OPERATING SYSTEMS

Process Management 4

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
  - First-Come, First-Served Scheduling
  - Shortest Job First / Next Non-Preemptive 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

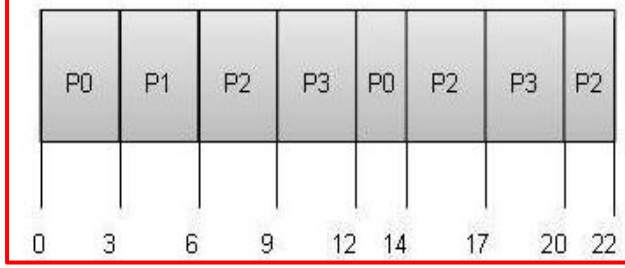
Sample Schemas and Tools for solving Scheduling Problems

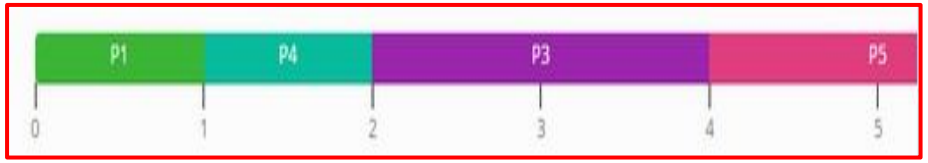
Process	Arrival Time	Execute Time	Service Time

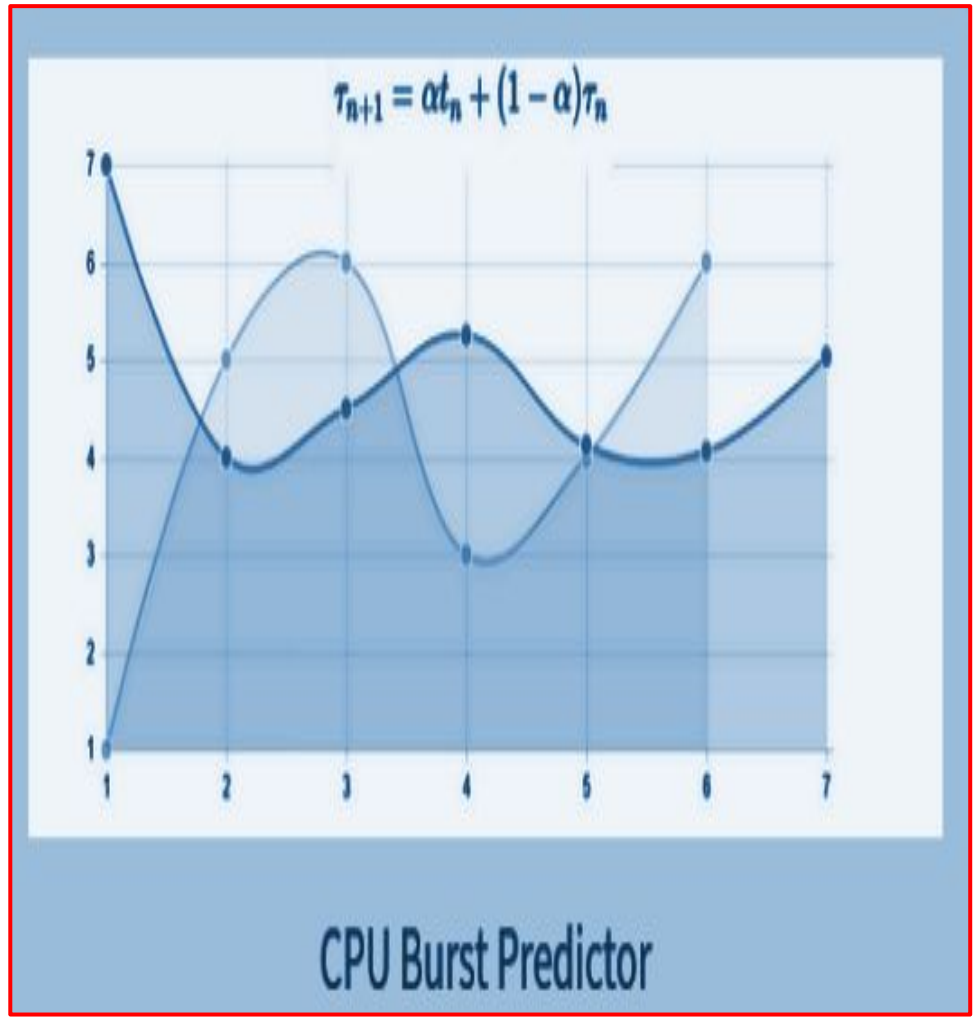
Process	Execution time	Arrival time

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: 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.

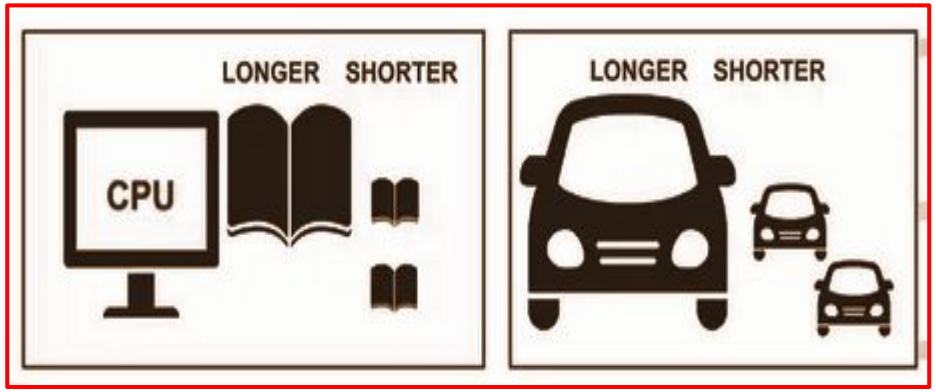
Waiting Time = Turnaround Time – Burst Time

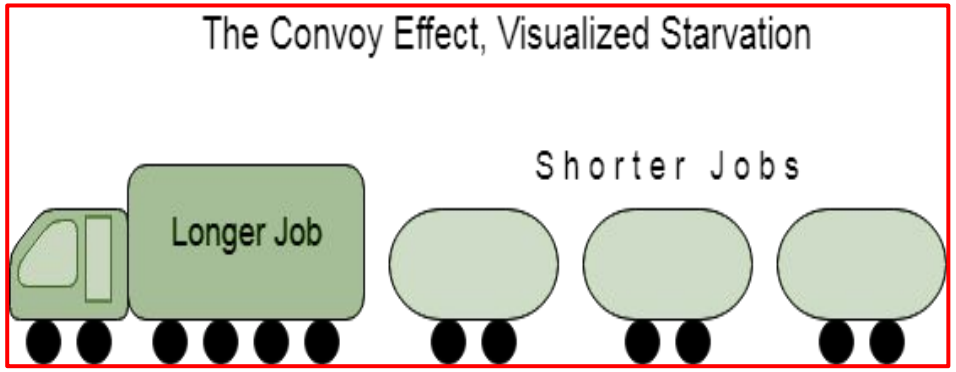
First-Come, First-Served Scheduling

- The simplest CPU -scheduling algorithm is the **First-come, First-served aka FCFS** scheduling algorithm.
- The implementation of the **FCFS** policy is easily managed with a **FIFO queue**.
- When a process enters the ready queue, its **PCB** is linked onto the **tail** of the queue.
- When the **CPU** is **free**, it is allocated to the process at the **head** of the queue.
- The running process is then **removed** from the queue. The code for FCFS scheduling is simple to write and understand.
- On the flip side, the average waiting time under the FCFS policy is often quite long.
- It is inherently **Non-Preemptive**

First-Come, First-Served Scheduling

- There is a **convoy effect** as all the other processes wait for the one big process to get off the CPU .
- This effect results in **lower CPU and device utilization** than might be possible if the shorter processes were allowed to go first
- Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU , either by terminating or by requesting I/O .
- The FCFS algorithm is thus particularly a **disadvantage** for **time-sharing systems**
- It is **never** recommended to allow one process to keep the **CPU** for an **extended** period





OPERATING SYSTEMS - UE19CS254_FCFS_01

Using the below given Process Table, applying First Come First First Serve (FCFS) Scheduling Algorithm. Find the following by drawing appropriate GANTT Chart

1. Waiting Time (WT) for each process
2. Turn Around Time (TAT) for each process
3. Response Time (RT) for each process
4. Average Response Time (ART)
5. Average Waiting Time (AWT)
6. Average Turnaround Time (ATAT)

Note: FCFS algorithm is inherently non-preemptive, the Response time depends on the arrival time,and the size of the job. In this case the larger process arriving earlier makes the smaller process to wait till the larger process completes its turn. It also results in Convoy effect where in the shortest is behind the longer process

GANTT Chart

Process Table			
Process ID	Arrival Time	CPU Burst	Remarks
P0	0	129	
P1	87	112	
P2	165	178	
P3	308	194	
P4	14	167	
Considering Shortest Job First since the arrival time is same			

Process ID	Response Time	Waiting Time	Turn Around Time
P0			
P1			
P2			
P3			
P4			
Average			
Remarks	ART	AWT	ATAT

Shortest Job First / Next Non-Preemptive Scheduling

- It is associated with each job as a unit of time to complete.
- This **algorithm** method is helpful for **batch-type** processing, where waiting for jobs to complete is not critical.
- It can **improve** process **throughput** by making sure that shorter jobs are executed first, hence possibly have a short turnaround time.
- It **improves** job output by offering **shorter** jobs, which should be executed first, which mostly have a **shorter** turnaround time.
- In any non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminated

OPERATING SYSTEMS - UE19CS254_SJF_01

Using the below given Process Table, applying Shortest Job First / Next (SJF / SJN) non-preemptive Scheduling Algorithm. Find the following by drawing appropriate GANTT Chart

1. Waiting Time (WT) for each process
2. Turn Around Time (TAT) for each process
3. Response Time (RT) for each process
4. Average Response Time (ART)
5. Average Waiting Time (AWT)
6. Average Turnaround Time (ATAT)

Homework Problem

GANTT Chart

Process Table			
Process ID	Arrival Time	CPU Burst	Remarks
P0	0	31	
P1	0	33	
P2	0	12	
P3	0	1	
P4	0	14	

Process ID	Response Time	Waiting Time	Turn Around Time
P0			
P1			
P2			
P3			
P4			
Average			
Remarks	ART	AWT	ATAT

OPERATING SYSTEMS - UE19CS254_SJF_01

Using the below given Process Table, applying Shortest Job First / Next (SJF / SJN) non-preemptive Scheduling Algorithm. Find the following by drawing appropriate GANTT Chart

1. Waiting Time (WT) for each process
2. Turn Around Time (TAT) for each process
3. Response Time (RT) for each process
4. Average Response Time (ART)
5. Average Waiting Time (AWT)
6. Average Turnaround Time (ATAT)

Homework Problem

GANTT Chart

Process Table			
Process ID	Arrival Time	CPU Burst	Remarks
P0	77	11	
P1	83	1	
P2	108	9	
P3	117	1	
P4	114	32	

Process ID	Response Time	Waiting Time	Turn Around Time
P0			
P1			
P2			
P3			
P4			
Average			
Remarks	ART	AWT	ATAT

Shortest Job First / Next Non-Preemptive Scheduling

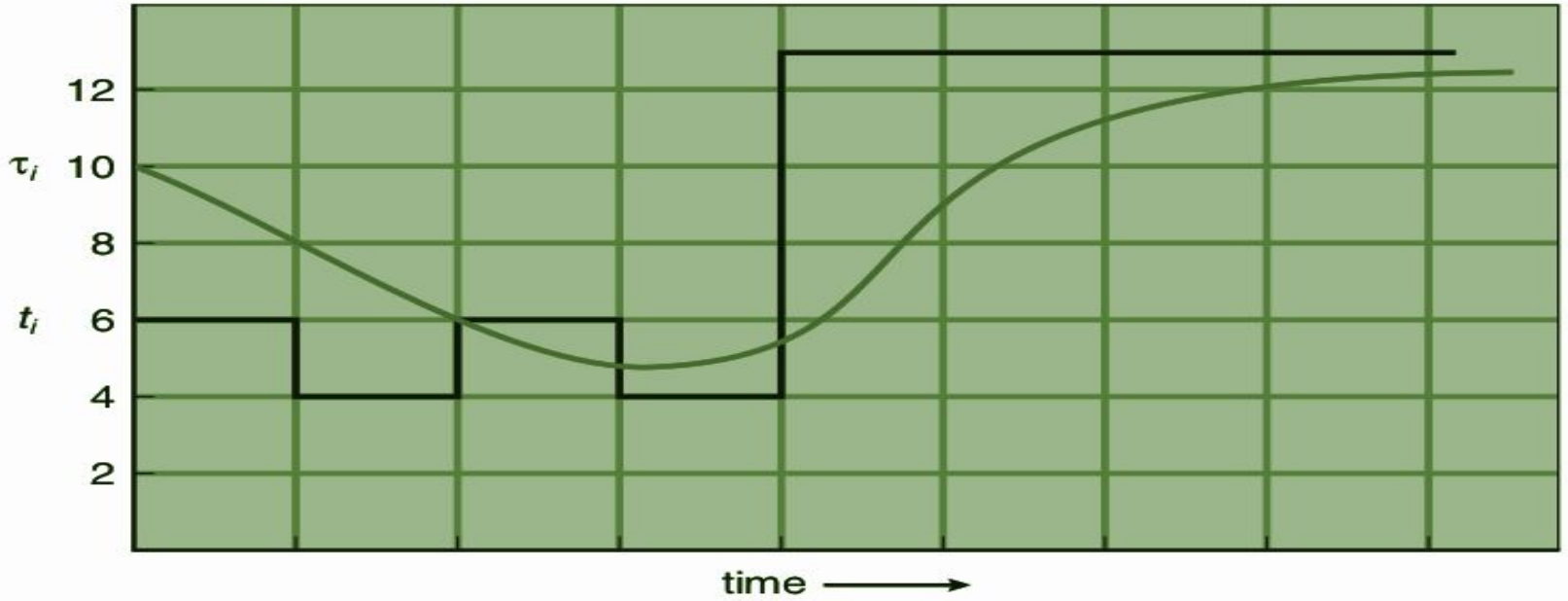
Advantages of SJF / SJN

- SJF is frequently used for long term scheduling.
- It reduces the average waiting time over FIFO (First in First Out) algorithm.
- SJF method gives the lowest average waiting time for a specific set of processes.
- It is appropriate for the jobs running in batch, where run times are known in advance.
- For the batch system of long-term scheduling, a burst time estimate can be obtained from the job description.
- For Short-Term Scheduling, we need to predict the value of the next burst time.
- Probably optimal with regard to average turnaround time.

Disadvantages of SJF / SJN

- Job completion time must be known earlier, but it is hard to **predict**.
- It is often used in a batch system for long term scheduling.
- SJF can't be implemented for CPU scheduling for the short term. It is because there is no specific method to predict the length of the upcoming CPU burst.
- This algorithm may cause very long turnaround times or starvation.
- Requires knowledge of how long a process or job will run.
- It leads to the starvation that does not reduce average turnaround time.
- It is hard to know the length of the upcoming CPU request.
- Elapsed time should be recorded, that results in more overhead on the processor.

Exponential average to predict the length of next CPU Burst



CPU burst (t_i)		6	4	6	4	13	13	13 ...
"guess" (τ_i)	10	8	6	6	5	9	11	12 ...

Prediction of the length of the next CPU burst.

Let t_n be the length of the nth CPU burst, and let τ_{n+1} be our predicted value for the next CPU burst. Then, for α, $0 \leq \alpha \leq 1$, define

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\tau_n.$$

Topics Uncovered

- Scheduling Algorithms
  - First-Come, First-Served Scheduling
  - Shortest Job First / Next Non-Preemptive 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