OPERATING SYSTEMS

Process Management 5

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 1

UNIT 1: Introduction and Process Management

Operating-System Structure & Operations, Kernel Data Structures,Computing Environments, Operating-System Services, OperatingSystem Design and Implementation. Process concept: Process in memory, Process State, Process Control Block, Process Creation and Termination, CPU Scheduling and Scheduling Algorithms, IPC - Shared Memory & Message Passing, Pipes - Named and Ordinary. Case Study: Linux/Windows Scheduling Policies.

OPERATING SYSTEMS

Course Outline

Class No.	Chapter Title / Reference Literature	Topics to be covered	% of Portions covered	
			Reference chapter	Cumulative
1	1.1-1.2	What Operating Systems Do, Computer-System Organization?	1	21.4
2	1.3,1.4,1.5	Computer-System Architecture, Operating-System Structure & Operations	1	
3	1.10,1.11	Kernel Data Structures, Computing Environments	1	
4	2.1,2.6	Operating-System Services, Operating System Design and Implementation	2	
5	3.1-3.3	Process concept: Process in memory, Process State, Process Control Block, Process Creation and Termination	3	
6	5.1-5.2	CPU Scheduling: Basic Concepts, Scheduling Criteria	5	
7	5.3	Scheduling Algorithms: First-Come, First-Served Scheduling, Shortest-Job-First Scheduling	5	
8	5.3	Scheduling Algorithms: Shortest-Job-First Scheduling (Pre-emptive), Priority Scheduling	5	
9	5.3	Round-Robin Scheduling, Multi-level Queue, Multi-Level Feedback Queue Scheduling	5	
10	5.5,5.6	Multiple-Processor Scheduling, Real-Time CPU Scheduling	5	
11	5.7	Case Study: Linux/Windows Scheduling Policies	5	
12	3.4,3.6.3	IPC – Shared Memory & Message Passing, Pipes – Named and Ordinary	3,6	

Topics Outline

- Scheduling Algorithms
- Non-Preemptive Priority Scheduling
- Shortest Job First / Next Preemptive Scheduling aka Shortest Remaining Time First Scheduling

CPU Scheduling: 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.
- **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 typically used for time-sharing environment

CPU Scheduling Algorithm Optimisation Criteria

- Maximize CPU utilization
- Maximize Throughput
- Minimize TurnAround Time
- Minimize Waiting Time
- Minimize Response Time

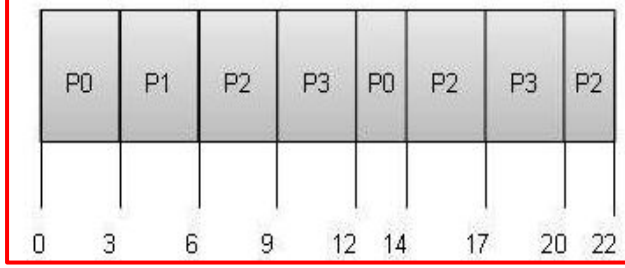
Sample Schemas and Tools for solving Scheduling Problems

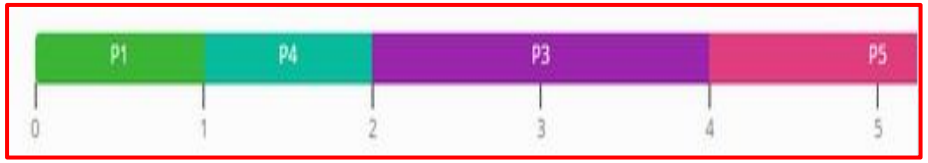
Process	Arrival Time	Execute Time	Service Time

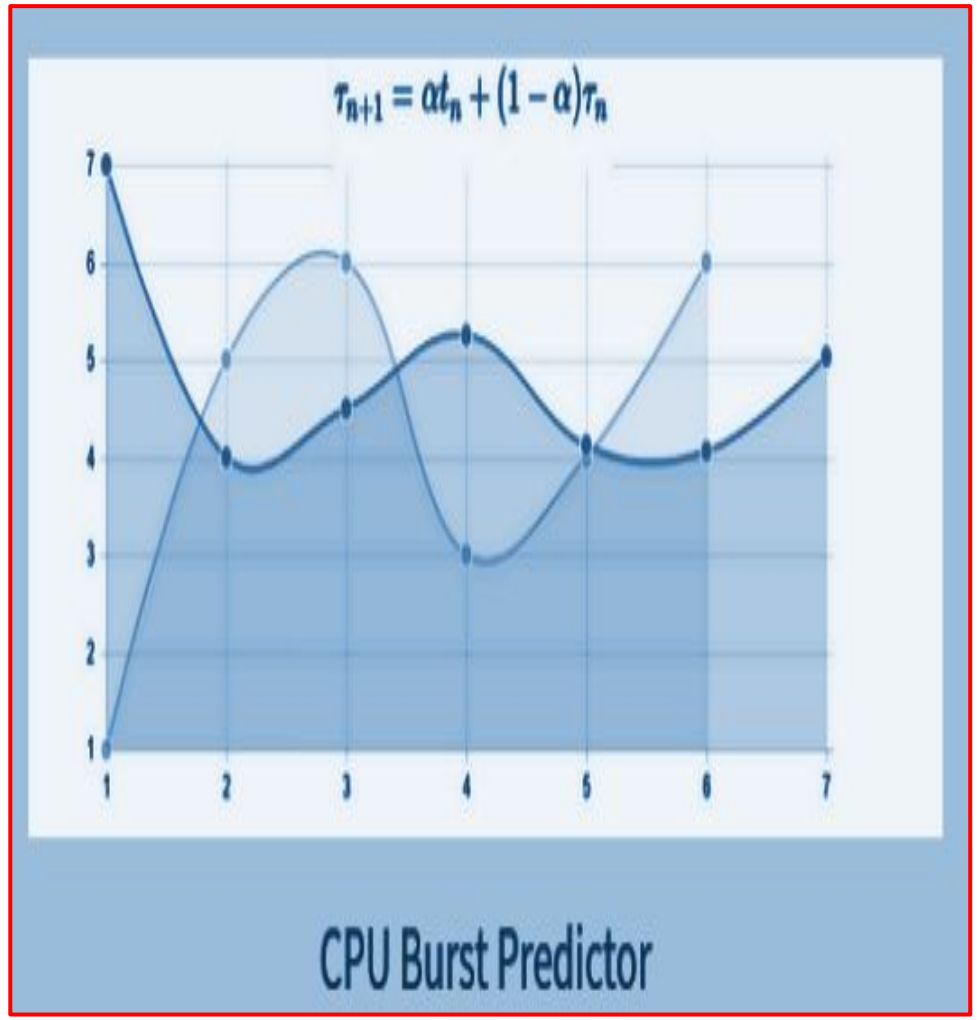
Process	Execution time	Arrival time

Process	Arrival Time	Execution Time	Priority	Service Time

GANTT Chart







Sample Schemas and Tools for solving Scheduling Problems

- **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 (TAT)** : Time Difference between completion time and arrival time.
  - **Turn Around Time** = **Completion Time** – **Arrival Time**
- **Waiting Time(W.T)**: Time Difference between turn around time and burst time.
  - **Waiting Time** = **Turnaround Time** – **Burst Time**

Non-Preemptive Priority Scheduling

Characteristics of Non-Preemptive Priority Scheduling

A CPU algorithm that schedules processes based on priority.

- In **Non-Preemptive Priority** scheduling, a **number** is assigned to each process that indicates its priority level.
- **Lower** the **number**, **higher** is the **priority**. (in **Unix**), **in Windows it is the other way)**
- **Non-Preemptive Priority Scheduling** is another type of technique which is commonly used to schedule processes in **batch operating systems**.
- In Non-Preemptive Priority Scheduling there is a priority assigned to each process and processes are executed according to their priority and since it is non-preemptive so a process can't be preempted by another process in the midst of execution of a process

Non-Preemptive Priority Scheduling

Characteristics of Non-Preemptive Priority Scheduling

A CPU algorithm that schedules processes based on priority.

- In this type of scheduling algorithm, if a newer process arrives, that is having a **higher priority** than the currently running process, then the currently running process will be allowed to complete its execution, then new decision is taken
- At the time of **taking** decision based , If two jobs having the **same priority** are ready, it is recommended to schedule on a **SJF/SJN**, even in that in case of **contention**, **FIFO** is used, if contention still exist then scheduling based on **Process ID** method is recommended
- However, in case of contention, one has to explain how the contention is resolved if one is following other non-standard method

Non-Preemptive Priority Scheduling

Advantages of **Non-Preemptive Priority Scheduling**

- Easy to use scheduling method
- Processes are executed on the basis of priority so high priority does not need to wait for long which saves time
- This method provides a good mechanism where the relative important of each process may be precisely defined.
- Suitable for applications with fluctuating time and resource requirements.

Disadvantages of **Non-Preemptive Priority Scheduling**

- If the system eventually crashes, all low priority processes get lost.
- If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinite time.
- This scheduling algorithm may leave some low priority processes waiting indefinitely.
- A process will be blocked when it is ready to run but has to wait for the CPU because some other process is running currently.
- If a new higher priority process keeps on coming in the ready queue, then the lower priority process which is in the waiting state may need to wait for a long duration of time.

Non-Preemptive Priority Scheduling

PES UNIVERSITY ONLINE

Example

Preemptive Shortest Remaining First Scheduling / SJF

Characteristics of Preemptive - Shortest Remaining First Scheduling / SJF

- A **preemptive** SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst.

- Preemptive SJF scheduling is sometimes called **shortest-remaining-time-first** scheduling.

Preemptive Shortest Remaining First Scheduling / SJF

Advantages of Preemptive Shortest Remaining Time Scheduling (**SRTF**)

- SRTF is optimal and guarantees the minimum average waiting time.
- It provides a standard for other algorithms since no other algorithm performs better than it.

Disadvantages of Preemptive Shortest Remaining Time Scheduling (**SRTF**)

- It cannot be implemented practically since burst time of the processes cannot be known in advance.
- It leads to starvation for processes with larger burst time.
- Priorities cannot be set for the processes.
- Processes with larger burst time have poor response time.

Preemptive Shortest Remaining First Scheduling / SJF

Example

Topics Outline

- Scheduling Algorithms
  - Non-Preemptive Priority Scheduling
  - Shortest Job First / Next Preemptive Scheduling aka Shortest Remaining Time First Scheduling

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com