Serving Education for Last 40 Years

NEERAJ®

MCS-203

Operating Systems

Chapter Wise Reference Book
Including Many Solved Sample Papers

Based on

I.G.N.O.U.

& Various Central, State & Other Open Universities

By: *Anand Prakash Srivastava*

NEERAJ PUBLICATIONS
(Publishers of Educational Books)

Mob.: 8510009872, 8510009878 E-mail: info@neerajbooks.com
Website: www.neerajbooks.com

MRP ₹ 320/-

Content

OPERATING SYSTEMS

S.No.	*Chapterwise Reference Book*	*Page*

BLOCK-1: INTRODUCTION TO OPERATING SYSTEMS AND PROCESS MANAGEMENT

BLOCK-2: MEMORY MANAGEMENT, FILE MANAGEMENT AND SECURITY

S.No.	Chapterwise Reference Book	Page

Sample Preview of the Solved Sample Question Papers

Published by:

NEERAJ PUBLICATIONS
www.neerajbooks.com

QUESTION PAPER

June – 2024

(Solved)

OPERATING SYSTEMS

MCS-203

Time: 3 Hours] ***[Maximum Marks : 100***
Weightage : 70%

Note: *Question No. 1 is compulsory. Attempt any* ***three*** *questions from the rest.*

Q. 1. *(a)* Characterize a deadlock in a system. Using a resource allocation graph, illustrate a deadlock.

Ans. Ref.: See Chapter-4, Page No. 39, 'Characterisation of a Deadlocks' and 'Resource Allocation Graph'.

***(b)* Discuss the design goals and design issues in a distributed system.**

Ans. Ref.: See Chapter-10, Page No. 96, 'Design Goals of Distributed Systems' and Page No. 97, 'Design Issues involved in Distributed Systems'.

***(c)* With the help of a block diagram, explain the iOS-layered architecture with a focus on essential functions of all the layers.**

Ans. Ref.: See Chapter-15, Page No. 157, 'Architecture of iOS'.

***(d)* With reference to memory management in Windows-10 operating system, explain the following:**

***(i)* Virtual Memory Organization.**

Ans. Ref.: See Chapter-12, Page No. 121, 'Virtual Memory Organization'.

***(ii)* Demand Paging.**

Ans. Ref.: See Chapter-12, Page No. 121, 'Demand Paging'.

Q. 2. *(a)* Discuss the file management and security features in Android operating system.

Ans. Ref.: See Chapter-14, Page No. 150, 'File Management in Android and Security Features in Android'.

***(b)* Elucidate the general design issues for a mobile operating system in detail.**

Ans. Ref.: See Chapter-11, Page No. 109, 'Design Issues in Mobile OS'.

Q. 3. *(a)* Write and explain Bakery's algorithm that handles critical section problem for '*n*' processes.

Ans. Ref.: See Chapter-3, Page No. 28, 'Bakery's algorithm'.

***(b)* Write and explain briefly, all the essential functions of an operating system.**

Ans. Ref.: See Chapter-1, Page No. 4, 'Functions of OS'.

Q. 4. *(a)* Explain Virtual Memory and its principle of operation, elaborating the virtual to physical address mapping.

Ans. Ref.: See Chapter-6, Page No. 60, 'Virtual Memory'.

***(b)* With the help of a diagram, explain LINUX architecture with a special focus on 'Kernel' and its components.**

Ans. Ref.: See Chapter-13, Page No. 132, 'Linux Kernel' and Page No. 133, 'Fundamental Architecture of Linux'.

Q. 5. Write short notes on the following:

***(i)* Types of Schedulers (Short-term, Long- term and Medium term).**

Ans. Ref.: See Chapter-2, Page No. 14, 'Types of Schedulers'.

***(ii)* Demand Paging.**

Ans. Ref.: See Chapter-6, Page No. 61, 'Demand Paging'.

***(iii)* Mutual Exclusion in Distributed Systems.**

Ans. Ref.: See Chapter-10, Page No. 99, 'Mutual Exclusion in Distributed Systems'.

***(iv)* Libraries and Application Framework Layers of Android Architecture.**

Ans. Ref.: See Chapter-14, Page No. 147, 'Layered Architecture of Android OS', and Page No. 148, 'Libraries', 'Application Framework'.

■■

QUESTION PAPER

December – 2023

(Solved)

OPERATING SYSTEMS

MCS-203

Time: 3 Hours] ***[Maximum Marks : 100***
Weightage : 70%

Note: *Question No. 1 is compulsory. Attempt any* ***three*** *questions from the rest.*

Q. 1. ***(a)*** **Consider the following set of processes (P_1, P_2, P_3, P_4, P_5) with the length of the CPU burst time given in milliseconds along with the arrival time:**

Process	Arrival Time	CPU Burst Time
P_1	1	6
P_2	2	4
P_3	4	1
P_4	3	2
P_5	5	3

Draw Gantt charts illustrating the execution of the processes using FCFS, SJF and RR (with quantum = 2) scheduling algorithms. Also find the average turn around time, average waiting time, processor utilization and throughput for all the algorithms.

Ans. Step 1: Understanding the Given Processes:
We are given the following processes:

Process	Arrival Time	Burst Time
P_1	1	6
P_2	2	4
P_3	4	1
P_4	3	2
P_5	5	3

Arrival Time: When the process arrives in the ready queue.

Burst Time: The amount of time the process needs on the CPU.

Waiting Time (WT): Time spent waiting in the ready queue = (Turnaround Time - Burst Time).

Turnaround Time (TAT): Total time from process arrival to completion = (Completion Time - Arrival Time).

Step 2: FCFS (First Come First Serve) Scheduling:

Algorithm:
1. Sort the processes by arrival time.
2. Execute each process in order of arrival.

Execution:

P_1 arrives at time 1 → Starts execution immediately.

P_2 arrives at time 2 but has to wait until P_1 finishes at time 7.

P_4 arrives at time 3, but P_2 is next in line.

P_3 arrives at time 4 but has to wait.

P_5 arrives at time 5 but has to wait.

Process	Arrival	Burst	Start Time	Completion Time	Turnaround Time (TAT)	Waiting Time (WT)
P_1	1	6	1	7	6 (7-1)	0 (1-1)
P_2	2	4	7	11	9 (11-2)	5 (7-2)
P_3	3	2	11	13	10 (13-3)	8 (11-3)
P_4	4	1	13	14	10 (14-4)	9 (13-4)
P_5	5	3	14	17	12 (17-5)	9 (14-5)

Gantt Chart for FCFS:

		P_1	P_1	P_1	P_1	P_1	P_1	P_2	P_2	P_2	P_2	P_4	P_4	P_3	P_5	P_5	P_5
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

FCFS Average Times:

Average Turnaround Time = (6 + 9 + 10 + 10 + 12)/5 = 9.4

Average Waiting Time = (0 + 5 + 8 + 9 + 9)/5 = 6.2

Step 3: SJF (Shortest Job First) Scheduling:

Algorithm:

1. Select the process with the shortest burst time among the available ones.

Sample Preview
of
The Chapter

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

OPERATING SYSTEMS

Operating System: An Overview

1

INTRODUCTION

Working with Computers, an Operating System (OS) would be an unpleasant complication of details. Carrying out a seemingly simple procedure, like loading an application program from disk to primary memory, would require hours of work. Application programs themselves would be longer, more complicated and more expensive than they are now. Special versions of software would have to be written for virtually every model of machines. Luckily the availability of standard Operating System has prevented all of this and more.

CHAPTER AT A GLANCE

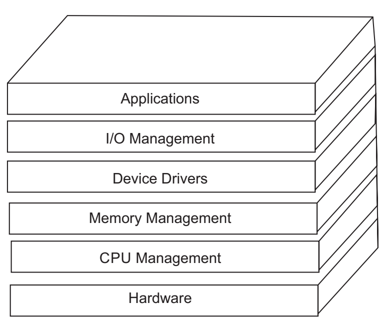
WHAT IS AN OPERATING SYSTEM?

An Operating System run on computer hardware and serves as a platform for other software to run on the computer system.

An Operating System is a program (system software) which acts as an interface between user and computer hardware.

A Computer System can be logically divided into four components: Computer Hardware, Operating System, Application Programs and Users.

An Operating System controls and coordinates the hardware components (CPU, Memory, I/O devices) among various application programs for different users.



The Operating System control every task your computer carries out and manages system resources.

Operating System has two main objectives:

Convenience: An Operating System makes a computer more convenient to us.

Efficiency: An Operating System allows the computer resources to be used in an efficient manner.

Operating System is also called a Resource Manager. Operating System manages all computer resources and allocates them to a specific program and uses as it require completing its tasks.

The Operating System hides all the innermost details of the working of computer hardware components and provides a simple, usable and effective model of a computer. This type of image of a computer is called virtual machine.

The common operating systems are the windows family of operating systems (*viz*. Windows 95, 98, 2000.NT, VISTA), the UNIX family of operating systems (which includes UNIX, LINUX and many other derivatives) and the Macintosh Operating System.

GOALS OF AN OPERATING SYSTEM

Any Operating System should meet the following major goals:

(a) Optimize the use of computer resource so as to maximize its throughput.

(b) Create a user-friendly computing environment for accessing the computer resources.

(c) To hide details of hardware by creating abstraction.

(d) To allocate resources to processes (Manage resources).

(e) Provide a pleasant and effective user interface.

GENERATIONS OF OPERATING SYSTEMS

The history of computer development is often referred to in reference to the different generations of computing devices. Each generation of computer is characterized by a major technological development that fundamentally changed the way computers operate, resulting in increasingly smaller, cheaper, and more powerful and more efficient and reliable devices.

Read about each generation and the developments that led to the current devices that we use today.

0th Generation: (1642-1940s)

The term "0th generation" refers to the early development of computing before the commercial production of computer equipment. This period dates back to Charles Babbage's invention of the Analytical Engine. Key developments include John Atanasoff's computer in 1940, the Mark I by Howard Aiken and IBM engineers in 1944, ENIAC by Wallace Eckert and John Mauchly, and EDVAC (1944-46) by John Von Neumann, Arthur Burks, and Herman Goldstine, which introduced stored programs and serial instruction execution. EDVAC's innovations laid the foundation for commercial computing and operating systems. During this era, hardware relied on vacuum tubes, and programs were written in machine language, with no operating systems to manage operations.

First Generation (1951-1956)

The first generation of commercial computing began with Eckert and Mauchly's UNIVAC I in 1951 and the IBM 701 shortly after. This era featured vacuum tubes and a "closed shop" operation mode, where hired operators managed tasks manually. Programs were written in higher-level languages, requiring operators to compile, link, and run them, gradually expanding their responsibilities.

Second Generation (1956-1964)

The second generation of computers saw transistors replace vacuum tubes, marking a key hardware advancement. Systems were largely card and tape-based, with disk storage emerging later. Centralized computers operated under mono-programmed batch processing systems dominated this era.

Third Generation (1964-1979)

The third generation began in April 1964 with IBM's System/360, introducing integrated circuits (ICs) for improved speed and cost efficiency. Operating systems advanced with multiprogramming and spooling, exemplified by IBM's HASP system. Spooling used system readers to move input from cards to disk and system writers to direct output to printers or other media, all transparent to users.

Fourth Generation (1979-Present)

The fourth generation introduced personal computers and workstations, driven by very large-scale integration (VLSI), enabling thousands of transistors on a single chip. This advancement allowed desktop computers to surpass the power of earlier room-sized machines. Operating systems returned to an open-shop model, with each user having full control of a computer during job execution, now enhanced by user-friendly interfaces and significant technological progress.

TYPES OF OPERATING SYSTEMS

Modern operating systems are categorized into three types based on user-program interaction during processing: batch, time-sharing, and real-time systems.

Batch Processing Operating System

In a batch processing operating system, users submit jobs to a central location where they are grouped into a batch and placed in an input queue for processing. Users do not interact with the jobs during execution. The computer's response time, known as turnaround time, is the duration from job submission to the completion of execution and result delivery.

Time Sharing

Time-sharing operating systems enable multiple users to access computing services concurrently online. Users share the central processor, memory, and other resources, managed by the operating system. This setup allows near-complete interaction with programs during execution, with response times typically within a few seconds.

Real Time Operating System (RTOS)

Real-time operating systems (RTOS) are designed for applications where immediate response is critical to avoid errors, misrepresentation, or disaster. Examples include airline reservations, machine tool control, and nuclear power station monitoring. These systems are built to handle external signals requiring instant attention. RTOS is commonly used to control machinery, scientific instruments, and industrial systems, with minimal user interface and no end-user utilities.

Multiprogramming Operating System

A multiprogramming operating system allows multiple active user programs (or parts of them) to reside in main memory simultaneously. While time-sharing systems are a type of multiprogramming system, not all multiprogramming systems are time-sharing systems. For instance, batch and real-time operating systems often handle multiple active programs in main storage. A related term is "multiprocessing", which refers to systems with multiple processors working together.

Multiprocessing System

A multiprocessing system is a hardware setup with multiple independent processing units. This configuration is typically associated with large computer systems used in advanced scientific or commercial applications.

Networking Operating System

A networked computing system consists of interconnected physical computers. Each computer's operating system must support not only its standalone functionality, but also communication and data transfer between connected systems. While network operating systems require additional components like a network interface controller, low-level drivers, and software for remote login and file access, these additions do not fundamentally alter the core structure of the operating system.

Distributed Operating System

A distributed computing system consists of multiple interconnected computers that collaboratively share or distribute processing loads among themselves, either evenly or based on specific configurations. Its operating system must not only perform standard standalone tasks, but also coordinate operations and manage information flow across all connected systems. Distributed and networked computing environments require more advanced functionality. Unlike a distributed system, in a networked operating system, users are aware of multiple computers, can log in to remote machines, and transfer files between them. Each computer in a network runs its own local operating system and serves its own users.

Operating Systems For Embedded Devices

As embedded systems (such as PDAs, cellphones, point-of-sale devices, VCRs, industrial robots, and even toasters) evolve with increasingly complex hardware and additional features, their applications often rely on operating system code to streamline development. Some popular embedded operating systems include:

- **Nexus's Conix:** An embedded OS designed for ARM processors.
- **Sun's Java OS:** A standalone virtual machine not dependent on another OS, primarily aimed at embedded systems.
- **Palm Computing's Palm OS:** A leading OS for PDAs, widely supported with numerous applications and partnerships.

DESIRABLE QUALITIES OF OS

The desirable qualities of an operating system include Usability, Facilities, Cost, and Adaptability:

Usability:

- **Robustness:** Handles all valid inputs effectively.
- **Consistency** and proportionality.
- **Convenience:** Offers powerful, high-level features.

Facilities: Sufficient, complete, and appropriate for intended use.

Cost: Low cost and efficient services with good algorithms. Optimized space/time trade-offs and minimal idle overhead. Low maintenance requirements.

Adaptability: Tailored to the environment, supporting essential activities without unnecessary restrictions. Flexible to adapt to changing needs and resources, such as new devices or users. Extensible to accommodate new features seamlessly.

OPERATING SYSTEMS: SOME EXAMPLES

In the previous section, we explored the types of operating systems. Here, we will examine some of the most popular operating systems.

DOS

DOS (Disk Operating System) was the first widely-used operating system for personal computers. It serves as a master control program that runs automatically when the PC starts, enabling program execution and file management. A single-user OS developed by Microsoft, DOS was the foundation for Windows 3.1, 95, 98, and ME. Later versions like Windows NT, 2000, and XP included DOS emulation to support existing DOS applications.

UNIX

UNIX is widely used in workstation products from companies like Sun Microsystems, Silicon Graphics, and IBM. It played a key role in the development of the Internet and the shift from individual computing to network-based computing. Linux, a UNIX derivative available as both free and commercial software, is growing in popularity as an alternative to proprietary operating systems.

Developed by AT&T and written in C, UNIX and C were freely distributed to government and academic institutions, leading to widespread portability across various machine families. This made UNIX synonymous with "open systems".

WINDOWS

Windows, developed by Microsoft, is a widely-used personal computer operating system that, along with applications like Microsoft Word and Excel, has become a standard for individual users in corporations and homes. It includes built-in networking capabilities, enabling file and application sharing on connected PCs.

In large enterprises, Windows clients often connect to UNIX or NetWare servers, while server versions like Windows NT and 2000 are increasingly popular for providing a Windows-only client-server solution. Supported by Microsoft and a vast network of developers, Windows owes much of its success to its networking features. However, versions like Windows 95, 98, ME, NT, 2000, and XP are known for their complexity.

MACINTOSH

The Macintosh (Mac), introduced by Apple in 1984, was the first widely-sold personal computer with a graphical user interface (GUI). It aimed to provide a user-friendly experience with features like the mouse, icons to represent actions, point-and-click, click-and-drag actions, and intuitive window operations. Microsoft later adopted many of these interface concepts in its Windows operating system.

While the Mac has fewer applications compared to Windows, it offers all essential software, making it suitable for most users. Data compatibility between Windows and Mac, though sometimes a concern, is often exaggerated and easily resolved.

FUNCTIONS OF OS

The main functions of an operating system are as follows:

Process Management

The CPU executes numerous programs, including user programs and system activities, collectively called processes. A process is a program in execution, which can include batch jobs, time-shared programs, or system tasks like spooling. While often seen as jobs or time-shared programs, the concept of a process is broader.

The operating system manages processes by handling:

- Creation and deletion of user and system processes.
- Suspension and resumption of processes.
- Mechanisms for process synchronization.
- Mechanisms for deadlock handling.

Memory Management

Memory, a costly component of a computer system, consists of a large array of addresses for storing and retrieving data. The CPU interacts with memory by reading from and writing to specific addresses.

The operating system handles memory management by:

- Tracking memory usage and identifying users.
- Determining which processes to load into memory when space becomes available.
- Allocating and deallocating memory as needed.

Secondary Storage Management

The primary purpose of a computer system is to execute programs, which, along with their data, must reside in main memory during execution. However, since main memory is limited, secondary storage, typically disks, is used to store programs and data. Modern systems rely on disks as the primary online storage for both. Programs like compilers, assemblers, and editors are stored on disks until loaded into memory and often use the disk as a source and destination during processing. Therefore, efficient disk storage management is critical to a computer system.

I/O Management

An operating system aims to abstract the complexities of hardware devices from users. For instance, in UNIX, the I/O system conceals device-specific details even from much of the operating system itself. The OS manages I/O through:

- A buffer caching system.
- Activating general device driver code.
- Running specific hardware driver software when needed.

File Management

File management is a key service provided by an operating system. Computers store data in various physical forms, such as magnetic tape, disk, and drum, each with unique characteristics and organization.

To simplify usage, the operating system offers a uniform logical view of storage, abstracting the physical properties of storage devices into a logical unit called a file. The operating system maps these logical files onto physical storage devices.

Protection

Processes in an operating system must be protected from interfering with each other. To ensure this, the operating system implements mechanisms that allow access to files, memory, CPU, and other resources only to authorized processes.

Examples include:

- **Memory addressing hardware:** Restricts processes to their own address space.
- **Timers:** Ensure no process monopolizes the CPU.
- **I/O restrictions:** Prevent processes from directly managing peripheral devices, maintaining their integrity.

Protection involves controlling and enforcing access to resources, ensuring system security and stability.

Networking

A distributed system consists of multiple processors, each with its own local memory and no shared memory or clock. Communication between processors occurs through various channels, such as high-speed buses or telephone lines. Distributed systems can range in size and function, involving components like microprocessors, workstations, minicomputers, and large general-purpose computers.

Command Interpretation

The command interpreter is a key component of an operating system, serving as the primary interface between the user and the system.

In batch systems or when a user logs into a time-shared system, a program that reads and interprets control statements is automatically executed. Known as the control card interpreter, command line interpreter, or shell (in Unix), its primary function is straightforward: retrieve the next command and execute it.