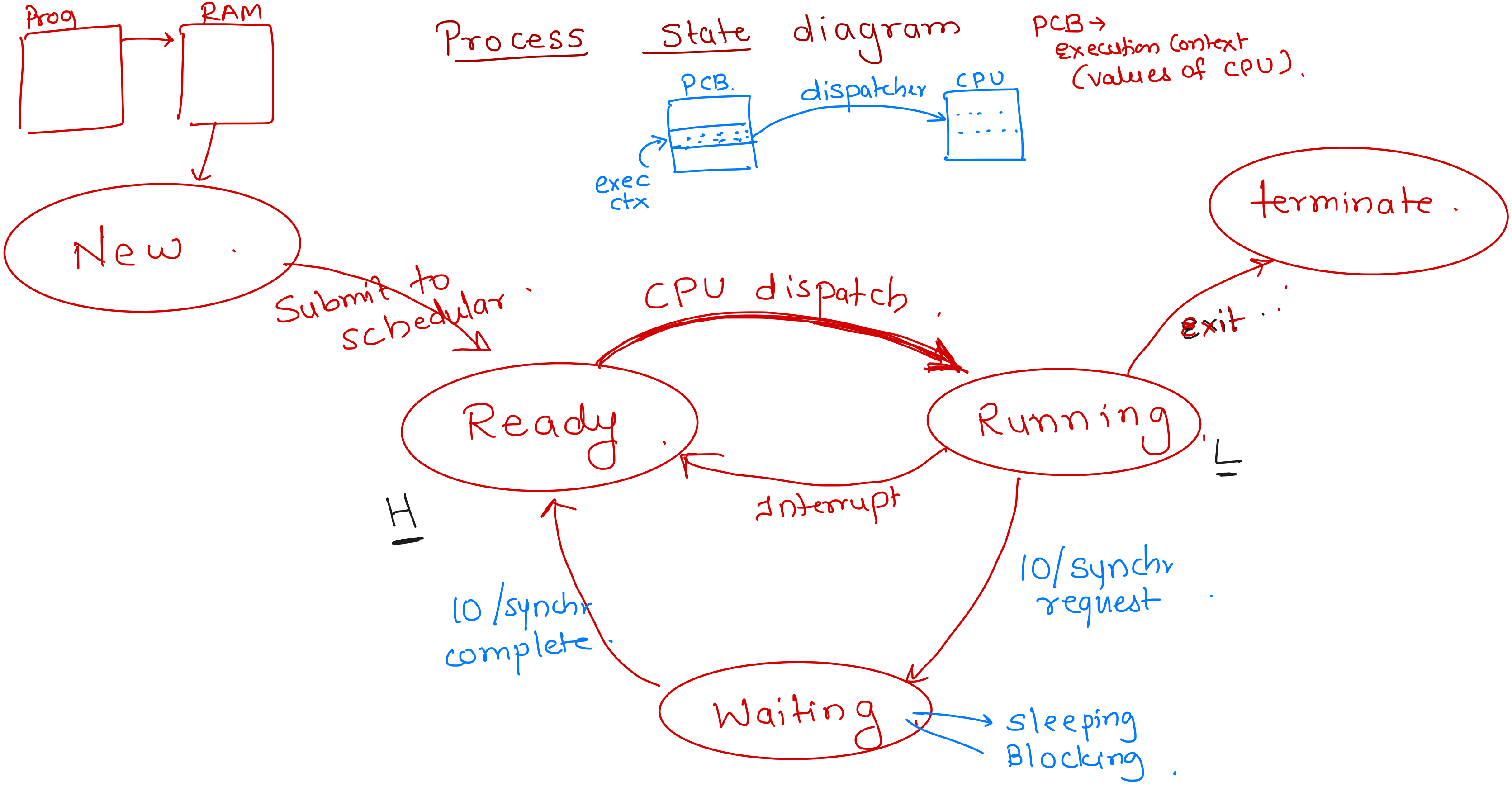
3. **Waiting queue:** it contains list of PCB's of processes waiting for io device or for synchronization.



Sunbeam Infotech

Process State diagram

PCB → execution context (values of CPU).



CPU Scheduler → decide which process to run next.

CPU Dispatch → dispatch the selected process on CPU.

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**
  - The job scheduler loads the programs into the main memory. Used in older mainframe systems.

- **CPU Scheduler/Short-term schedulers**
  - CPU scheduler picks the process to be executed on the CPU from ready processes.
  - selects which process should be executed next and allocates CPU

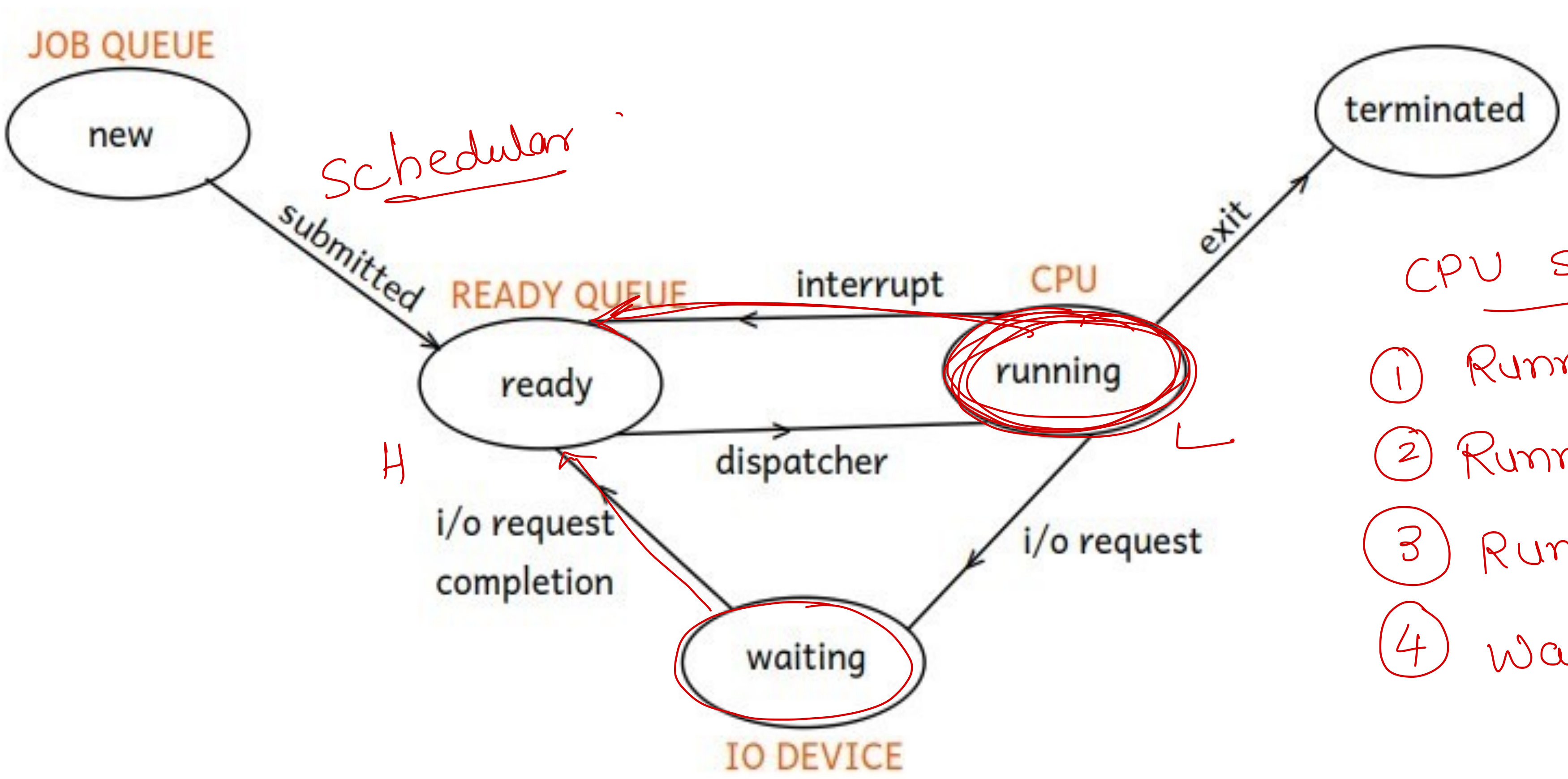
- **CPU Dispatcher**
  - It is a system program that loads a process onto the CPU that is scheduled by the CPU scheduler.
  - The time required for the dispatcher to stop the execution and one process and start the execution of another process is called **"dispatcher latency".**

CPU Scheduling

Context Switch

- Execution context is values of CPU registers (while executing the process).
- Current running process execution context is saved in PCB of that process and next process's execution context is loaded from its PCB into CPU. This is called as context switch.
- The context switch needs some time (in us). Having too many context switches will reduce overall system performance.

Process State Diagram



PROCESS STATE DIAGRAM

CPU sched invoke.

1. Running → terminated
2. Running → waiting
3. Running → Ready
4. Waiting → Ready

Sunbeam Infotech

CPU Management

- **CPU scheduler is invoked**

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

non-pre-emptive sched.

↓

co-operative sched.

→ pre-emptive sched.

CPU Management

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

CPU Scheduling algorithms

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

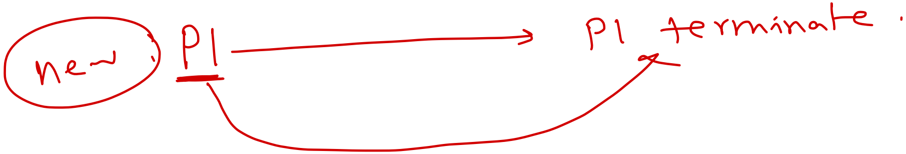
1. FCFS: First Come First Served ✓
2. SJF: Shortest Job First ✓
3. Priority Scheduling ✓
4. Round Robin ✓
5. Multi-level Queue
6. Multi-level Feedback Queue

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 be 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 the waiting time is longer (not getting CPU time for execution) -- Starvation.
- **Turn-around time**: Ideal - CPU burst + IO burst → min
  - Time from the 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 the arrival of the process (in the ready queue) till the allocated CPU for the first time.

Turn around time.

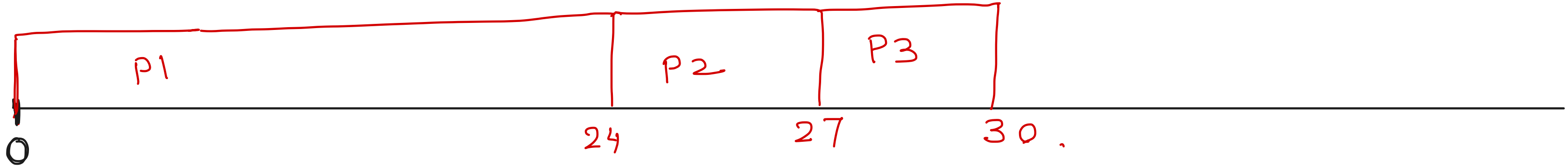


Turn around time = CPU waiting time + CPU Burst time + IO waiting time. + IO Burst time.

FCFS → First Come First Serve.

Process	Arrival	CPU. Burst.	Waiting time. ✓	Response time	Turn around time
✓→ P1	0	24	0	0	24
✓→ P2	0	3.	24	24	27.
✓→ P3	0	3	27	27	30.

non-preemptive.



P1
P2
P3

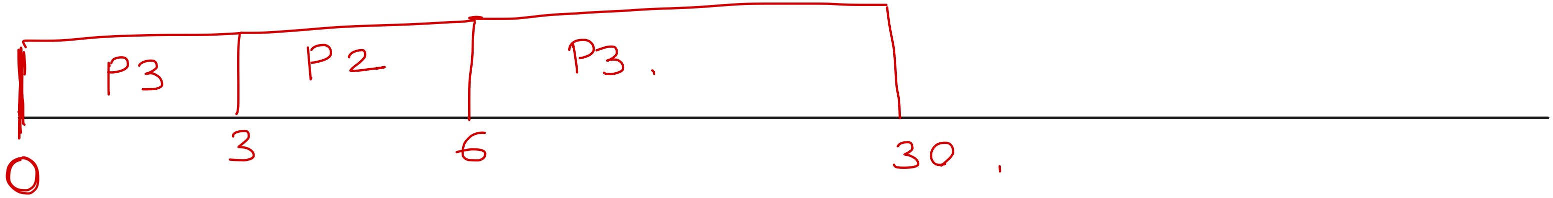
Gantt's chart.

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

$$\text{avg turn around time} = \frac{24+27+30}{3} = 27.$$

FCFS

Process	Arrival	CPU Burst	waiting time
P3 ✓	0	3	0
P2 ✓	0	3	3
P1	0	24	6



P3
P2
P1

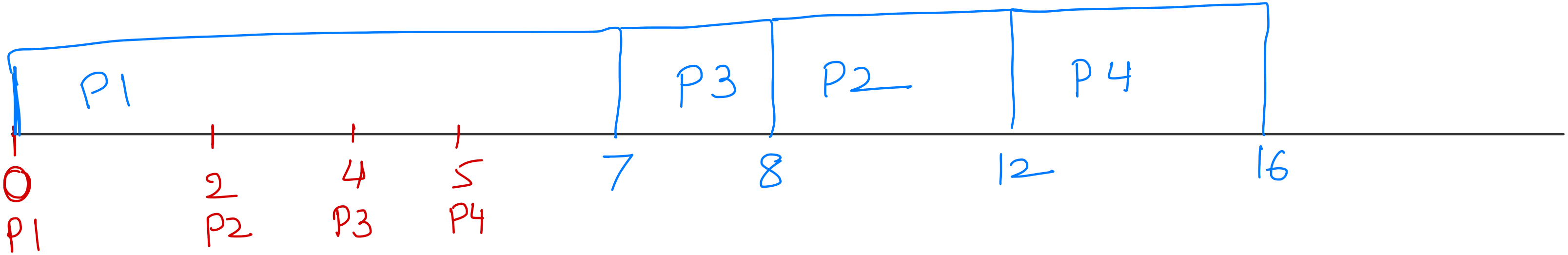
avg waiting time = $\frac{0+3+6}{3} = \frac{9}{3} = 3$

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

SJF → (Shortest Job First).

non-preemptive.

Process	Arrival	CPU Burst.	Waiting time
✓P1	0	7	0
✓P2	2	4	6
✓P3	4	1	3
✓P4	5	4	7

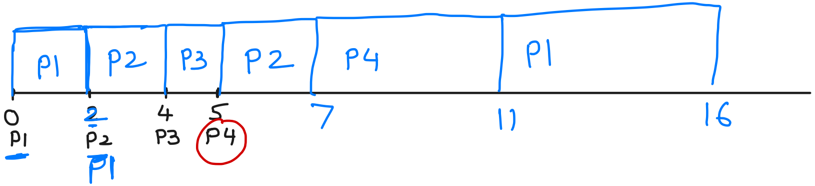


$$\text{avg waiting time} = \frac{0+6+3+7}{4} = \frac{16}{4} = 4.$$

SRTF - Shortest Remaining Time First.

pre-emptive.

Process	Arrival	CPU Burst.		remaining time.	waiting time.	response time
✓P1	0	7	→	7-2=5.	9	0
P2	2	4	→	4-2=~~2~~=	1	0
P3	4	1	→	~~1~~=0	0	0
4✓P4	5	4	→	4	2	2

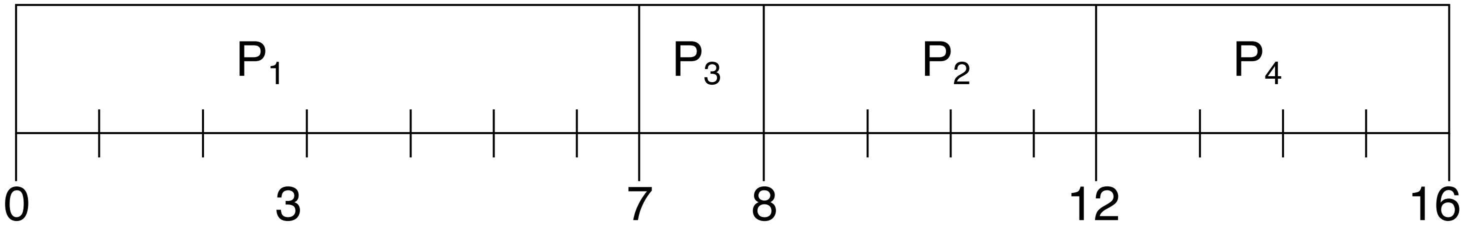


$$\text{avg waiting time} = \frac{9+1+0+2}{4} = 3.$$

Example of SJF

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

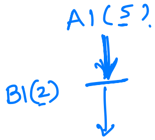
- Average waiting time = (0 + 6 + 3 + 7)/4 = 4
  - *P1 waiting time = 0*
  - *P2   waiting time = 6 (8-2)*
  - *P3   waiting time = 3(7-4)*
  - *P4   waiting time = 7(12-5)*

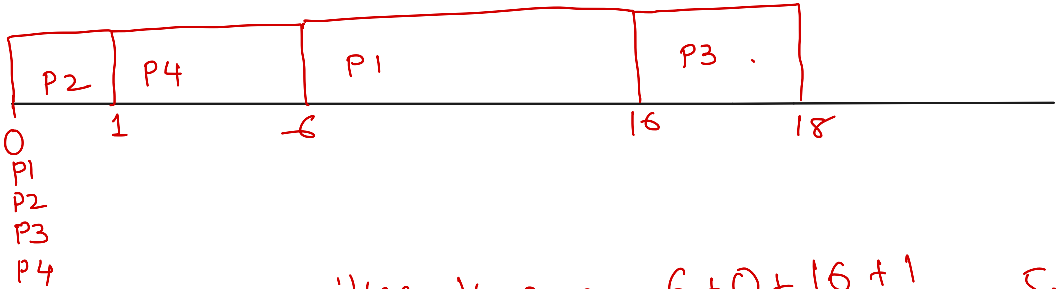


Priority

non-pre-emptive

Process	Arrival.	CPU Burst	Priority.	waiting time.
✓P1	0	10.	3	6
✓P2	0	1	1 (highest prio)	0
P3	0	2	4 (lowest prio).	16
✓P4	0	5	2	1





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

Process arrival order (non pre-emptive)
P1(8)
P2(10)
P3(3)
P4(5)
P5(7).
P6(9)
P7(2).

Starvation — A process is not getting enough CPU time for it execution.

Solution.
Aging :- The process spending long time in ready queue, increase its priority dynamically.