

Operating Systems

Quiz

Q. Which of the following statement/s is/are false about a process?

- A. Process is a running entity of a program
- B. Program in main memory is referred as a process
- C. One program may has multiple running instances i.e. processes.
- D. Program in execution is referred as a process.
- E. All of the above
- ☒ F. None of the above

Q. Process which is in the main memory waiting for the CPU time considered in a _____.

- A. waiting state
- B. new state
- ☒ C. ready state
- D. running state

Quiz

Q. _____ copies an execution context of a process which is scheduled by the scheduler from its PCB and restores it onto the CPU registers.

- A. Loader
- B. Interrupt Handler
- ☒ C. Dispatcher
- D. Job Scheduler

Q. Which of the following CPU scheduling algorithm leads to starvation?

- A. FCFS
- ☒ B. Priority
- C. Round Robin
- D. None of the above
- E. All of the above


Quiz

Q. Convoy effect occurs in _____ scheduling algorithm.

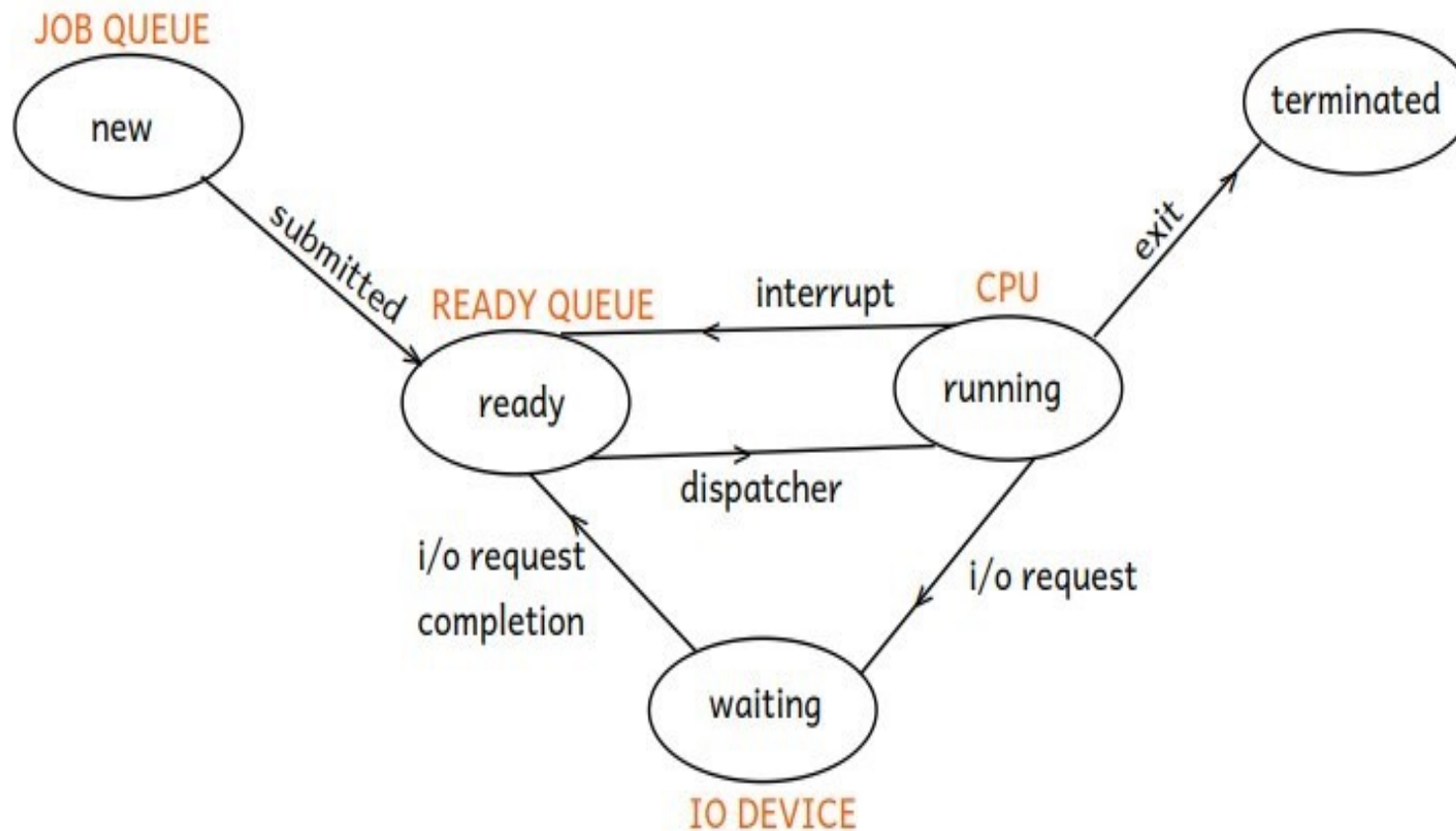
A. Priority

B. Shortest Remaining Time First

C. Shortest Next Time First

 D. None of the above

Process State Diagram



PROCESS STATE DIAGRAM

CPU scheduler is invoked

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

CPU Scheduling Criteries

Scheduling criterias

- **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 → min
 - 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.

CPU Scheduling Algorithms

- **FCFS**
 - Process added first in ready queue should be scheduled first.
 - Non-preemptive scheduling
 - Scheduler is invoked when process is terminated, blocked or gives up CPU is ready for execution. Convoy Effect: Larger processes slow down execution of other processes.
- **SJF**
 - Process with lowest burst time is scheduled first.
 - Non-preemptive scheduling
 - Minimum waiting time
- **SRTF - Shortest Remaining Time First**
 - Similar to SJF - but Pre-emptive scheduling
 - Minimum waiting time
- **Priority**
 - Each process is associated with some priority level. Usually lower the number, higher is the priority.
 - Pre-emptive scheduling or Non-Preemptive scheduling

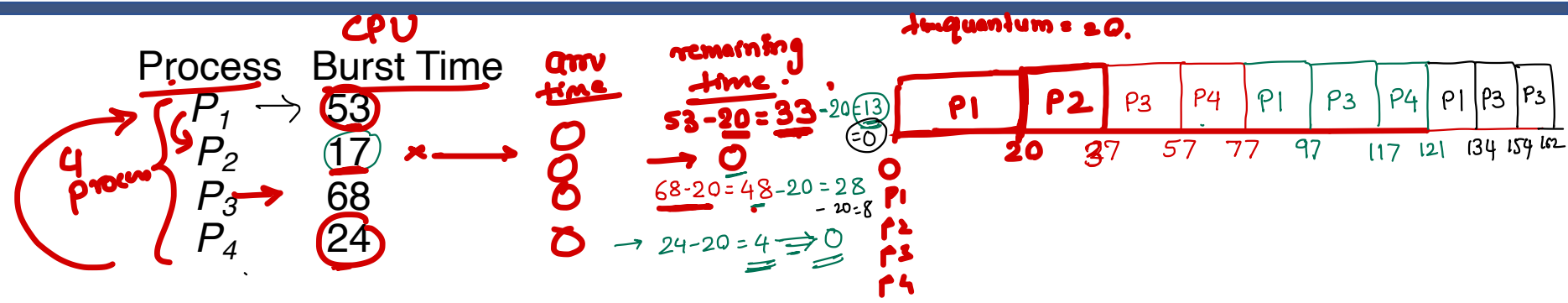
CPU Scheduling Algorithms

- Starvation
 - Problem may arise in priority scheduling.
 - Process not getting CPU time due to other high priority processes.
 - Process is in ready state (ready queue).
 - May be handled with aging -- dynamically increasing priority of the process.
- Round-Robin
 - Pre-emptive scheduling
 - Process is assigned a time quantum/slice. Once time slice is completed/expired, then process is forcibly
 - Pre-empted and other process is scheduled.
 - Min response time.
- Multi-level queue
 - In modern OS, the ready queue can be divided into multiple sub-queues and processes are arranged in them depending on their scheduling requirements. This structure is called as "Multi-level queue".
 - If a process is starving in some sub-queue due to scheduling algorithm, it may be shifted into another sub-queue. This modification is referred as "Multi-level feedback queue".
 - The division of processes into sub-queues may differ from OS to OS.

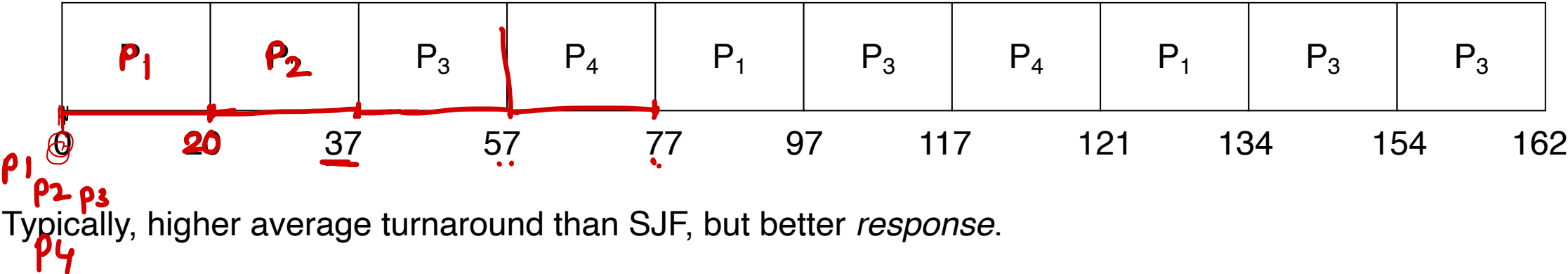
Round Robin (RR)

- Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- The ready queue is treated as a circular queue.
- The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
- Round Robin algorithm is a preemptive .

Example of RR with Time Quantum = 20 → eg → Response time is good.



The Gantt chart is:



Typically, higher average turnaround than SJF, but better response.

important service → priority

Background service → SJF

GUI appls → RR algo

Other task → FCFS

Multi-level
Ready state

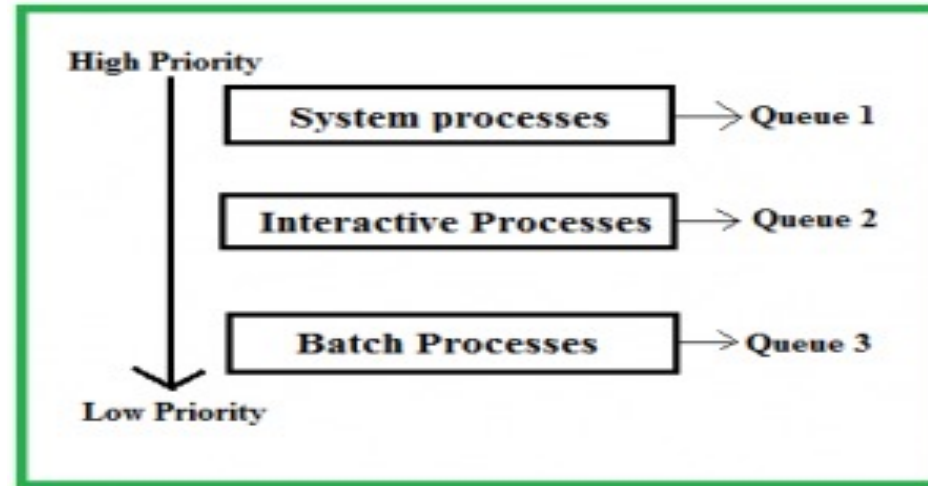
SJF → [20]

RR → [20]

→ [2]

multi-level
feedback
queue

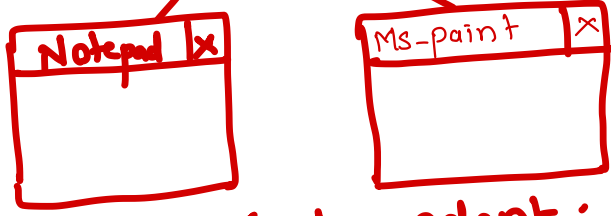
Multilevel Queue Scheduling Algorithm



- A multi-level queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.
- processes are permanently stored in one queue in the system and **do not move between the queue**.
- separate queue for foreground or background processes
- **For example:** A common division is made between foreground(or interactive) processes and background (or batch) processes.
- These two types of processes have different response-time requirements, and so might have different scheduling needs. In addition, foreground processes may have priority over background processes

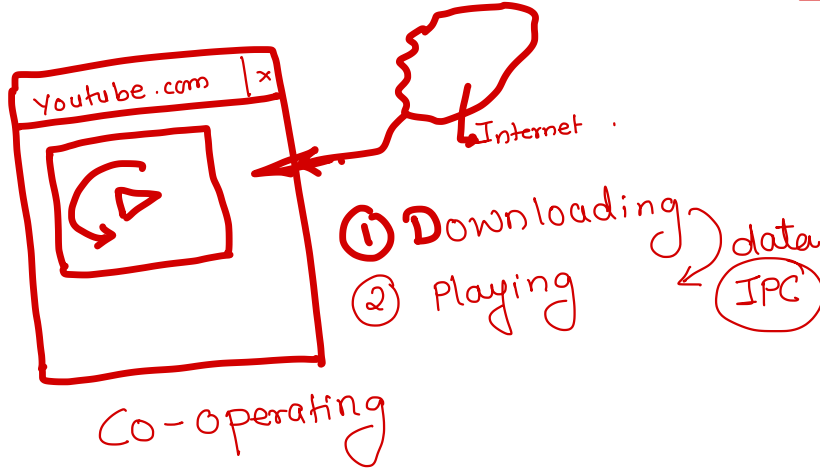
Multitasking.

①



Independent.

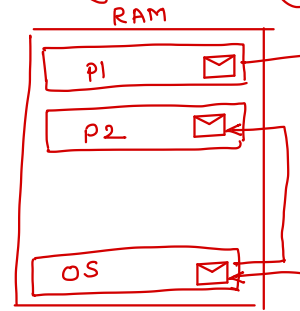
②



IPC

Message Switching

Shared Memory

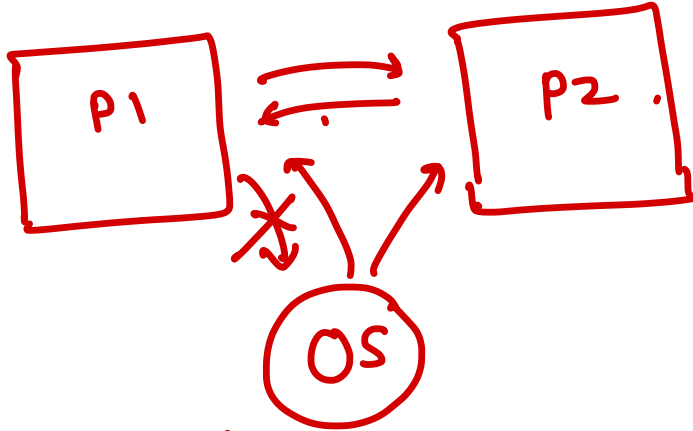


UNIX/LINUX 5 IPC mech.

- ① Signal.
 - ② Message.
 - ③ Pipe
 - ④ Socket.
 - ⑤ Shared Mem → shared mem.
- Message Switch

Signal

→ Predefine value



64 signal

Inter process Communication (IPC)

- A process cannot access the memory of another process directly. OS provides IPC mechanisms so that processes can communicate with each other.

Independent process

- do not get affected by the execution of another process

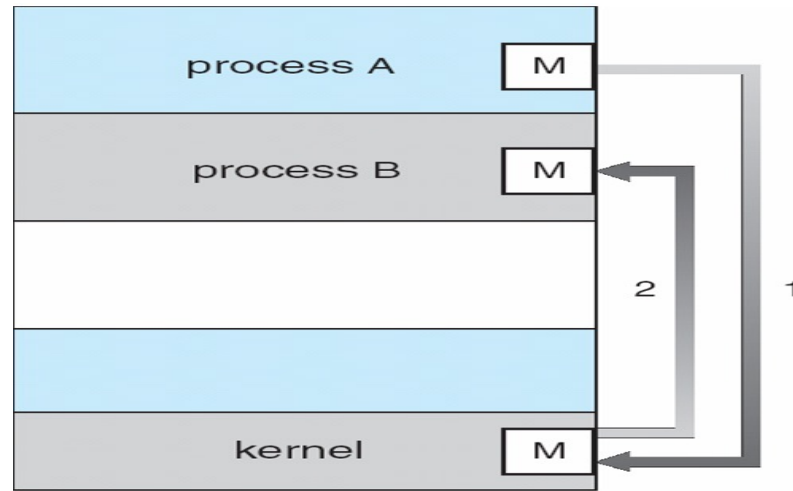
Cooperating process

- get affected by the execution of another process

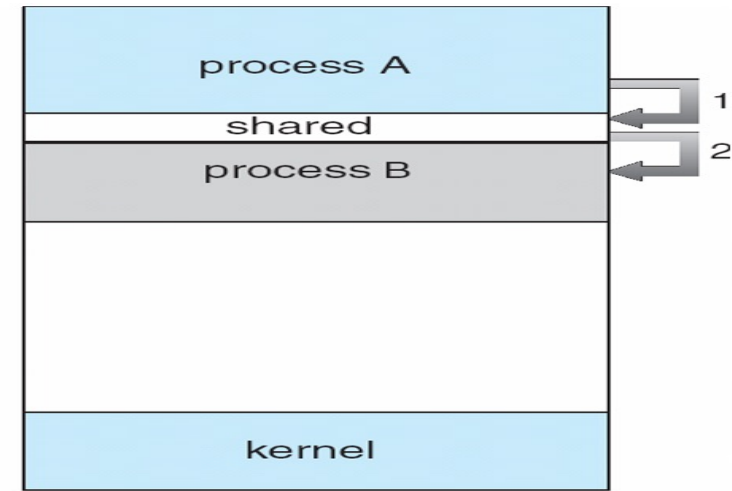
Reasons for cooperating processes:

- Information sharing
- Computation speedup
- Modularity
- Convenience
- Cooperating processes need **inter process communication (IPC)**

Mechanisms of IPC



(a) Message Passing



(b) Shared Memory Model

Message Passing

- communication takes place by means of messages exchanged between the cooperating processes
- Uses two primitives : Send and Receive

Shared Memory

- In the shared-memory model, a region of memory that is shared by cooperating processes is established.
- Processes can then exchange information by reading and writing data to the shared region.

Mechanisms of IPC

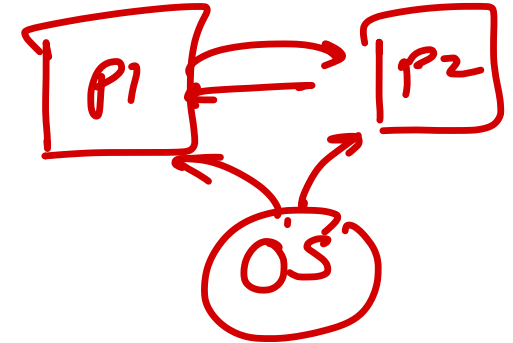
Unix/Linux IPC mechanisms

- Signals
- Shared memory
- Message queue
- Pipe
- Socket

Mechanisms of IPC

Signals

- OS have a set of predefined signals, which can be displayed using command
 - `terminal> kill -l`
- A process can send signal to another process or OS can send signal to any process.



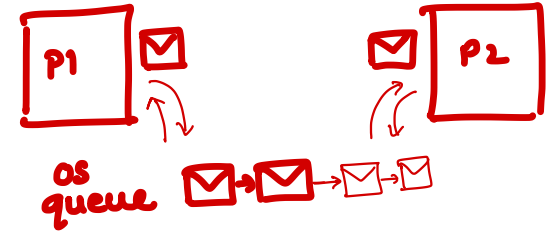
Important Signals

- SIGINT (2): When CTRL+C is pressed, INT signal is sent to the foreground process.
- SIGKILL (9): During system shutdown, OS send this signal to all processes to forcefully kill them. Process cannot handle this signal.
- SIGSTOP (19): Pressing CTRL+S, generate this signal which suspend the foreground process. Process cannot handle this signal.
- SIGCONT (18): Pressing CTRL+Q, generate this signal which resume suspended the process.
- SIGSEGV (11): If process access invalid memory address (dangling pointer), OS send this signal to process causing process to get terminated. It prints error message "Segmentation Fault".

Mechanisms of IPC

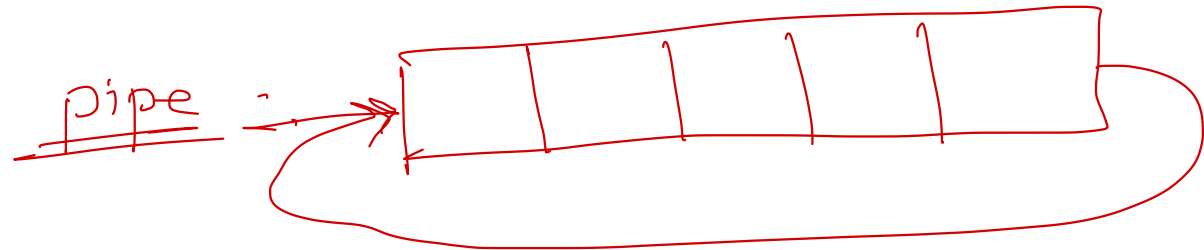
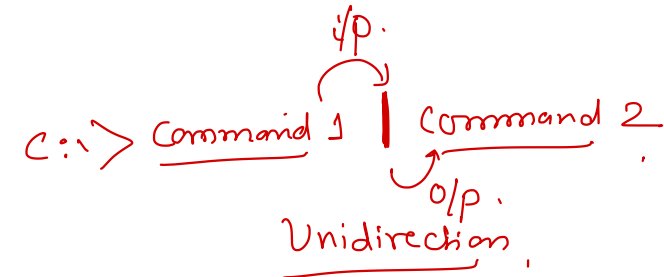
Message Queue

- Used to transfer packets of data from one process to another.
- It is bi-directional IPC mechanism.
- Internally OS maintains list of messages called as "message queue" or "mailbox".

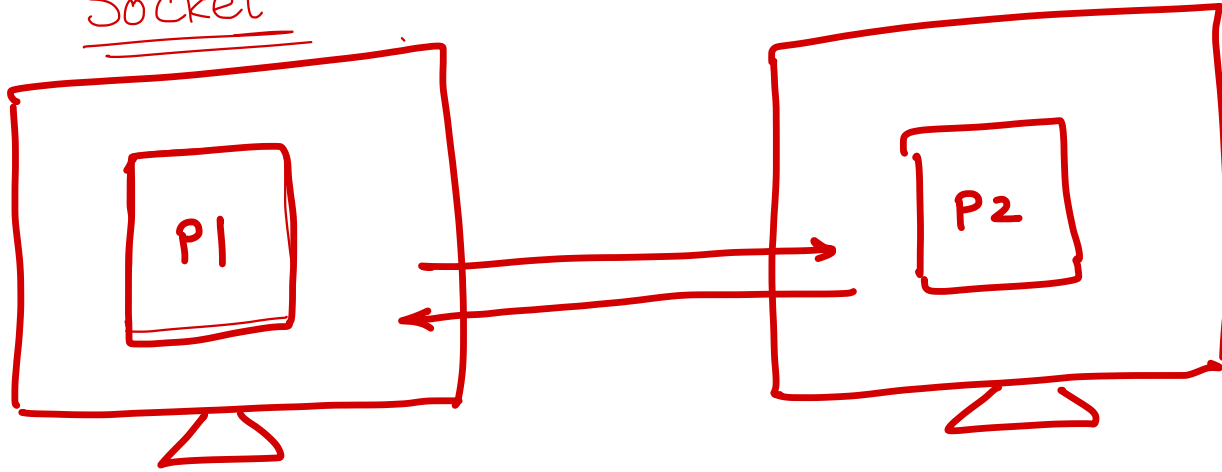


Pipe

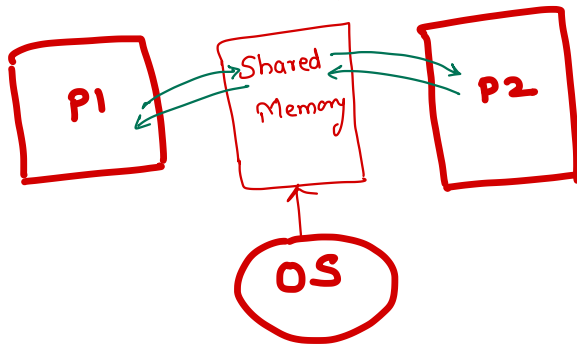
- Pipe is used to communicate between two processes.
- It is stream based uni-directional communication.
- Pipe is internally implemented as a kernel buffer, in which data can be written/read.
- There are two types of pipe:
 - Unnamed Pipe
 - Named Pipe (FIFO)



Socket



Shared Memory



Mechanisms of IPC

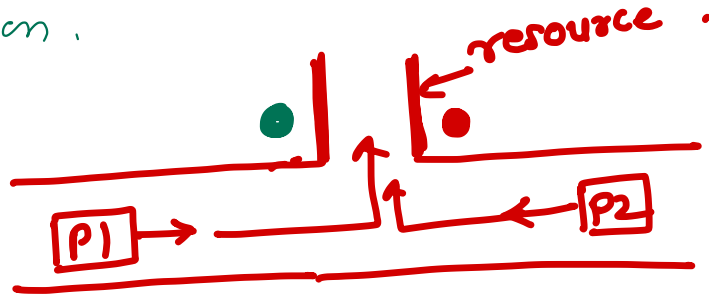
Socket

- Socket is defined as communication endpoint.
- Sockets can be used for bi-directional communication.
- Using socket one process can communicate with another process on same machine (UNIX socket) or different machine (INET sockets) in the same network.
- Sockets can also be used for communication over Bluetooth, CAN, etc.

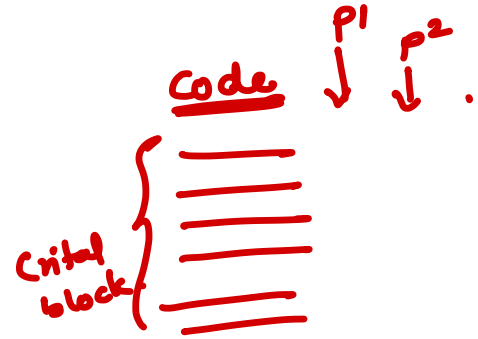
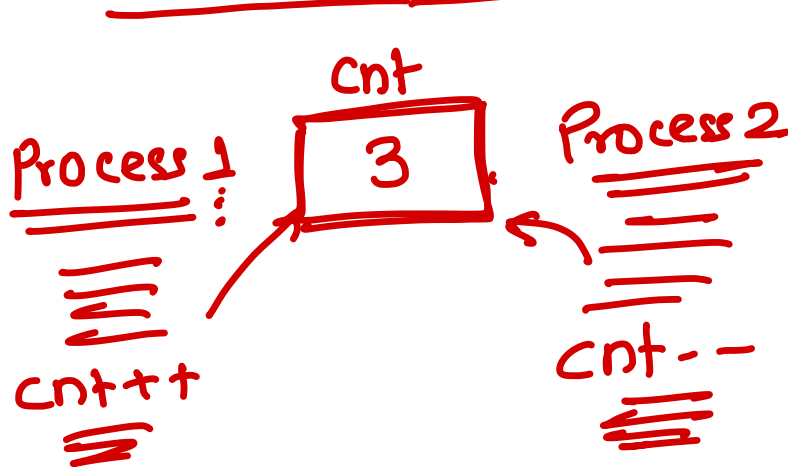
Shared memory

- OS creates a memory region that is accessible to multiple processes.
- Multiple processes accessing a shared memory need to be synchronized to handle race condition problem.
- Fastest IPC mechanism.
- Both processes have direct access to shared memory(no os invoked)

If 2 process access same resources than race around condition.



synch



Except → unchanged cnt

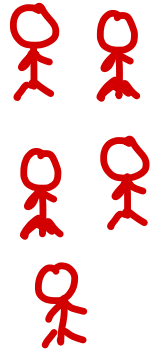
But → Both 2 try access same → data inconsistency.
2 or 4

Synchronization

- If two/multiple processes try to access same resource at same time , it is referred as “race condition”
- When race condition occurs, resource may get corrupted (unexpected results).
- Peterson's problem: If two processes are trying to modify same variable at the same time, it can produce unexpected results.
- Synchronization Mechanism ensure that only one process will access resource at a time and thus data inconsistency is avoided.
- A block of code ,executed by multiple processes at same time cause data inconsistency. Such kind of code block is called Critical section.
- To resolve race condition problem, one process can access resource at a time. This can be done using sync objects/primitives given by OS.
- OS provide some Synchronization objects
 - 1) Semaphore
 - 2) Mutex

Semaphore

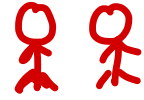
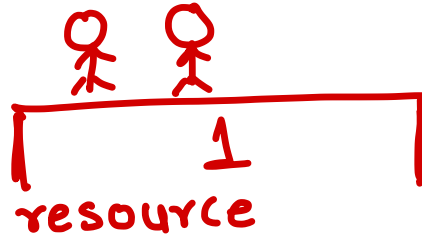
- Semaphore was suggested by Dijkstra scientist (dutch math)
- Semaphore is a counter
- On semaphore two operations are supported:
 - wait operation: decrement op: P operation:
 - semaphore count is decremented by 1.
 - if $cnt < 0$, then calling process is blocked(block the current process).
 - typically wait operation is performed before accessing the resource.
 - signal operation: increment op: V operation:
 - semaphore count is incremented by 1.
 - if one or more processes are blocked on the semaphore, then wake up one of the process.
 - typically signal operation is performed after releasing the resource.
- Q. If sema count = -n, how many processes are waiting on that semaphore?
 - Answer: "n" processes waiting



+1
Incr/v
←

-1
decre/p
←

Process



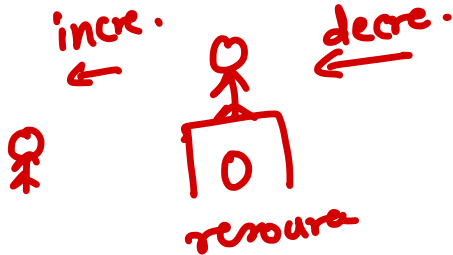
accesses
same resource

multiple process
→ Classic / Count :

Semaphore

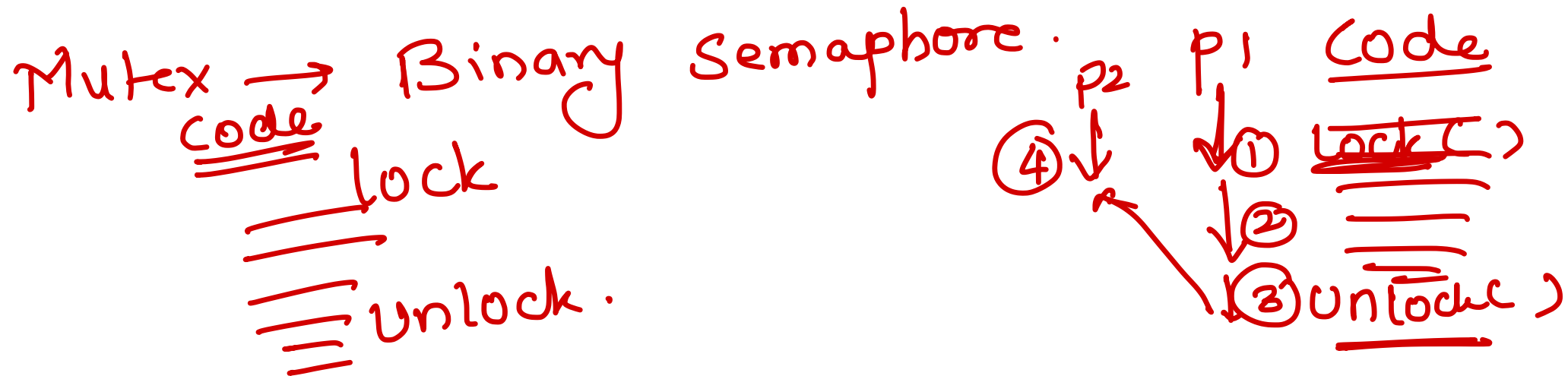
Binary → 1, 0

→ single process access
single resource



Semaphore

- Counting Semaphore/classic
 - When multiple processes can access a resource.
 - Allow "n" number of processes to access resource at a time.
 - Or allow "n" resources to be allocated to the process.
- Binary Semaphore
 - When a single process can access a resource at a time.
 - Allows only 1 process to access resource at a time or used as a flag/condition



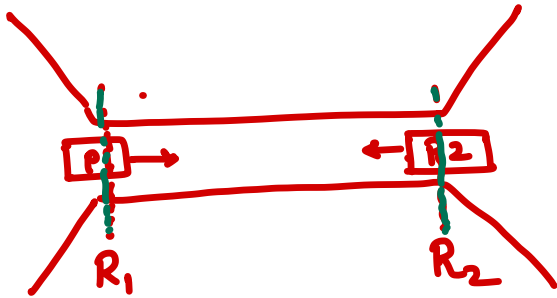
Mutex

Mutex(Mutual Exclusion)

- When only one process access resource like binary semaphore
- Mutex has two states ie. Unlocked or locked state.
- Rule of mutex is that which process has lock the mutex can only unlock the mutex.

Semaphore vs Mutex

- S: Semaphore can be decremented by one process and incremented by same or another process. M: The process locking the mutex is owner of it. Only owner can unlock that mutex.
- S: Semaphore can be counting or binary.
M: Mutex is like binary semaphore. Only two states: locked and unlocked.
- S: Semaphore can be used for counting, mutual exclusion or as a flag.
M: Mutex can be used only for mutual exclusion.



deadlock char .

- ① No preemption.
- ② Mutual Exclusion
- ③ Hold and wait.
- ④ Circular queue

Deadlock

- The processes are blocked indefinitely in deadlock
- Deadlock occurs when four conditions/characteristics hold true at the same time.
 - No preemption: A resource should not be released until task is completed.
 - Mutual exclusion: Resources is not sharable.
 - Hold & Wait: Process holds a resource and wait for another resource.
 - Circular wait: Process P1 holds a resource needed for P2, P2 holds a resource needed for P3 and P3 holds a resource needed for P1.
- **Deadlock Prevention**
 - OS system are designed so that at least one deadlock condition is never true.
 - This will prevent deadlock

Deadlock

Deadlock Avoidance

- The processes should inform system about the resources it needs later. OS will accept or reject resource request based on deadlock possibility.
- Algorithms used for this are:
 - Resource allocation graph: OS maintains graph of resources and processes. A cycle in graph indicate circular wait will occur. In this case OS can deny a resource to a process.
 - Banker's algorithm: A bank always manage its cash so that they can satisfy all customers.
- Deadlock Recovery
 - Deadlock can be solved by resource preemption or terminating one of the process (involved in deadlock).

Resource pre-emption -> forcibly withdraw resources given to the processes.

Process termination -> forcibly kill one of the process.

Deadlock

Starvation vs Deadlock

- Starvation: The process not getting enough CPU time for its execution.
 - Process is in ready state/queue.
Reason: Lower priority (CPU is busy in executing high priority process).
- Deadlock: The process not getting the resource for its execution.
 - Process is in waiting state/queue indefinitely.
Reason: Resource is blocked by another process (and there is circular wait).

Computer Fundamental

Memory

- Memory holds (digital) data or information.
 - Bit = Binary Digit (0 or 1) --> Internally it is an electronic circuit i.e. FlipFlop.
1 Byte = 8 Bits
B, KB (2^{10}), MB (2^{20}), GB (2^{30}), TB (2^{40}), PB (2^{50}), XB (2^{60}), ZB (2^{70})
- Primary memory – Memory
 - Directly accessible by the CPU.
E.g. CPU registers, Cache, RAM.
- Secondary memory -- Storage
Memory accessible via Primary memory. E.g. Disk, CD/DVD, Tape, ROM.
- Volatile vs Non-volatile memory
 - Volatile memory: The contents of memory are lost when power is OFF.
 - Non-volatile memory: The contents of memory are retained even after power is OFF.

Bit \rightarrow 0 or 1

1 Byte \rightarrow 8 bit

KB = 1024 Bytes

MB = 1024 KB

Mem → primary → RAM.
→ Secondary → HDD.

Computer Fundamental

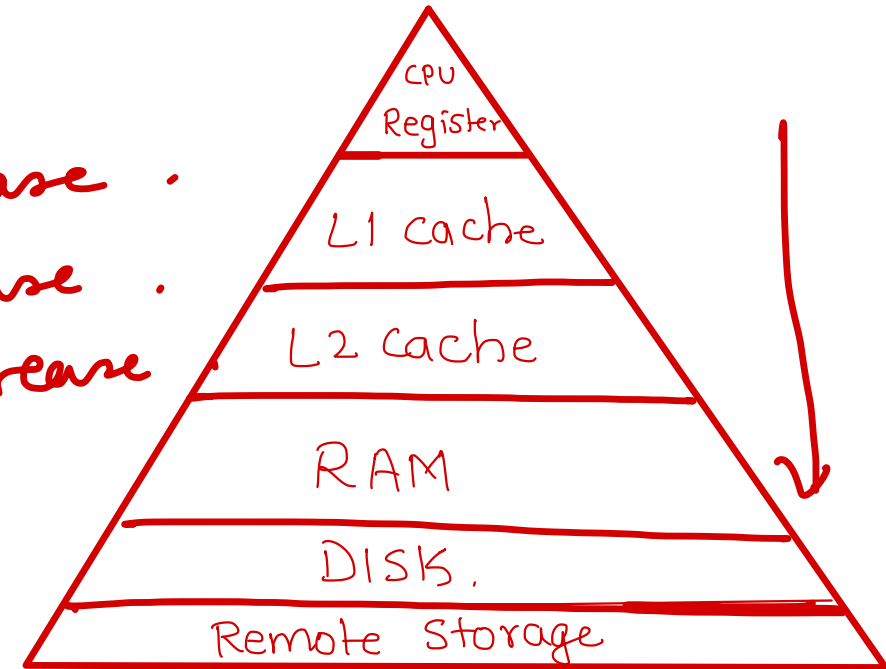
Memory Hierarchy

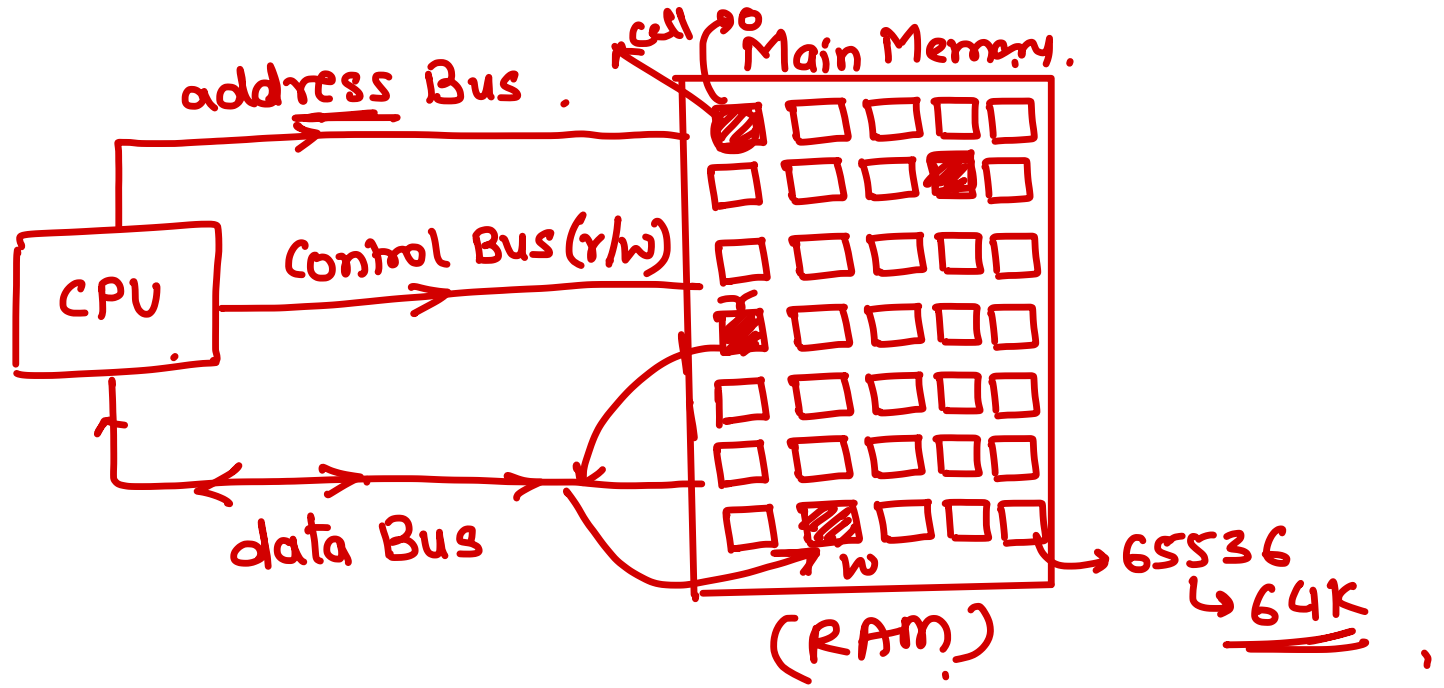
- Level 0: CPU Registers → B.
- Level 1: L1 Cache → KB.
- Level 2: L2 Cache → MB.
- Level 3: RAM → GB.
- Level 4: Disk (Local) → TB.
- Level 5: Remote storage (NFS)

• Comparison

- Capacity: Level 0 is smallest (B) to Level 5 is largest (TB)
- Speed: Level 0 is fastest and Level 5 is slowest
- Cost: Level 0 is costlier and Level 5 is cheaper

cost → decrease .
Size → increase .
Speed → decrease .





Bus → group of wires.

① Sequential access → eg magnetic tape.

② Direct access → eg hard disk

③ Random access → eg RAM

④ Associate mem → key value pair → eg cache

Computer Fundamental

Memory Access

- CPU <----> Memory
- Address bus
 - Unidirectional from CPU to the memory
 - Address represent location of the memory to read/write
 - Number of lines = number of locations
- Data bus
 - Bi-directional from/to CPU to/from the memory
 - Carries the data
 - Number of lines = width of data unit
- Control bus
 - Read/Write operation
- CPU <---> Cache <---> RAM <---> Disk
- Sequential access: Read/write sequentially from start to end. e.g. Magnetic tapes
- Direct access: Read/write to the block address e.g. Hard disk
- Random access: Read/write to the memory (byte) address to e.g. RAM
- Associative access: Read/write to the scan/tag lines(Key –value storage)e.g. Cache

Computer Fundamental

RAM (Random Access Memory)

- RAM is packaged as a chip, Basic storage unit is a cell (one bit per cell)
- Internal memory of the CPU for storing data, program, and program result
- Used for Read/ Write
- Volatile (Temporary Storage)

Static RAM (SRAM)

- Retains its contents as long as power is being supplied.
- Made up of transistors.
- SRAM is more often used for system cache.
- SRAM is faster than DRAM.

Dynamic RAM (DRAM)

- Must be constantly refreshed or it will lose its contents.
- This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second.
- Made up of memory cells composed of capacitors and one transistor.
- DRAM is typically used as the main memory in computers.

