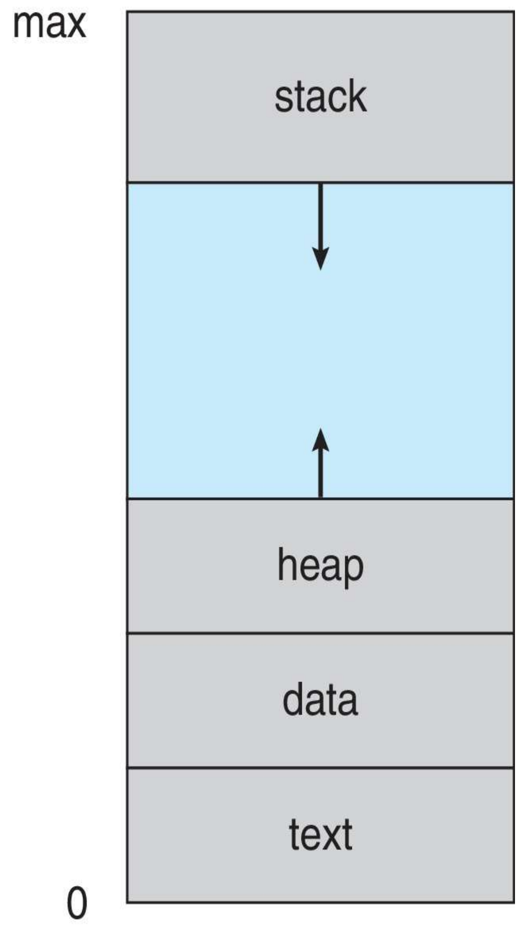
Process concept

- A program in execution
- An instance of a program running on a computer
- A program is a list of instructions stored on secondary memory.
- A program becomes a process when it is loaded into main memory and its PCB is created.

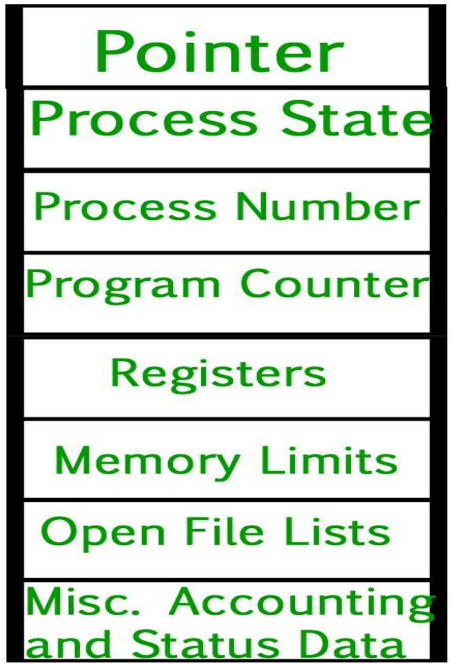
Process in Memory

- **Text Section**: A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the Program Counter. It contains executable code.
- **Stack**: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
- **Data Section**: Contains the global variable.
- **Heap Section**: Dynamically memory allocated to process during its run time.



Process control block

- The process control block (PCB) is used to track the process's execution status.
- When the process makes a transition from one state to another, the operating system must update information in the process's PCB.
- PCB of each process resides in the main memory.
- There exists only one PCB corresponding to each process.



Process Control Block

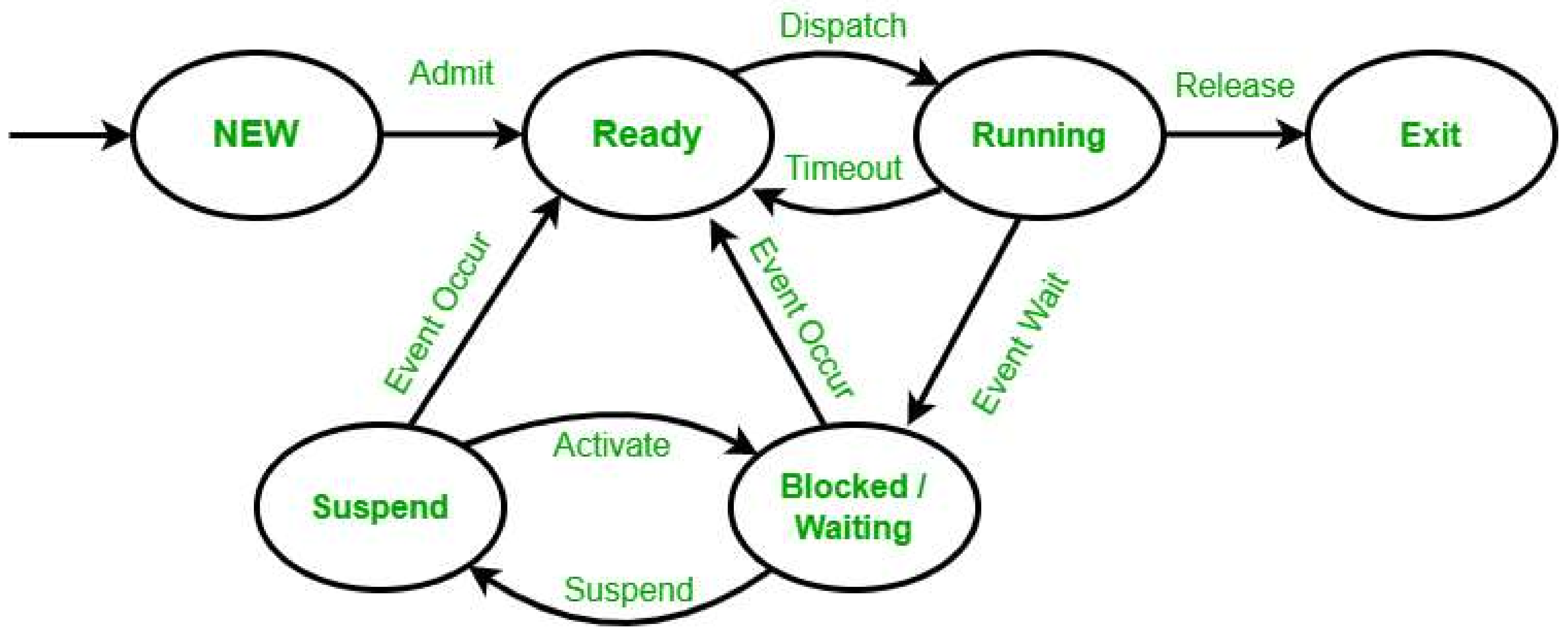
Process control block

- Identifier: A unique identifier associated with this process, to distinguish it from all other processes.
- State: If the process is currently executing, it is in the running state.
- Priority: Priority level relative to other processes.
- Program counter: The address of the next instruction in the program to be executed.
- Pointers: It is a stack pointer that is required to be saved when the process is switched from one state to another to retain the current position of the process..

Process control block

- Register: When a processes is running and it’s time slice expires, the current value of process specific registers would be stored in the PCB and the process would be swapped out. When the process is scheduled to be run, the register values is read from the PCB and written to the CPU registers.
- I/O status information: Includes outstanding I/O requests, I/O devices (e.g., tape drives) assigned to this process, a list of files in use by the process, and so on.
- Accounting information: May include the amount of processor time and clock time used, time limits, and so on.
- Memory limits: This field contains the information about memory management system used by the operating system. This may include page tables, segment tables, etc.

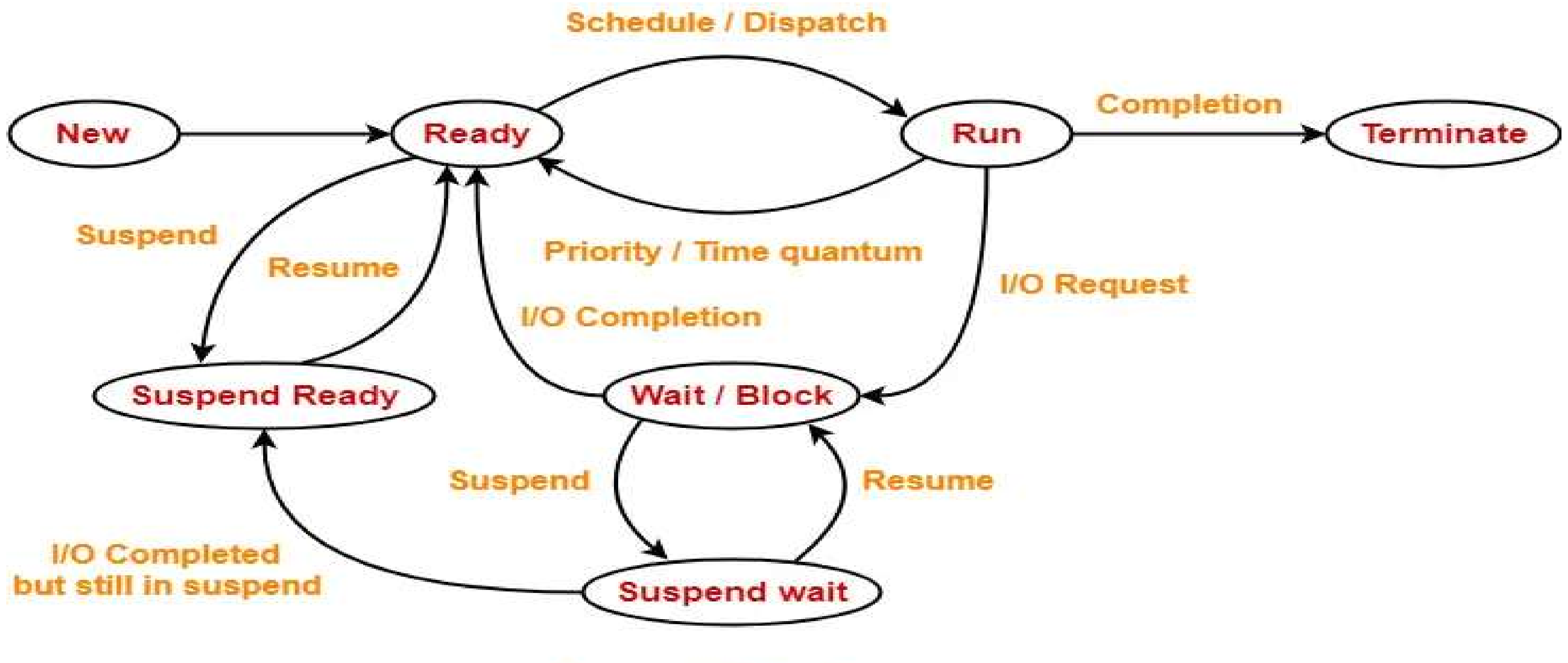
Process states diagram



Process states

- New: A process that has just been created but has not yet been admitted to the pool of executable processes by the OS. Typically, a new process has not yet been loaded into main memory, although its process control block has been created.
- Ready: The process is waiting to be assigned to a processor
- Running: The process that is currently being executed.
- Blocked/Waiting: A process that cannot execute until some event occurs, such as the completion of an I/O operation.
- Exit: A process that has been released from the pool of executable processes by the OS, either because it halted or because it aborted for some reason.

Process states diagram



Process State Diagram

Process states

Suspend Wait State-

- A process moves from wait state to suspend wait state if a process with higher priority has to be executed but the main memory is full.

•After the resource becomes available, the process is moved to the suspend ready state.

•After main memory becomes available, the process is moved to the ready state.

Suspend Ready State-

•A process moves from ready state to suspend ready state if a process with higher priority has to be executed but the main memory is full.

•The process remains in the suspend ready state until the main memory becomes available.

Context Switch

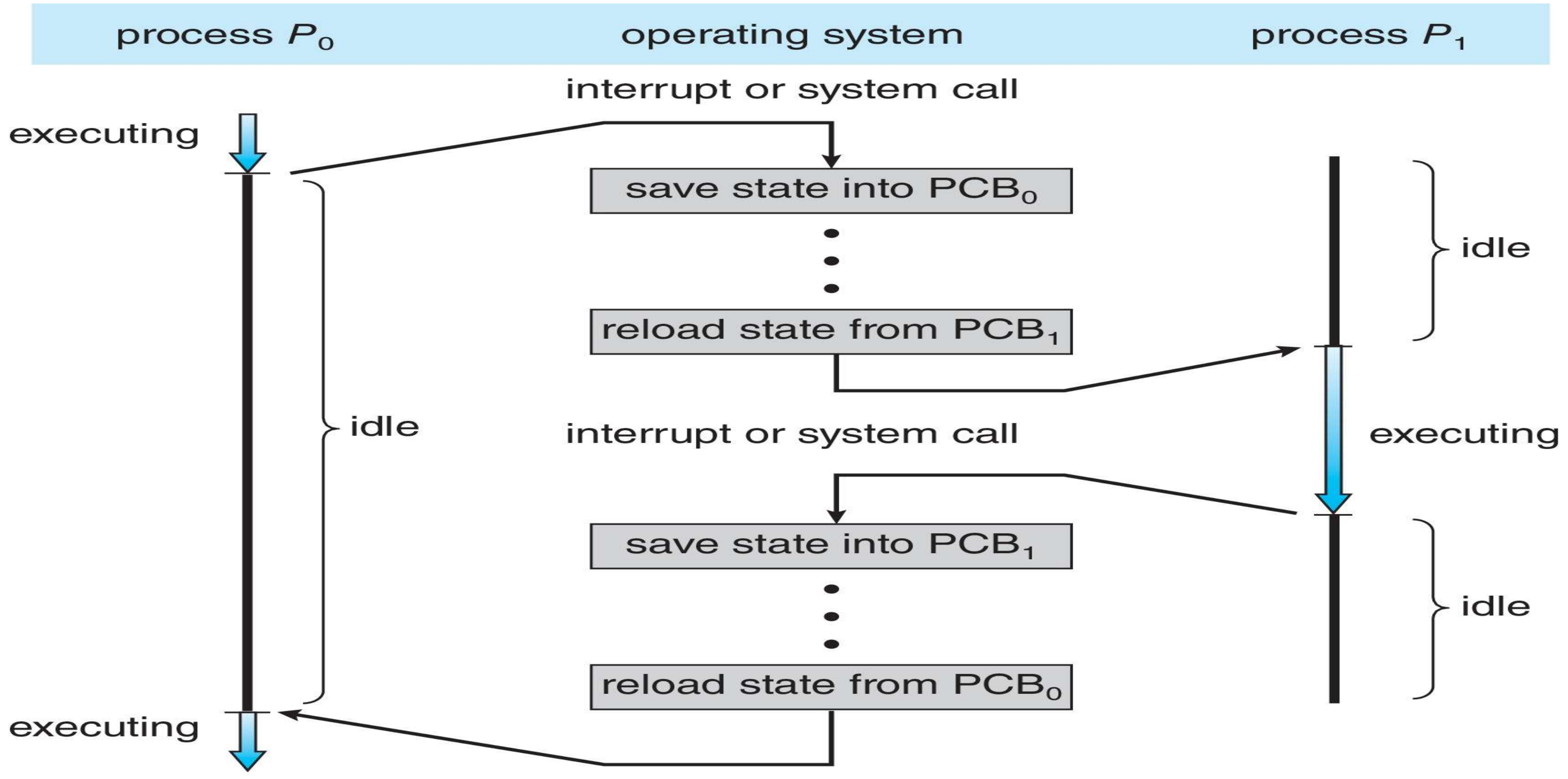
- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB.

Context Switching Happen:

- When a high-priority process comes to a ready state (i.e. with higher priority than the running process).
- An Interrupt occurs.

CPU Switch From Process to Process

A **context switch** occurs when the CPU switches from one process to another.



Operations on Processes

- **Creation**

Once the process is created, it will be ready and come into the ready queue (main memory) and will be ready for the execution.

- **Scheduling**

Out of the many processes present in the ready queue, the Operating system chooses one process and start executing it. Selecting the process which is to be executed next, is known as scheduling.

- **Execution**

Once the process is scheduled for the execution, the processor starts executing it. Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.

- **Deletion/killing**

Once the purpose of the process gets over then the OS will kill the process. The Context of the process (PCB) will be deleted and the process gets terminated by the Operating system.

Process control

Process control is a key function responsible for managing the life cycle of processes. The process control mechanism handles various tasks, including process creation, synchronization, program loading into memory, and process termination. This lecture will cover these critical aspects of process control.

1) <u>Process creation-</u>

- **PCB allocation**
- **Memory allocation**

2) <u>Waiting for Processes</u>

- Sometimes, the process must wait for the process to finish before continuing. This is essential for synchronization.
- While waiting, a process can be in various states:
- **Ready**: Ready to run, waiting for CPU time.
- **Waiting**: Waiting for an event, such as I/O completion or a child process to finish.

Process control

3) **Loading Programs into Memory**

- For a process to execute, its program must be loaded into memory. This involves several steps, including loading the program's instructions and data, setting up stack space, and linking shared libraries.

4) **Process Termination**: A process terminates when it finishes executing or is forcibly stopped by the system or user. When a process terminates, it frees the resources it was using, such as memory and open files.

Types of Termination:

- Normal Termination: Occurs when a process completes its execution naturally, usually by reaching the end of its code or calling the exit() function.
- **Abnormal Termination**: Happens when a process encounters an error or is forcibly killed. Eg: kill command or invalid memory

Process Scheduler

- **Process scheduler** selects among available processes for next execution on CPU core
- Goal -- Maximize CPU use, quickly switch processes onto CPU core
- Maintains **scheduling queues** of processes
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Wait queues** – set of processes waiting for an event (i.e., I/O)
  - Processes migrate among the various queues

Types of process schedulers

- Operating system uses various schedulers for the process scheduling.
- They are:
- Long term scheduler
- Short term scheduler
- Mid term scheduler

Types of process schedulers

- **Long Term or Job Scheduler**
- It brings the new process to the ‘Ready State’.
- It controls the Degree of Multi-programming, i.e., the number of processes present in a ready state at any point in time.
- It selects a balanced mix of I/O bound and CPU bound processes from the secondary memory (new state).

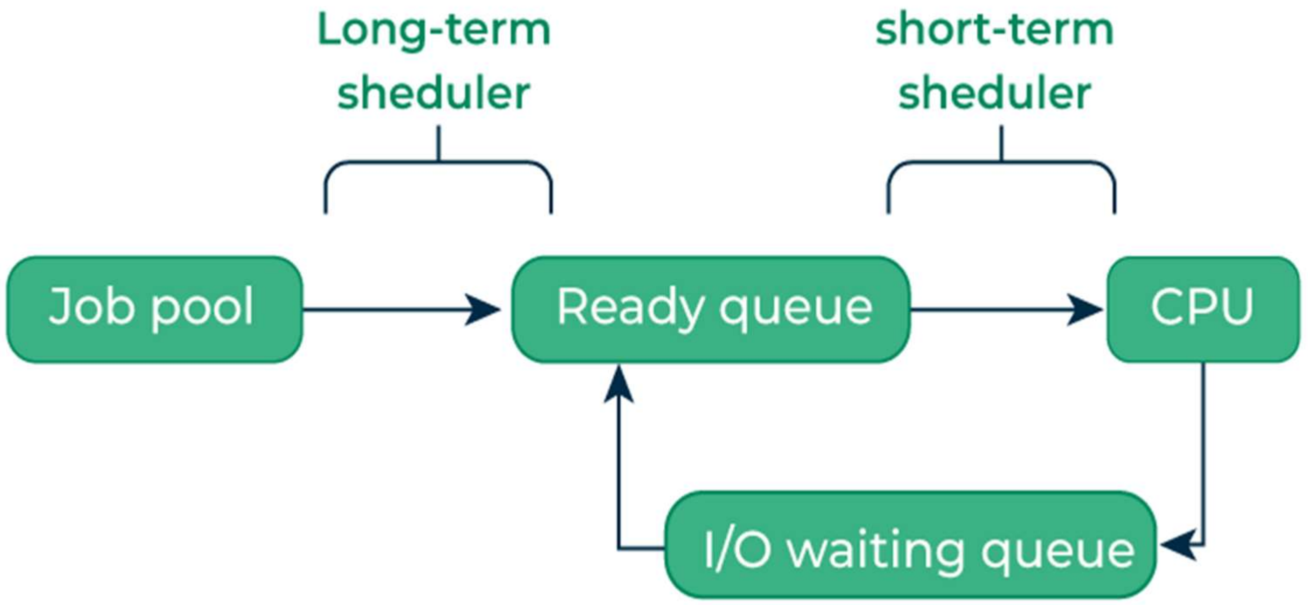
- **Short-Term or CPU Scheduler**
- It is responsible for selecting one process from the ready state for scheduling it on the running state.
- A scheduling algorithm is used to select which job is going to execute.

Types of process schedulers

- **Medium-Term Scheduler**
- It is responsible for suspending and resuming the process.
- The primary objective of medium-term scheduler is to perform swapping.
- Medium-term scheduler swaps-out the processes from main memory to secondary memory to free up the main memory when required.

Types of process schedulers

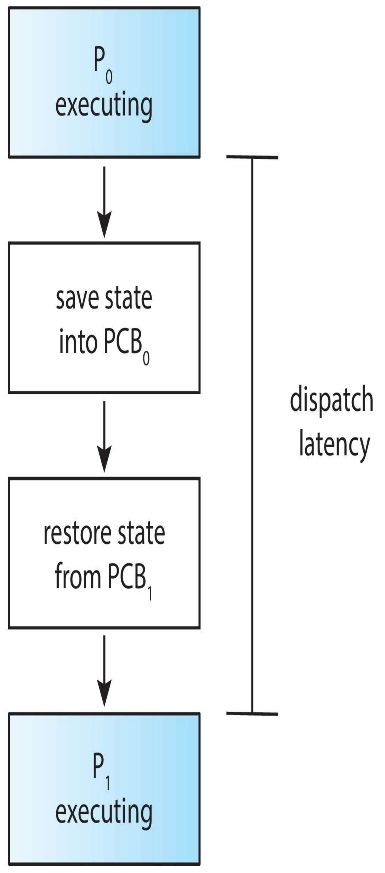
- .



Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the CPU scheduler.

- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running



CPU scheduling

- It is a process of determining which process in the ready queue is allocated to the CPU.
- Different scheduling algorithms like FCFS, SJF, Round Robin, etc.

Process scheduling

- Process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process based on a particular strategy.
- Types of process scheduling:
- Preemptive
- Non preemptive

Process scheduling

- **Non-preemptive:** In this case, a process's resource cannot be taken before the process has finished running. When a running process finishes and transitions to a waiting state, resources are switched.
- Under Non preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases it either by terminating or by switching to the waiting state.

- **Preemptive:** In this case, the OS assigns resources to a process for a predetermined period. The process switches from running state to ready state or from waiting for state to ready state during resource allocation.
- All modern operating systems including Windows, MacOS, Linux, and UNIX use preemptive scheduling algorithms.

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – No. of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced.

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Terminologies

- **Arrival Time:** Time at which the process arrives in the ready queue.
- **Completion Time:** Time at which process completes its execution.
- **Burst Time:** Time required by a process for CPU execution.
- **Turn Around Time:** Completion time-Arrival time

  Or

  Burst Time + Waiting Time
- **Waiting time**: Turn Around Time-Burst time

First Come First Serve(FCFS)

- The process which arrives first in the ready queue is firstly assigned the CPU.
- It is non-preemptive in nature.

Example

- Consider the set of 5 processes whose arrival time and burst time are given below. If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Process Id	Arrival time	Burst time
P1	3	4
P2	5	3
P3	0	2
P4	5	1
P5	4	3

Example

- Consider the set of 3 processes whose arrival time and burst time are given below. If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Process Id	**Arrival time**	**Burst time**
P1	0	2
P2	3	1
P3	5	6

First Come First Serve(FCFS)

Advantages

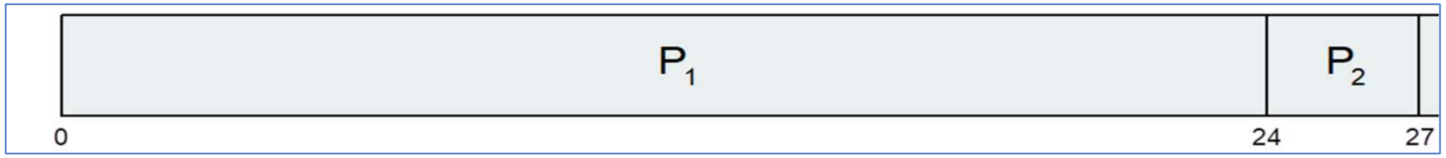
- Easy to implement using queue data structure.
- It is simple and easy to understand.
- Starvation cannot occur.
- **Starvation**- Starvation is a condition where a process does not get the resources it needs for a long time because the resources are being allocated to other processes. It generally occurs in a Priority based scheduling System.

Disadvantages

- It suffers from **convoy effect**.
- It does not consider the priority or burst time of the processes.

Convoy Effect

- When smaller processes have to wait for a long time for longer processes to release the CPU then it is called convoy effect.
- Example-



Shortest Job First(SJF)

- The shortest job first (SJF) policy selects the process with the smallest execution time to execute next.
- It can be preemptive or non-preemptive.

Shortest Remaining Time First(SRTF)

- **Shortest remaining time first** is the preemptive version of the Shortest job.
- SRTF algorithm makes the processing of the jobs faster than SJF algorithm,

Disadvantages

- Like the shortest job first, it may also lead to process starvation.

Example-1

- Consider the set of 5 processes whose arrival time and burst time are given below If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turn around time.

Process Id	**Arrival time**	**Burst time**
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Example-2

- Consider the set of 6 processes whose arrival time and burst time are given below. If the CPU scheduling policy is shortest remaining time first, calculate the average waiting time and average turn around time..

Process Id	Arrival time	Burst time
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	1

Example-3

- Consider the set of 4 processes whose arrival time and burst time are given below. If the CPU scheduling policy is SRTF, calculate the waiting time of process P2.

Process Id	Arrival time	Burst time
P1	0	20
P2	15	25
P3	30	10
P4	45	15

Shortest Job First

Advantages

- It reduces the average waiting time.

Disadvantages

- SJF may cause starvation.
- It is hard to implement because the exact Burst time for a process can't be known in advance.

Round Robin (RR) Scheduling

- This type of scheduling works on Time sharing technique.
- The period of time for which a process is allowed to run in a pre-emptive mode is called time **quantum**.
- Each process gets a small unit of CPU time called as (time quantum *q*).After this time has elapsed, the process is preempted and added to the end of the ready queue.
- Timer interrupts every quantum to schedule next process.
- It is also the preemptive version of the First come First Serve CPU Scheduling algorithm.
- Smaller value of time quantum is better in terms of response time. Smaller time quantum increases no. of context switches and decreases the chance of starvation.
- Larger value of time quantum is better in terms of context switch. Larger time quantum decreases no. of context switches and increases the chance of starvation.
- The performance of Round Robin scheduling heavily depends on the value of time quantum.

Round Robin (RR) Scheduling-1

- Consider the set of 5 processes whose arrival time and burst time are given below. If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turn around time.

Process Id	Arrival time	Burst time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

Round Robin (RR) Scheduling-2

- Consider the set of 6 processes whose arrival time and burst time are given below. If the CPU scheduling policy is Round Robin with time quantum = 3 unit, calculate the average waiting time and average turn around time.

Process Id	**Arrival time**	**Burst time**
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

Round Robin (RR) Scheduling-3

- Consider the set of 6 processes whose arrival time and burst time are given below. If the CPU scheduling policy is Round Robin with time quantum = 2, calculate the average waiting time and average turn around time.

Process Id	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	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
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

Priority Scheduling-1

- Consider the set of 5 processes whose arrival time and burst time are given below. If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

Priority Scheduling-2

- Consider the set of 5 processes whose arrival time and burst time are given below. If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

Priority Scheduling-3

- Consider the set of 7 processes whose arrival time and burst time are given below. If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority) . Solve this with non-preemptive scheduling also.

Process Id	Arrival time	Burst time	Priority
P1	0	1	2
P2	1	7	6
P3	2	3	3
P4	3	6	5
P5	4	5	4
P6	5	15	10
P7	15	8	9

CPU SCHEDULING EXAMPLE

- Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is preemptive shortest remaining-time first. The average turn around time of these processes is __________

Process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

CPU SCHEDULING EXAMPLE

- Consider the set of 4 processes whose arrival time and burst time are given below. If the CPU scheduling policy is Shortest Remaining Time First, calculate the average waiting time and average turn around time.

Process No.	Arrival Time	Burst Time		
		CPU Burst	I/O Burst	CPU Burst
P1	0	3	2	2
P2	0	2	4	1
P3	2	1	3	2
P4	5	2	2	1

CPU SCHEDULING EXAMPLE

- Consider the set of 4 processes whose arrival time and burst time are given below. If the CPU scheduling policy is Priority Scheduling, calculate the average waiting time and average turn around time. (Lower number means higher priority)

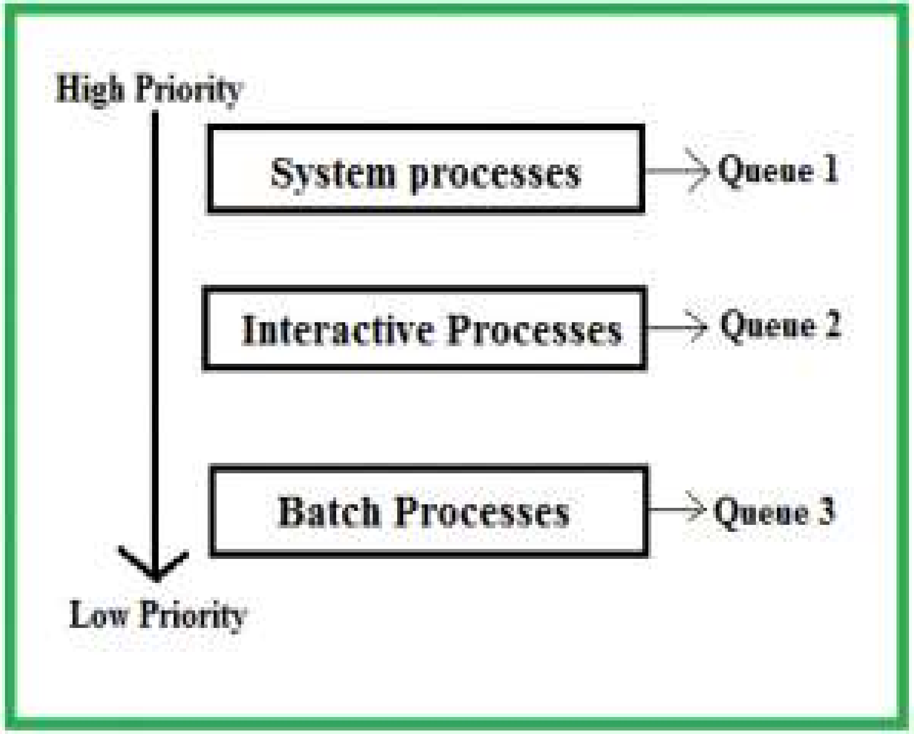
Process No.	**Arrival Time**	**Priority**	**Burst Time**		
			CPU Burst	**I/O Burst**	**CPU Burst**
P1	0	2	1	5	3
P2	2	3	3	3	1
P3	3	1	2	3	1

Multilevel Queue

- The ready queue consists of multiple queues
- Multilevel queue scheduler defined by the following parameters:
  - Number of queues
  - Scheduling algorithms for each queue
  - Scheduling among the queues
- Processes are divided into multiple queues based on their priority, with each queue having a different priority level.
- Different scheduling algorithms can be used for each queue,

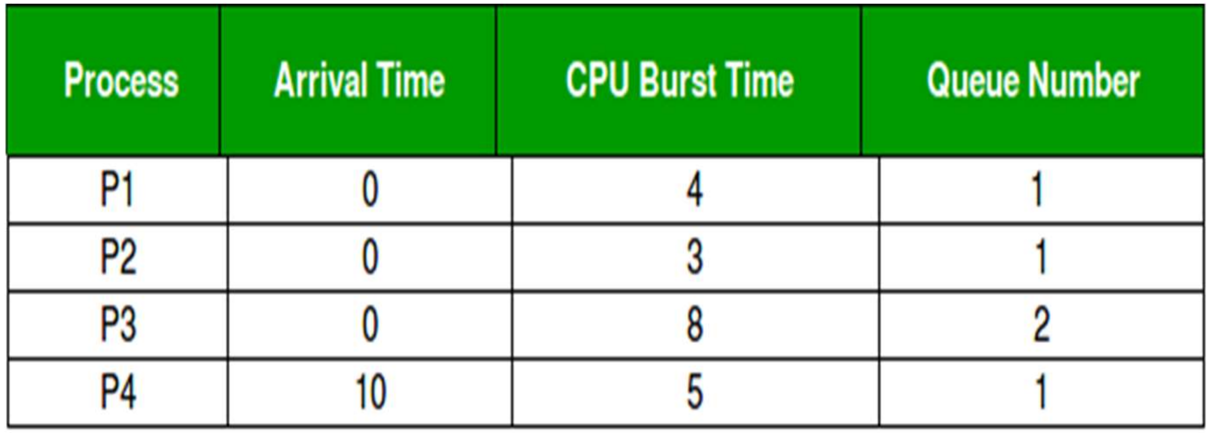
Multilevel Queue

- .



Multilevel Queue-Example

- Consider the below table of four processes under Multilevel queue scheduling. Queue number denotes the queue of the process. Priority of queue 1 is greater than queue 2. queue 1 uses Round Robin (Time Quantum = 2) and queue 2 uses FCFS.

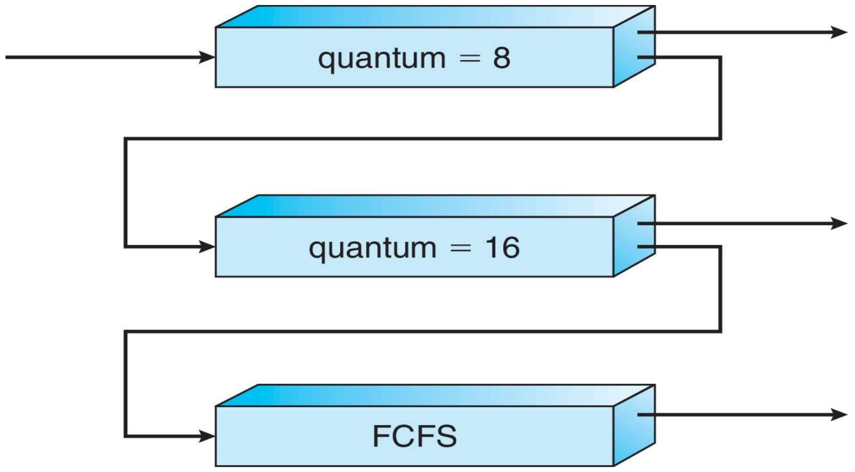


Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Multilevel Feedback Queue

- A process can move between the various queues.
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - Number of queues
  - Scheduling algorithms for each queue
  - Method used to determine when to upgrade a process
  - Method used to determine when to demote a process
  - Method used to determine which queue a process will enter when that process needs service
- Aging can be implemented using multilevel feedback queue

Example of Multilevel Feedback Queue



- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS

- Scheduling
  - A new process enters queue Q_0 which is served in RR
    - When it gains CPU, the process receives 8 milliseconds
    - If it does not finish in 8 milliseconds, the process is moved to queue Q_1
  - At Q_1 job is again served in RR and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q_2