Module 2: CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Round Robin Scheduling
- Priority scheduling
- Multiple queues
- Shortest Job First
- Guaranteed scheduling
- Two- level scheduling
- Preemptive scheduling,
- Multiple-Processor Scheduling

Topics

- Process Management: Process concept, Process State, PCB, Operations on processes, Multithreading-Benefits.
- Process Scheduling: Basic concepts, Preemptive Scheduling, Dispatcher, Scheduling criteria, Scheduling Algorithms (FCFS, SJF, Priority scheduling, Round Robin Scheduling, Multi level queue scheduling, Multi level feedback queue scheduling).
- Inter process communication (Shared memory, message passing, pipes and socket).

PROCESSES

Definitions

PROCESS CONCEPT:

- ✔ Program is a group of instruction whereas process is the activity.
- ✔ A **program** is passive; such as the contents of a file stored on disk.
- ✔ A **process** is active; with program counter and a set of resources associated with it.

A process is an instance of a program in execution.

Attributes held by a process include :

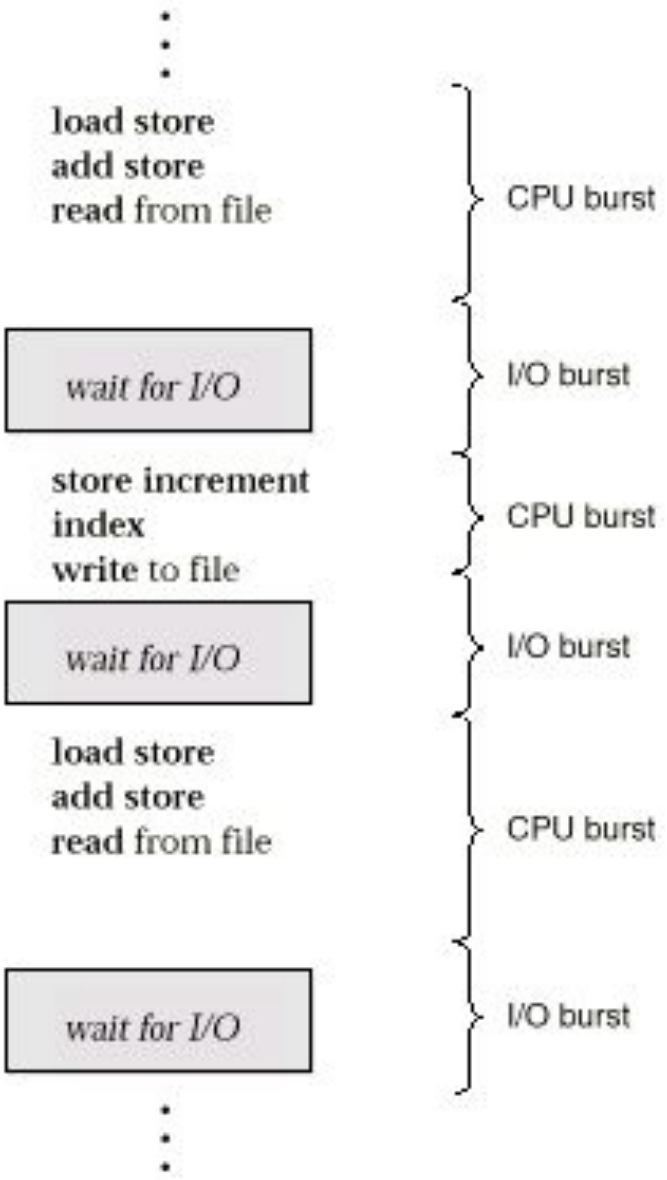
- Hardware state,
- Memory,
- CPU,
- Progress (executing)

WHY HAVE PROCESSES?

- Resource sharing (logical (files) and physical(hardware)
- Computation speedup - taking advantage of multiprogramming
- Modularity for protection.

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a cycle of CPU execution and I/O wait.
- CPU burst distribution

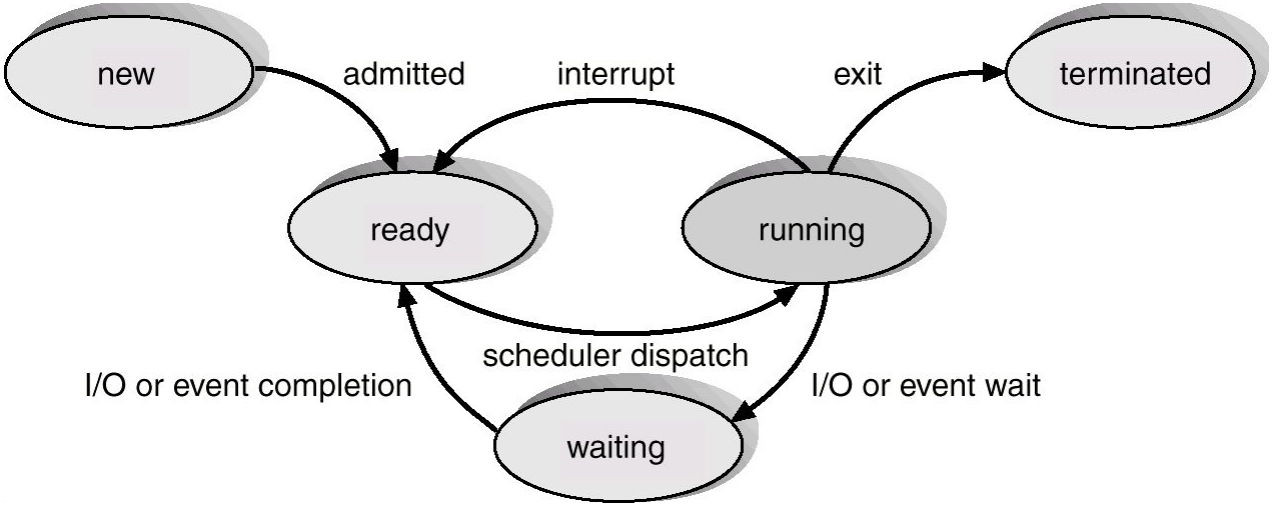
Alternating Sequence of CPU And I/O Bursts



PROCESSES

PROCESS STATES

- **New** The process is just being put together.
- **Running** Instructions being executed. This running process holds the CPU.
- **Waiting** For an event (hardware, human, or another process.)
- **Ready** The process has all needed resources - waiting for CPU only.
- **Suspended** Another process has explicitly told this process to sleep. It will be awakened when a process explicitly awakens it.
- **Terminated** The process is being torn apart.



How a process moves between these states?

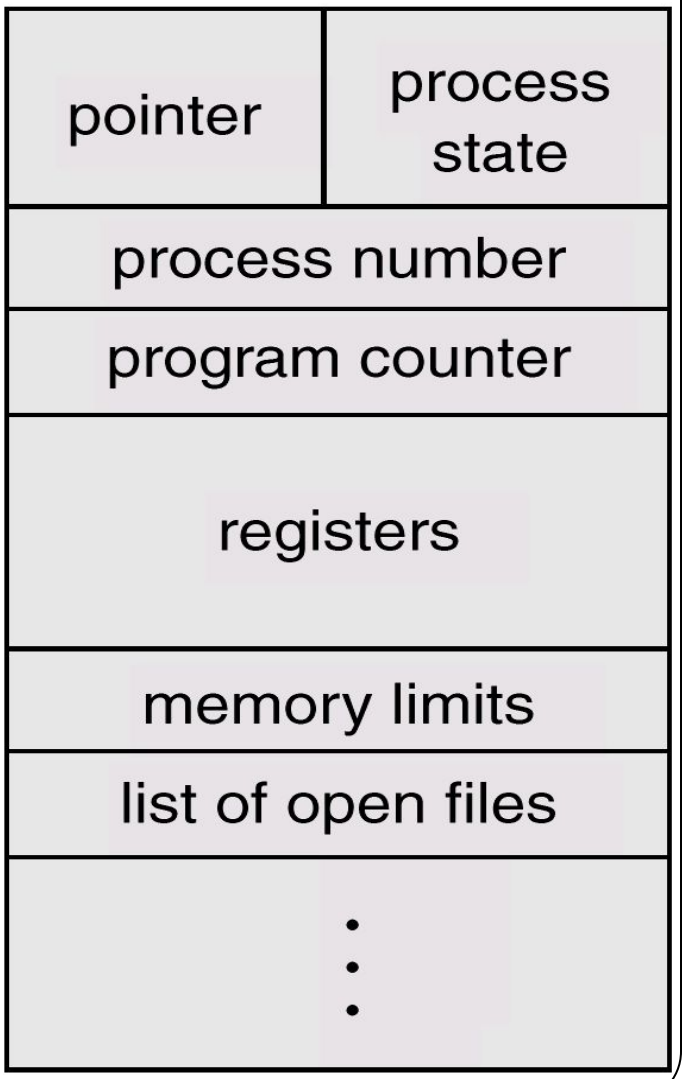
- A process moves from ready to running, when it is dispatched by the scheduler.
- A process moves from running to ready, when it is pre-empted by the scheduler.
- A process moves from running to blocked, when it issues a request for a resource.
- A process moves from blocked to ready, when completion of request is signaled.

Process Control Block (PCB)

Contains information associated with each process:

It's a data structure holding:

- PC, CPU registers,
- memory management information,
- accounting (time used, ID, ...)
- I/O status (such as file resources),
- scheduling data (relative priority, etc.)
- Process State (so running, suspended, etc. is simply a field in the PCB).



Multithreading-Benefits.

- Multithreading is a function of the CPU that permits multiple threads to run independently while sharing the same process resources. A thread is a conscience sequence of instructions that may run in the same parent process as other threads.

- Multithreading allows many parts of a program to run simultaneously. These parts are referred to as threads, and they are lightweight processes that are available within the process. As a result, multithreading increases CPU utilization through multitasking. In multithreading, a computer may execute and process multiple tasks simultaneously.

- Multithreading needs a detailed understanding of these two terms: process and thread. A process is a running program, and a process can also be subdivided into independent units called threads.

- Multithreading allows the execution of multiple parts of a program at the same time.
- These parts are known as threads and are lightweight processes available within the process. So multithreading leads to maximum utilization of the CPU by multitasking.

Benefits of Multithreading

- Responsiveness
- Resource Sharing
- Economy
- Scalability
- Better Communication
- Minimized system resource usage

Disadvantages of Multithreading

- It needs more careful synchronization.
- It can consume a large space of stocks of blocked threads.
- It needs support for thread or process.
- If a parent process has several threads for proper process functioning, the child processes should also be multithreaded because they may be required.
- It imposes context switching overhead.

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state.
  2. Switches from running to ready state.
  3. Switches from waiting to ready.
  4. Terminates.
- Scheduling under 1 and 4 is *nonpreemptive*.
- All other scheduling is *preemptive.*

Scheduling Categories

- CPU Scheduling deals with the problem of deciding which of the processes in ready queue is to be allocated CPU.
- Major division among the several scheduling algorithms is preemptive or non-preemptive discipline.

1. Non-preemptive scheduling

- Once a process has been allocated to the CPU, the CPU cannot be taken away from the process
- Jobs are made to wait by longer jobs, but treatment of all process is fairer.

Algorithms used are:

- First Come First Served (FCFS), Shortest Job First (SJFS)

2. Preemptive scheduling

- Preemption means OS moves a process from running to ready without the process requesting it.
- Algorithm switches to the processing of a new request before completing the processing of the current request.
- Preempted request is put back to the pending request list and resumed again when it get rescheduled.
- Preemptive scheduling is useful in high priority process which requires immediate response.
- Algorithms used are:
- Round Robin Scheduling, Priority based scheduling, Shortest Remaining Time Next scheduling (SRTN)

The decision whether to schedule preemptive or not, depends on the environment and type of application most likely to be supported by OS.

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context : save the process states and register contents in PCB and reload contents of next program
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running.
- Context switch: switching of CPU from one process to another

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process; interval from time of submission to time of completion
- 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, **not** output (for time-sharing environment)

Optimization Criteria

✔ Max CPU utilization ✓ Max throughput

✔ Min turnaround time ✓ Min waiting time

✔ Min response time

$$\text{CPU utilization} = \frac{\text{(processor busy time)}}{\text{(processor busy time + processor idle time)}}$$

Throughput = (no. of process completed)/(time unit)

Turnaround time = t(process completed) – t(process submitted)

Waiting time = Turnaround time – Processing time

Response time = t(first response) – t(submission of request)

1. LONG TERM SCHEDULER: Job scheduler

- Selects processes from disk (spool)
- Run seldom (when job comes into memory)
- Controls degree of multiprogramming
- Tries to balance arrival and departure rate through an appropriate job mix (I/O bound, CPU bound).

SHORT TERM SCHEDULER

Called CPU scheduler

Selects process from memory (ready queue) for execution

Contains three functions:

- Code to remove a process from the processor at the end of its run.
  a)Process may go to ready queue or to a wait state.

- Code to put a process on the ready queue –
  a)Process must be ready to run.
  b)Process placed on queue based on priority.

- Code to take a process off the ready queue and run that process (also called **dispatcher**).
  a) Always takes the first process on the queue (no intelligence required)
  b) Places the process on the processor.

This code runs frequently and so should be as short as possible

MEDIUM TERM SCHEDULER

- Mixture of CPU and memory resource management.
- Swap out/in jobs to improve mix and to get memory.
- Controls change of priority.

Categories of Scheduling Algorithms

- Batch
- Interactive
- Real time

Scheduling Algorithm Goals

All systems

- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems

- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems

- Response time - respond to requests quickly
- Proportionality - meet users' expectations

Real-time systems

- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems

Some goals of the scheduling algorithm under different circumstances.

Scheduling in Batch Systems

- First-come first-served
- Shortest job first
- Shortest remaining Time next

First-Come, First-Served (FCFS) Scheduling

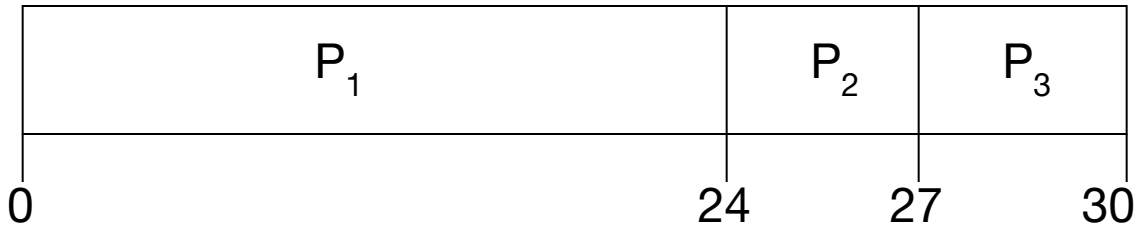
- Example: Process Burst Time

  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 for the schedule is:



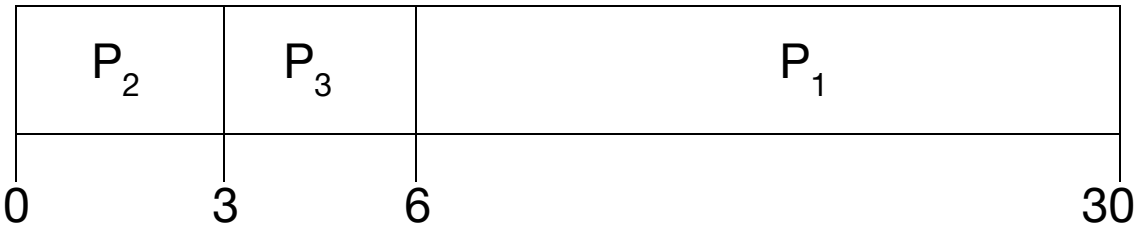
- Waiting time and response time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17
- Turnaround time of P1 = 24; P2 = 27; P3 = 30

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

P_2 , P_3 , P_1 .

- The Gantt chart for the schedule is:



- Waiting time and response time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Turnaround time of P1 = 3, P2 = 6; P3 = 30
- Much better than previous case.

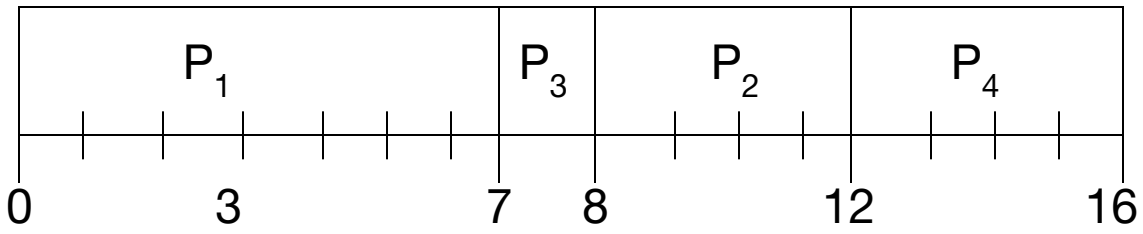
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
  - Preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF).
- SJF is optimal – gives minimum average waiting time for a given set of processes.

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)
- Waiting time and response time of P1 = 0; P2 = 8 – 2 = 6; P3 = 7 – 4 = 3



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Turnaround time = Completion time – arrival time

- Turnaround time (P1) = 7 - 0 = 7
- Turnaround time (P2) = 12 – 2 = 10
- Turnaround time (P3) = 8 – 4 = 4
- Turnaround time (P4) = 16 – 5 = 11

Scheduling in Interactive Systems

- Round-robin scheduling
- Priority scheduling
- Multiple queues
- Shortest process next
- Guaranteed scheduling
- Lottery scheduling
- Fair-share scheduling

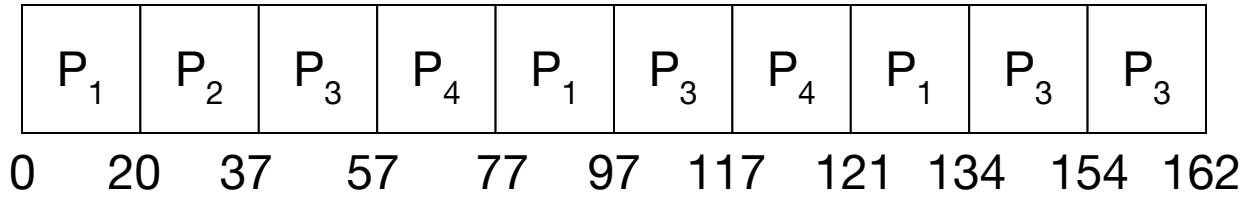
Round Robin (RR)

- 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.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high.

Example: RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better *response*.
- Assume all process arrived at time zero

Turnaround time = Completion time – arrival time

- Turnaround time (P1) = 134 – 0 = 134
- Turnaround time (P2) = 37 – 0 = 37
- Turnaround time (P3) = 162 – 0 = 162
- Turnaround time (P4) = 121 – 0 = 121

Waiting time = Turnaround time – Burst Time

- Waiting time (P1) = 134 – 53 = 81
- Waiting time (P2) = 37 – 17 = 20
- Waiting time (P3) = 162 – 68 = 94

Response Time = First execution start time – arrival time

- Response Time (P1) = 0 – 0 = 0
- Response Time (P2) = 20 – 0 =20
- Response Time (P3) = 37 – 0 = 37
- Response Time (P4) = 57 – 0 = 57

What is Preemptive Scheduling?

- Preemptive Scheduling is a scheduling method where the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running.

- At that time, the lower priority task holds for some time and resumes when the higher priority task finishes its execution.

What is Non- Preemptive Scheduling?

- In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating.

- It is the only method that can be used for various hardware platforms. That's because it doesn't need specialized hardware (for example, a timer) like preemptive Scheduling.

- Non-Preemptive Scheduling occurs when a process voluntarily enters the wait state or terminates.

Advantages of Preemptive Scheduling

- Preemptive scheduling method is more robust, approach so one process cannot monopolize the CPU
- Choice of running task reconsidered after each interruption.
- Each event cause interruption of running tasks
- The OS makes sure that CPU usage is the same by all running process.
- In this, the usage of CPU is the same, i.e., all the running processes will make use of CPU equally.
- This scheduling method also improvises the average response time.

Advantages of Non-preemptive Scheduling

- Offers low scheduling overhead
- Tends to offer high throughput
- It is conceptually very simple method
- Less computational resources need for Scheduling

Disadvantages of Preemptive Scheduling

- Need limited computational resources for Scheduling
- Takes a higher time by the scheduler to suspend the running task, switch the context, and dispatch the new incoming task.
- The process which has low priority needs to wait for a longer time if some high priority processes arrive continuously

Disadvantages of Non-Preemptive Scheduling

- It can lead to starvation especially for those real-time tasks
- Bugs can cause a machine to freeze up
- It can make real-time and priority Scheduling difficult
- Poor response time for processes