OPERATING SYSTEMS

Process Management 3

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

- CPU Scheduling
  - Basic Concepts
    - CPU – I/O Burst Cycle
    - CPU Scheduler
    - Preemptive Scheduling
    - Dispatcher
  - Scheduling Criteria
- Typical Q and As

CPU Scheduling

- **CPU scheduling** is the basis of **multiprogrammed operating systems**.
- By **switching** the CPU among processes, the operating system can make the computer more productive.
- On operating systems that support **threads**, it is **kernel-level threads** not processes that are in fact being **scheduled** by the operating system.
- **Process scheduling** is used when discussing general scheduling concepts and **thread scheduling** to refer to thread-specific ideas

CPU Scheduling

- **CPU scheduling** is the basis of **multiprogrammed operating systems**.
- By **switching** the CPU among processes, the operating system can make the computer more productive.
- On operating systems that support **threads**, it is **kernel-level threads** not processes that are in fact being **scheduled** by the operating system.
- **Process scheduling** is used when discussing general scheduling concepts and **thread scheduling** to refer to thread-specific ideas

CPU Scheduling: Basic Concepts

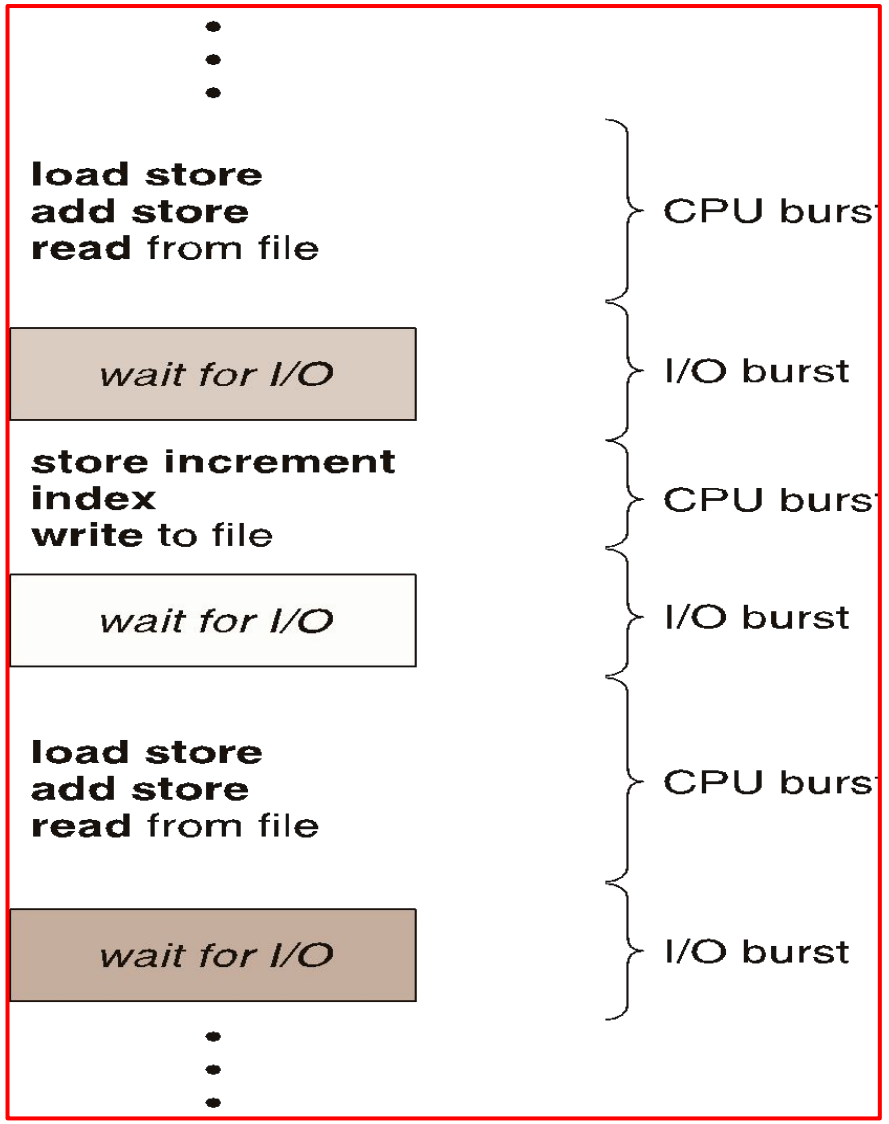
- In a system with a **single** CPU core, only **one** process can run at a time.
- **Other Processes** must wait until the CPU is **free** and can be **rescheduled**.
- The objective of multiprogramming is to have some process running at all times, to **maximize CPU utilization**.
- Several processes are kept in **memory** at one time.
- When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process.
- This pattern continues. Every time one process has to wait, another process can take over use of the CPU.
- On a multicore system, this concept of keeping the CPU busy is extended to all processing cores on the system.

CPU Scheduling: Basic Concepts

- The idea is relatively simple. A process is **executed** until it must **wait**, typically for the completion of some I/O request.
- In a **simple** computer system, the CPU then just sits **idle**. All this waiting time is wasted; **no useful** work is accomplished.
- With multiprogramming, we try to use this time productively.
- **Scheduling** of this kind is a **fundamental** operating-system **function**.
- Almost **all** computer resources are **scheduled** before **use**.
- The **CPU** is one of the **primary** computer resources.
- **CPU scheduling** is **Central** to operating-system **design**.

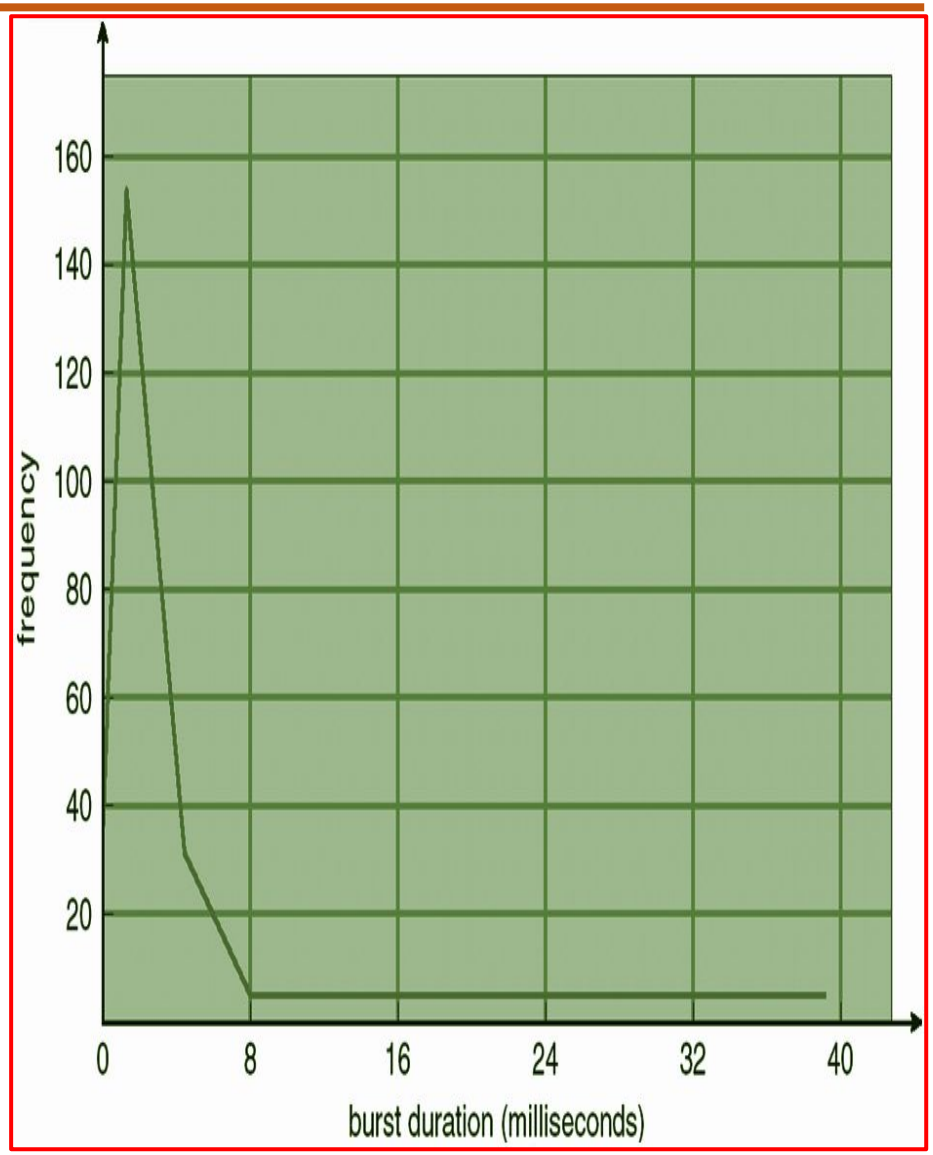
CPU Scheduling: Basic Concepts: CPU - I/O Burst Cycle

- 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 followed by I/O burst
- CPU **burst** distribution is of **main concern**



CPU Scheduling: Basic Concepts: Histogram of CPU Burst Cycle

- The **durations** of CPU bursts have been measured extensively.
- Although they vary greatly from process to process and from computer to computer, they **tend** to have a frequency **curve similar** to that shown in the Figure on the side
- An **I/O-bound program** typically has many **short CPU bursts**.
- A **CPU-bound program** might have a **few long CPU bursts**.
- This **distribution** can be **important** when **implementing** a CPU scheduling algorithm.



CPU Scheduling: CPU Scheduler

- **Short-term scheduler** selects from among the processes in **ready queue**, and allocates the CPU to one of them
- **Queue** may be **ordered** in **various** ways
- 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 **non-preemptive**
- All other scheduling is **Preemptive**
- **Preemptive Scheduling is mostly used in the following cases**
    - Consider **access** to **shared** data
    - Consider **preemption** while in **kernel mode**
    - Consider **interrupts** occurring during crucial **OS activities**

CPU Scheduling: Preemptive Versus Non-Preemptive

- 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.
- Virtually all modern operating systems including Windows, macOS, Linux, and UNIX use **preemptive** scheduling algorithms.
- **Flipside** of preemptive scheduling is that it can result in **race** conditions when data are shared among several processes.
    - For Example:
        - While one process is **updating** the **shared** data, it is **preempted** so that the second process can run.
        - The second process then tries to read the data, which are in an **inconsistent** state.
- A **pre-emptive kernel** requires mechanisms such as **mutex locks** to prevent race conditions when accessing shared kernel data structures.
- Most modern operating systems are now **fully preemptive** when running in **kernel mode**.

CPU Scheduling: Dispatcher

- **Dispatcher** module gives control of the CPU to the process selected by the short-term scheduler which involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program

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

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

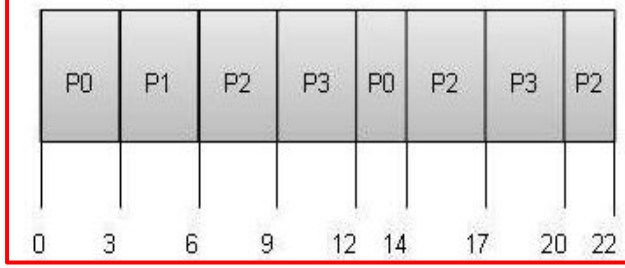
Sample Schemas and Tools for solving Scheduling Problems

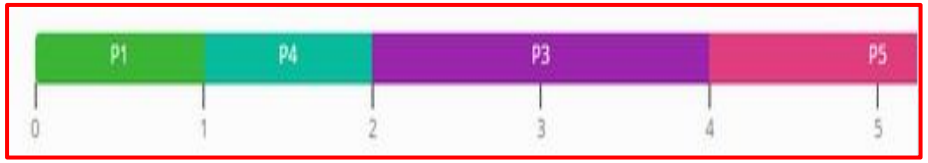
Process	Arrival Time	Execute Time	Service Time

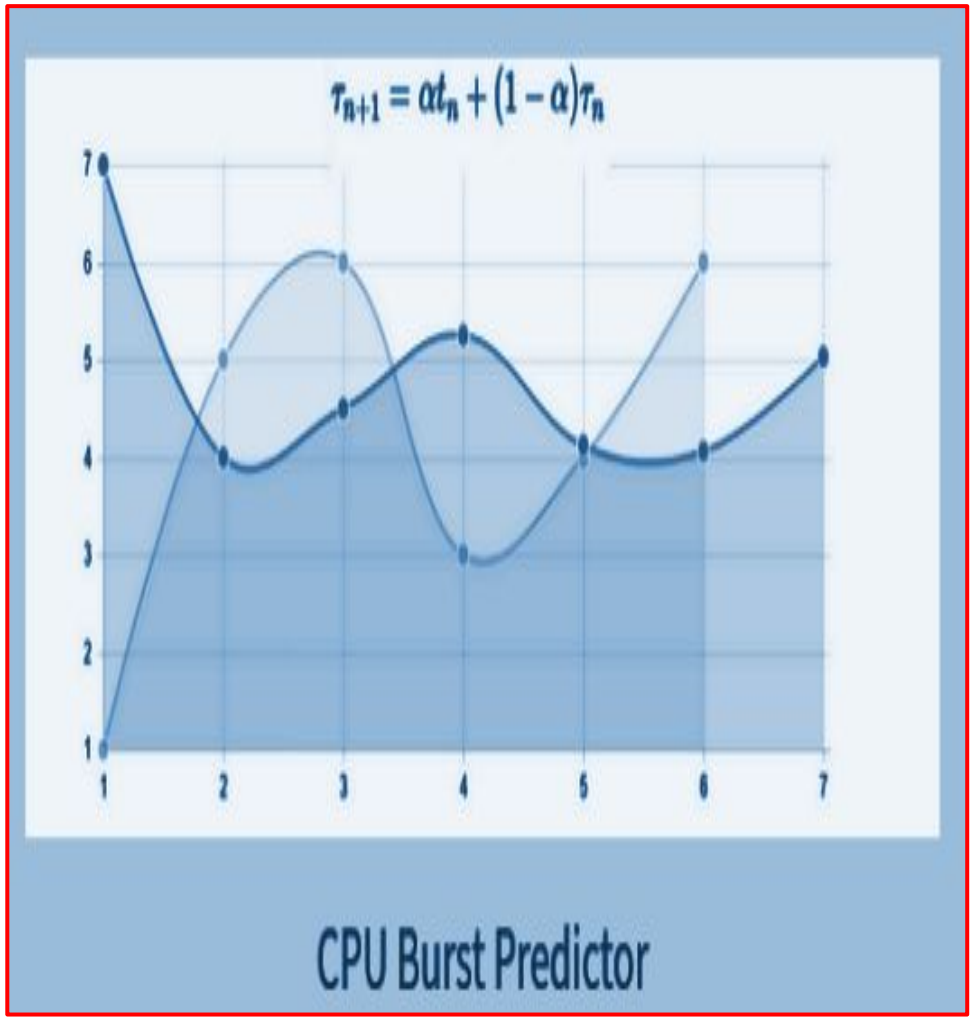
Process	Execution time	Arrival time

Process	Arrival Time	Execution Time	Priority	Service Time

GANTT Chart







Typical Q and As

short-term	_____ scheduler selects from among the processes in ready queue, 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
Running-Waiting & Terminates	non-preemptive
Running-Ready & Waiting-Ready	preemptive
CPU-I/O Burst Cycle	cycle of CPU execution and I/O wait
CPU Burst is followed by I/O Burst	...
CPU Burst distribution	main concern in CPU scheduling
dispatcher	gives control of the CPU to the process selected by the short-term scheduler

Typical Q and As

dispatcher underlying processes	1. switching context 2. switching to user mode 3. jumping to proper location in the user program
dispatch latency	time it takes for the dispatcher to stop one process and start running another one
Scheduling Criteria	1. CPU utilization 2. Throughput 3. Turnaround time 4. Waiting time 5. Response time
CPU utilization	keep CPU busy as possible

Typical Q and As

Throughput	number of processes that complete their execution
Turnaround time	time to execute a process
Waiting time	time taken by the process waiting in ready queue
Response time	time take from when a request was given until first response is produced
Scheduling Criteria Optimization	1. Max CPU utilization 2. Max throughput 3. Min. turnaround time 4. Min. waiting time 5. Min. response time

Topics Outline

- CPU Scheduling
  - Basic Concepts
    - CPU – I/O Burst Cycle
    - CPU Scheduler
    - Preemptive Scheduling
    - Dispatcher
  - Scheduling Criteria
- Typical Q and As

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