Page No.
Date

Operating System –
It is system software that acts as an interface between the user and the hardware.

Main Functions :
1. Process Management
creating, scheduling, and terminating processes
2. Memory Management
allocating and deallocating memory
3. File System Management
handling files and directories
4. Device Management
managing I/O devices with drivers
5. Security & Protection
ensuring safe execution of processes and data protection
6. Networking & Communication
allowing processes to communicate (IPC).

It hides hardware complexity and provides a convenient, efficient environment for users and applications.

Program : A static set of instructions stored on disk

Process : A program in execution, has its own memory space (stack, heap, code, data).

Page No.
Date

Thread : The smallest unit of execution within a process ; shares process memory but has its own execution stack.

Ex: A browser is a process → each tab is a thread

Types of Operating Systems –

- Batch OS
  executes job in batches without user interaction
- Multiprogramming OS
  multiple programs loaded in memory, CPU switches between them
- Multitasking OS
  allows multiple tasks to run seemingly simultaneously for users
- Real time OS
  guarantees response within deadlines, used in embedded systems.
- Distributed OS
  manages group of computers to work as one
- Network OS
  provides networking features across systems.

Batch OS → efficient use of CPU and peripheral devices but no direct interaction, debugging is hard.

Page No.
Date

Multiprogramming OS → focuses on efficiency., improves CPU utilization

Multitasking OS → focuses on user convenience.

- User Mode : Application programs run with limited access (no direct hardware access)
- Kernel Mode : OS runs with full access to hardware and system resources
  Provides protection → prevents user programs from crashing or corrupting the system

Process : A program in execution
States –
1) New → process created
2) Ready → waiting for CPU
3) Running → currently executing
4) Waiting / Blocked → waiting for I/O or event
5) Terminated → finished execution
OS scheduling relies on these states to manage CPU utilization.

Page No.
Date

Normal Function Call –
Runs in user mode, directly invokes another function within the program
Ex: add(x, y)

System Call –
Request made by a process to the OS for a service (eg. file handling I/O), switches from user mode to kernel mode
Ex: read(), write(), fork()

System calls provide controlled access to hardware, function calls just reuse code.

CPU
Process Scheduling Algorithms
Scheduling decides which process gets CPU when multiple are in ready queue.
- Preemptive Scheduling
  CPU can be taken away from a process if a higher priority job arrives.
  Ex: Round Robin, SRTF, Priority (Preemptive)
- Non-Preemptive Scheduling
  Once a process gets CPU, it runs till completion or waiting state.
  Ex: FCFS, SJF (non-preemptive)

Balances CPU utilization, throughput, waiting time, and fairness.

Page No.
Date

Scheduling Algorithms –

- FCFS (First come, First served)
  Processes served in arrival order. Simple, but poor for short jobs after long jobs.
- SJF (Shortest Job First)
  Executes shortest burst time job first. Minimizes average waiting time. Non-preemptive.
- SRTF (Shortest Remaining Time First)
  Preemptive version of SJF. Best for minimizing turnaround, but high context switching.
- RR (Round Robin)
  Each process gets fixed time slice in cyclic order. Ensures fairness. Used in time sharing systems.
- PBS (Priority Based Scheduling)
  CPU given to highest priority process. Can be preemptive / non preemptive.
  Problem → starvation

Starvation : Process waits indefinitely because high priority processes keep preempting. Common in priority scheduling.

Aging : Technique to avoid starvation by gradually increasing the priority of a waiting process.

Page No.
Date

Producer - Consumer Problem —

Problem : Two processes share a buffer →
Producer adds items, Consumer removes items
Need synchronization to avoid :
- Producer adding to full buffer
- Consumer removing from empty buffer

Solution : Use semaphores (empty, full, mutex) to ensure mutual exclusion and synchronization

Critical Section Problem —

Critical Section : code segment where shared resources are accessed

Problem : Multiple processes entering critical section simultaneously → data inconsistency.

Solution :

1) Mutual Exclusion - only one process inside
2) Progress - if no process is in critical section, one must be allowed to enter
3) Bounded Waiting - limit waiting time

Tools : Locks, Semaphores, Monitors

→ Synchronization : Ensures correct execution order when processes share resources

Tools

1. Semaphores - integer variable to manage resource access
2. Mutex - binary lock for exclusive access

3. Monitors – high level abstraction with condition variables

4. Spinlocks – busy wait locks used in kernel

→ Semaphore : Synchronization tool, integer variable, modified by wait (P) and signal (V) operations.

Types

1) Binary Semaphore (Mutex) – Only 0 or 1 → ensures mutual exclusion

2) Counting Semaphore – non negative integer → controls multiple resource instances.

Deadlock –

Set of processes waiting indefinitely for resources held by each other.

Necessary Conditions (Coffman):

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular Wait

Prevention Methods :

- Eliminate one condition (eg. avoid Hold & Wait)
- Force ordering of resources to break Circular wait.
- Use timeouts

Page No.
Date

Banker's Algorithm —
Deadlock avoidance algorithm based on resource allocation state.
Working :
- Each process declares max resources it may need.
- OS allocates resources only if it leaves system in a safe state
- Safe state = there exists at least one sequence of processes that can complete.

Example : used in banking systems analogy (ensures resources are never over allocated).

Memory Management —
OS function that allocates and deallocates memory to processes efficiently.
Placement Strategies :
- First Fit
  Allocate the first hole large enough → Fast, may cause external fragmentation
- Best Fit
  Allocate the smallest hole sufficient → Less wastage, but slower search
- Worst Fit
  Allocate the largest hole → leaves larger free blocks, but can waste space.

Page No.
Date / /

Virtual Memory –
Technique that gives illusion of a very large memory by combining physical memory + disk storage.
Uses :
- Allows processes larger than RAM
- Enables multiprogramming

Implementation : Paging + Demand Paging

Contiguous Allocation : Each process occupies one continuous memory block. Simple but causes fragmentation.

Paging : Divides memory into fixed size blocks (frames / pages). Eliminates external fragmentation.

Segmentation : Divides memory based on logical units (code, stack, data). Provides logical views but suffers external fragmentation.

Dynamic Binding –
Binding of a method call to method implementation at runtime.
Binding addresses at execution instead of compile/load time. (used in virtual memory, linking libraries).

Page No.
Date

Page Fault –

Occurs when a program accesses a page not in physical memory.

Handling Steps :

1) OS traps to kernel
2) Finds page in disk (swap space)
3) Brings it to RAM (may replace an existing page)
4) Updates page table & resumes process.

Page Replacement Algorithms

- FIFO (First In First Out)
  Replace oldest page. Simple but poor performance.
- LRU (Least Recently Used)
  Replace page not used for longest time. Good approximation of optimal
- Optimal
  Replace page that won't be used for longest future time. Theoretical best, but impractical (needs future knowledge)

Belady's Anomaly

In FIFO, increasing number of page frames may increase page faults (counterintuitive).

Example : Seen in reference strings like

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Page No.
Date

Thrashing –
Condition when CPU spends most time swapping pages in and out instead of executing instructions.
Cause : High degree of multiprogramming with insufficient frames
Solution : Reduce degree of multiprogramming, use working set model.

Disk Scheduling –
OS decides order of servicing I/O requests from disk queue. Goal → minimize seek time and latency.

Disk Scheduling Algorithms :

- FCFS
  Serve requests in arrival order. Fair but high average seek time
- SSTF (Shortest Seek Time First)
  Serve nearest request. Faster, but causes starvation.
- SCAN (Elevator Algorithm)
  Head moves back and forth like elevator, servicing requests.
- C-SCAN (Circular SCAN)
  Head moves in one direction, jumps back. Provides uniform wait time.
- LOOK & C-LOOK
  Like SCAN and C-SCAN, but head stops at last request instead of disk end.

Page No.
Date / /

Cache –
Small, high speed memory between CPU and main memory.
Stores frequently accessed data → reduces average memory access time.

Cache Mapping Techniques –
- Direct Mapping
Each block of memory maps to only one cache line.
Simple but can cause conflicts.
- Associative Mapping
Any block can go to any cache line. Flexible but costly.
- Set Associative Mapping
Hybrid, memory block maps to set, then placed in any line within set. Balance of cost and efficiency.

File System –
It is a method used by an OS to store, organize, and manage data on storage devices (disk / SSD).
Main components :
1) Files : Collection of related data identified by a name.
2) Directories : Organize files hierarchically
3) Metadata : Info about files (size, permissions, timestamps, owner)
4) File Control Block (FCB) / Inodes : Data structures

containing file details and disk block location

5) Free Space Manager : Tracks unallocated disk blocks.

6) Access Methods : How files are read / written (sequential, direct, indexed).

Types of File System

- FAT (File Allocation Table)
  Used in older windows, simple, but not efficient for large disks.
- NTFS (New Technology File System)
  Used in modern windows, supports security, journaling, large files
- EXT (Extended File system family - EXT2, EXT3, EXT4)
  Common in Linux, EXT4 supports journaling, large volumes
- HFS + / APFS
  Apple's file systems, APFS optimized for SSDs.
- exFAT
  Lightweight FS for USB / SD cards
- ISO 9660 / UDF
  Used for optical disks (CD / DVD)

Page No.
Date

File Allocation and Deallocation

Goal : Efficiently allocate disk blocks to files and free them when deleted

Techniques :

- Contiguous Allocation
  File stored in continuous blocks → Fast access, but causes external fragmentation
- Linked Allocation
  Each file block points to the next → Solves fragmentation, but random access is slow
- Indexed Allocation
  Uses an index block that stores addresses of all file blocks → Supports direct access, but index block can be large

Deallocation : When a file is deleted, blocks are marked as free in free space manager (bitmap or linked list)

Contiguous – Best for large sequential files
Linked – Good for sequential, poor for random
Indexed – Best for random access.

Page No.
Date

Fragmentation

Occurs when free memory / disk space is broken into small pieces, making allocation inefficient.

Types :

- Internal Fragmentation
  Wasted space inside allocated block
- External Fragmentation
  Enough total free space exists, but scattered in small non-contiguous blocks.

Solutions :

- Compaction – rearranging memory to combine free blocks
- Paging / Segmentation – avoids external fragmentation
- Dynamic storage allocation techniques (First Fit, Best Fit, Worst Fit)

Blocking

- Process waits (is blocked) until the operation completes
- Ex: Reading a file, the process stops until data is read
- Advantage : Simple, predictable
- Disadvantage : CPU idle while waiting

Non Blocking

- Process initiates operation and continues executing without waiting
- Ex: read() call in non blocking mode, process checks later if data is ready.

- Advantage : Better CPU utilization
- Disadvantage : More complex to manage

Synchronous
- Operation completes before the next one starts. Time aligned execution
- Often blocking, but can be non blocking in some cases
- Ex: Function call that returns only after execution

Asynchronous
- Operation starts, control returns immediately, completion is notified via callback, signal or interrupt
- Often used in non blocking I/O.
- Ex: AJAX call, asynchronous disk I/O.

Spooling –
Simultaneous Peripheral Operations On-Line
technique where data is temporarily stored on a buffer (usually disk) before sending to a slower device (like printer).
Use Case : Print jobs are spooled → CPU doesn't wait for printer → sends next job to print queue
Why Improvement : Improves CPU utilization and allows multiple processes to share slow devices efficiently.

Page No.
Date

RAID –

Redundant Array of Independent / Inexpensive Disks combines multiple physical disks to improve performance, reliability, or both.

Types :

1) RAID 0 - Striping : Data split across disks → High performance, no redundancy

2) RAID 1 - Mirroring : Exact copy on multiple disks → High reliability, no performance gain

3) RAID 5 - Striping with Parity : Data + parity distributed → Fault tolerant, good performance, minimum 3 disks

4) RAID 6 - Striping with Double Parity : Can tolerate 2 disk failures

5) RAID 10 (1+0) : Mirroring + Striping → High reliability + performance, costly

RAID improves data availability and system throughput, critical for servers and enterprise storage.

Security Threats and Vulnerabilities –

Threats exploit vulnerabilities to compromise confidentiality, integrity, or availability of systems.

Common Security Threats :

- Malware – Viruses, worms, trojans
- Phishing & Social Engineering – trick users to disclose info
- Denial of Service (DOS / DDOS) – overload system to make it unavailable.

Page No.
Date

- Man in the Middle Attack – intercept and alter communications
- SQL Injection / XSS – exploit web app vulnerabilities

Vulnerabilities : Weaknesses that can be exploited
Ex:- weak passwords, unpatched software, misconfigured servers, open ports.

Threat → Vulnerability → Impact → Prevention

Threat : Potential attack (virus, phishing)

OS provide security through authentication, access control, isolation of processes, and encryption for files / data.

RAID → improves performance and fault tolerance
Backup → ensures recovery after accidental deletion or corruption.

API (Application Programming Interface) –
A set of rules that allow two software applications to communicate.
Acts like a bridge between different software components.