Operating Systems

Adesh Lokhande

1. Evolution of Operating Systems

Operating System ka evolution time ke sath bahut badla hai. Pehle computers bina OS ke hote the jaha user ko manually instructions deni padti thi. Fir **Batch Processing Systems** aaye jaha jobs ek ke baad ek run hoti thi. Uske baad **Multiprogramming** aur **Time Sharing Systems** aaye jisse multiple users ek hi computer use kar sakte the. Fir **Personal Computers (PCs)** ke liye **Single-user OS** like MS-DOS aur Windows aaye. Aaj ke time me hum **Modern Operating Systems** use karte hai jaise Windows, Linux, macOS, Android, jisme multitasking, networking, aur security features included hote hai.

2. Types of Operating Systems

Operating Systems ke alag-alag types hote hain jo system ke use par depend karte hain:

1. **Batch Operating System** – Ye system similar tasks ko ek saath batch me execute karta hai, human interaction kam hoti hai.
2. **Time-Sharing Operating System** – Multiple users ek hi system ko share karte hain, har user ko thoda-thoda CPU time milta hai.
3. **Distributed Operating System** – Ye multiple computers ko connect karke ek single system jaisa kaam karata hai.
4. **Real-Time Operating System (RTOS)** – Ye system real-time tasks ko handle karta hai, jaise industrial control ya robotics me use hota hai.
5. **Embedded Operating System** – Ye specific devices ke liye design kiya jata hai, jaise washing machine, mobile phones, routers, etc.

3. The Process Concept

Process ek running program hota hai. Jab hum koi program execute karte hain, to OS uske liye ek process create karta hai. Har process ka apna **Program Counter**, **Stack**, **Data Section**, aur **Heap** hota hai. Process ko OS manage karta hai jisme creation, scheduling, execution aur termination include hote hain. Simple words me, program ek static entity hai aur process uska dynamic version hota hai jo actually run ho raha hota hai.

4. System Programmer’s View of Processes

System programmer ke nazariye se, process ek **active entity** hai jisme program code ke saath execution context bhi include hota hai — jaise CPU registers, memory, file handles, aur process state. Programmer ke liye process ek **resource container** hota hai jise OS manage karta hai. OS har process ko alag address space deta hai taaki ek process dusre ke data me interfere na kare. Is view se process creation, scheduling, communication (IPC), aur synchronization important hote hain.

5. Operating System’s Views of Processes

Operating System ke point of view se, process ek **unit of work** hai jise CPU execute karta hai. OS har process ko manage karta hai through its **Process Control Block (PCB)** jisme process ID, state, registers,

aur memory info hoti hai. OS process ko create, schedule, aur terminate karta hai, aur ensure karta hai ki sab processes efficiently CPU share karein. Simple shabdon me, OS process ko ek task ke form me dekhta hai jise manage aur coordinate karna uski responsibility hoti hai.

6. Operating System Services for Process Management

Operating System process management ke liye kuch important services provide karta hai jaise:

1. **Process Creation and Termination** – Naye processes banana aur complete hone par unhe end karna.
2. **Process Scheduling** – Decide karna ki kaunsa process CPU par chalega aur kis order me.
3. **Process Synchronization** – Multiple processes ke beech coordination maintain karna taaki data conflict na ho.
4. **Inter-Process Communication (IPC)** – Processes ke beech data exchange karne ke liye mechanism dena.
5. **Deadlock Handling** – Jab processes ek dusre ke resources ka wait karte rahe to us situation ko handle karna.

In services ke through OS system ko efficient aur stable banata hai.

7. Process Concept

Process ek **running program** hota hai jo execution ke dauran system resources jaise CPU time, memory, aur files use karta hai. Jab koi program run hota hai, to OS uske liye ek process banata hai jisme **Program Counter**, **Stack**, **Heap**, aur **Data Section** hota hai. Har process ka ek unique Process ID (PID) hota hai. Simple shabdon me, program static hota hai aur process uska dynamic form hota hai jo actually execute ho raha hota hai.

8. Process Scheduling

Process Scheduling ka kaam hota hai decide karna ki kaunsa process CPU par next chalega. OS ek scheduler use karta hai jo ready queue me se processes ko select karta hai. Iske main types hain — **Long-term**, **Short-term**, aur **Medium-term scheduler**. Scheduling algorithms jaise **FCFS**, **SJF**, **Priority**, aur **Round Robin** help karte hain CPU ko efficiently use karne me. Simple shabdon me, process scheduling system performance aur responsiveness ko improve karta hai.

9. Operations on Processes

Processes par kuch basic operations perform kiye jaate hain:

1. **Creation** – Naye process ko create karna using system calls.
2. **Termination** – Process complete hone par ya error ke wajah se end karna.
3. **Suspension/Resumption** – Process ko temporarily stop aur fir resume karna.
4. **Waiting/Signaling** – Process ko resources ke liye wait karwana aur signal dena jab resource available ho.
5. **Context Switching** – CPU ko ek process se dusre process me switch karna while saving state.

Ye operations OS ko processes ko efficiently manage karne me help karte hain.

10. Cooperating Processes

Cooperating Processes wo processes hote hain jo ek dusre ke saath **data aur resources share** karte hain. Ye processes ek dusre ke results ko affect kar sakte hain aur **Inter-Process Communication (IPC)** ke through coordinate karte hain. Advantages: faster execution, resource sharing, modularity, aur information exchange. Simple shabdon me, cooperating processes ek team ki tarah kaam karte hain jisse overall system efficiency badhti hai.

11. Inter Process Communication (IPC)

Inter Process Communication (IPC) ek mechanism hai jisse **processes apas me data aur information exchange** karte hain. Ye zaruri hota hai, specially jab processes **cooperating** ho aur shared resources use kar rahe ho. IPC ke common methods hain: **Shared Memory** (processes same memory space use karte hain) aur **Message Passing** (processes messages ke through communicate karte hain). Simple shabdon me, IPC processes ko ek dusre ke saath coordinate aur communicate karne ka tariqa deta hai.

12. Communication in Client-Server Systems

Client-Server systems me communication **request-response model** par based hota hai. Client server ko request bhejta hai aur server us request ka response return karta hai. Communication **network protocols** jaise TCP/IP ke through hoti hai. OS aur IPC mechanisms help karte hain data exchange ko efficiently manage karne me. Simple shabdon me, client-server communication me client aur server ek dusre ke saath coordinated way me interact karte hain.

13. Multithreading Models

Multithreading me ek process ke **multiple threads** simultaneously run karte hain. Threads lightweight processes hote hain jo **same address space aur resources share** karte hain. Common multithreading models:

1. **Many-to-One** – Multiple user threads map hote hain ek kernel thread par.
2. **One-to-One** – Har user thread ek kernel thread se linked hota hai.
3. **Many-to-Many** – Multiple user threads multiple kernel threads ke saath map hote hain.

Simple shabdon me, multithreading model define karta hai ki threads ka mapping aur scheduling kaise hota hai OS me.

14. Threading Issues

Threading me kuch common problems hoti hain:

1. **Race Condition** – Jab multiple threads ek hi resource ko simultaneously access karte hain aur unexpected results aate hain.
2. **Deadlock** – Threads ek dusre ke resources ka wait karte rah jaate hain aur execution stop ho jaata hai.
3. **Starvation** – Kuch threads CPU time nahi milta aur indefinitely wait karte rahte hain.
4. **Concurrency Bugs** – Synchronization na hone ki wajah se unpredictable behavior hota hai.

Simple shabdon me, threading issues threads ke coordination aur resource sharing me problems create karte hain, isliye proper synchronization zaruri hai.

15. Scheduling Concepts

CPU Scheduling ka matlab hai decide karna ki **ready queue me kaunsa process CPU par pehle chalega**. Key concepts:

1. **CPU-bound vs I/O-bound** – CPU-bound process jyada computation karta hai, I/O-bound jyada input/output.
2. **Preemptive vs Non-preemptive** – Preemptive me process ko interrupt karke dusre ko CPU mil sakta hai, Non-preemptive me process complete hone tak CPU rakhta hai.
3. **Throughput** – Kitne processes time me complete hote hain.
4. **Turnaround Time** – Process ke submit hone se complete hone tak ka time.
5. **Waiting Time** – Ready queue me wait karne ka total time.

Simple shabdon me, scheduling concepts help karte hain **CPU ko efficiently use** karne me aur system performance improve karte hain.

16. Scheduling Algorithms

CPU Scheduling algorithms decide karte hain ki ready queue me se kaunsa process **CPU pe pehle chalega**. Common algorithms:

1. **FCFS (First-Come, First-Served)** – Pehle aaya pehle paya basis pe process execute hota hai.
2. **SJF (Shortest Job First)** – Jo process sabse kam burst time ka hai, pehle execute hota hai.
3. **Priority Scheduling** – Har process ko priority di jaati hai, higher priority process pehle execute hota hai.
4. **Round Robin (RR)** – Processes fixed time slice (quantum) ke liye CPU lete hain, phir queue ke end me chale jaate hain.
5. **Multilevel Queue** – Processes ko different priority queues me divide karke schedule kiya jaata hai.

Simple shabdon me, ye algorithms **CPU utilization, turnaround time aur waiting time** optimize karne ke liye use hote hain.

17. Algorithm Evaluation

CPU scheduling algorithms ki performance evaluate karne ke liye kuch **metrics** use hote hain:

1. **CPU Utilization** – CPU kitna effectively use ho raha hai.
2. **Throughput** – Time period me kitne processes complete hote hain.
3. **Turnaround Time** – Process submission se completion tak ka time.
4. **Waiting Time** – Ready queue me process kitna wait karta hai.
5. **Response Time** – First response milne tak ka time (important for interactive systems).

Simple shabdon me, algorithm evaluation se hum decide karte hain ki kaunsa scheduling algorithm system performance ke liye best hai.

18. Multiple Processor Scheduling

Multiple processor systems me scheduling ka matlab hai **tasks ko different CPUs ke beech distribute karna**. Do main approaches hain:

1. **Asymmetric Multiprocessing** – Ek master processor scheduling decisions leta hai, baaki processors sirf tasks execute karte hain.
2. **Symmetric Multiprocessing (SMP)** – Sabhi processors equal hote hain aur ready queue se tasks pick kar sakte hain.

Simple shabdon me, multiple processor scheduling ensure karta hai ki **CPU load balance ho aur system efficiently kaam kare**.

19. Real-Time Scheduling

Real-time scheduling me processes ko **specific time constraints** ke saath execute karna hota hai. Do types hote hain:

1. **Hard Real-Time** – Deadlines strict hote hain; miss karna system failure ho sakta hai.
2. **Soft Real-Time** – Deadlines important hote hain lekin miss hone par system fail nahi hota, sirf performance degrade hoti hai.

Simple shabdon me, real-time scheduling ka goal hai **time-bound tasks ko guarantee ke saath complete karwana**.

20. Contiguous Allocation

Contiguous allocation me process ke liye **continuous memory block** allocate kiya jata hai. Iska fayda hai ki access fast hota hai, lekin problem hoti hai **external fragmentation** ki jab free memory chhote-chhote pieces me divide ho jati hai. OS ko ye manage karna padta hai ki suitable contiguous block available ho.

Simple shabdon me, contiguous allocation me process ek hi continuous memory area me store hota hai.

21. Static Swapping

Static swapping me process ko **memory me load karne se pehle disk par temporarily store** kiya jata hai aur jab execution complete ho jata hai, wapas disk me le jaya jata hai. Ye method simple hai lekin flexibility kam hai kyunki process size aur memory location fixed hote hain.

Simple shabdon me, static swapping me process ka memory placement **pre-decided aur fixed** hota hai.

22. Overlays

Overlays ek technique hai jisme **large program ko chhote parts me divide karke memory me load** kiya jata hai. Sirf wo part memory me rehta hai jo **current execution ke liye zaruri** hai, baaki disk par rehta hai. Isse **limited memory me bhi bade programs run** kiye ja sakte hain.

Simple shabdon me, overlays help karte hain **memory efficiently use karne me** without needing full program in memory.

23. Dynamic Partitioned Memory Allocation

Dynamic partitioning me OS **memory ko process ke size ke hisaab se allocate** karta hai, fixed partition ki jagah. Jab process terminate hota hai, memory free ho jati hai aur **external fragmentation** ka problem aa sakta hai. Isko manage karne ke liye techniques jaise **compaction** use ki jaati hain.

Simple shabdon me, dynamic partitioning memory ko **flexibly aur efficiently** allocate karne ka tarika hai.

24. Demand Paging

Demand paging me process ka **sirf wahi page memory me load hota hai jo abhi execute ho raha hai**. Baaki pages disk par rehte hain aur jab required hote hain tab hi memory me laaye jaate hain. Isse **memory efficiently use hoti hai** aur unnecessary pages load nahi hote.

Simple shabdon me, demand paging me pages **on-demand** memory me load hote hain.

25. Page Replacement

Page replacement tab hota hai jab **memory full ho jaati hai aur naya page load karna ho**. OS decide karta hai kaunsa page memory se remove karna hai. Common algorithms: **FIFO (First-In-First-Out), LRU (Least Recently Used), Optimal**.

Simple shabdon me, page replacement ensure karta hai ki **memory me important pages hamesha available rahein** aur system efficiently chale.

26. Segmentation

Segmentation me memory ko **logical segments** me divide kiya jata hai jaise code, data, stack, etc. Har segment ka **different size** ho sakta hai aur segment ke liye ek **segment number aur offset** use hota hai. Ye approach programmer-friendly hoti hai aur **modularity aur protection** provide karti hai.

Simple shabdon me, segmentation me program ko **logical parts me divide karke memory efficiently manage** ki jaati hai.

27. Non-Contiguous Allocation

Non-contiguous allocation me process ke liye memory **continuous block me nahi** diya jata, balki chhote-chhote free blocks me allocate hota hai. Isse **external fragmentation ka problem kam hota hai** aur memory efficiently use hoti hai. Paging aur segmentation is approach ke common examples hain.

Simple shabdon me, non-contiguous allocation me process **different memory locations me scattered** hota hai, lekin OS usse ek logical unit ke jaise manage karta hai.

28. Paging

Paging me process memory me **fixed-size blocks** me store hota hai, jinhe **pages** kehte hain. Memory ko bhi same size ke blocks me divide kiya jata hai, jinhe **frames** kehte hain. OS pages ko frames me map karta hai using **page table**. Isse **external fragmentation khatam** ho jata hai aur memory efficiently use hoti hai.

Simple shabdon me, paging me process ke pages **anywhere memory me load ho sakte hain** aur OS unhe logical order me manage karta hai.

29. Hardware Support

Memory management ke liye hardware support me **Memory Management Unit (MMU)** aur **Translation Lookaside Buffer (TLB)** ka use hota hai. MMU logical address ko physical address me convert karta hai aur TLB frequently used addresses ko fast access ke liye store karta hai. Ye hardware features **paging aur segmentation** ko efficiently implement karne me help karte hain.

Simple shabdon me, hardware support memory ko **fast aur accurately manage** karne me crucial role play karta hai.

30. Virtual Memory

Virtual memory ek technique hai jisme OS **disk space ko RAM ki tarah use** karta hai. Isse system ko lagta hai ki **zyada memory available hai** than actual physical memory. Pages ko demand par RAM me load kiya jata hai aur baaki disk par rehta hai.

Simple shabdon me, virtual memory se **large programs chhote physical memory me bhi run** ho sakte hain efficiently.

31. A Simple File System

Simple file system me files ko **sequentially store** kiya jata hai, aur file ke liye ek **directory entry** maintain hota hai jisme file name, size aur location hoti hai. Ye approach easy hai lekin large files aur frequent updates me **inefficient** ho sakti hai.

Simple shabdon me, simple file system **basic file storage aur access** provide karta hai without complex structures.

32. General Model of a File System

General file system model me kuch main components hote hain:

1. **Files** – Data store karne ki basic unit.
2. **Directories** – Files ko organize karte hain hierarchical structure me.
3. **File Control Block (FCB)** – File ka metadata store karta hai, jaise name, type, size, location.
4. **I/O Control** – File read/write operations ko manage karta hai.

Simple shabdon me, ye model **files ko organize, access aur manage** karne ka basic framework provide karta hai.

33. Symbolic File System

Symbolic file system me files ko **logical names** diye jaate hain jo **physical storage location** se alag hote hain. Ye names **user-friendly** hote hain aur OS mapping ke through actual location ko access karta hai. Isse file management **simpler aur flexible** ho jata hai.

Simple shabdon me, symbolic file system me **users ko file ke actual location ke bare me sochne ki zarurat nahi hoti**.

34. Access Control Verification

Access control verification me OS check karta hai ki **kaun user ya process file access kar sakta hai** aur kaise (read, write, execute). Ye **permissions aur security policies** ke through manage hota hai, jaise owner, group, aur others ke liye rights define karna.

Simple shabdon me, access control verification ensure karta hai ki **sirf authorized users hi files ko access kar sake**.

35. Logical File System

Logical file system OS ka wo part hai jo **user ke liye file abstraction** provide karta hai. Ye manage karta hai **file naming, directory structure, aur access control**. User yaha se files create, delete, aur access karta hai, bina actual physical storage ke details jaane.

Simple shabdon me, logical file system **user-friendly view of files aur directories** provide karta hai.

36. Physical File System

Physical file system OS ka wo part hai jo **files ko actual storage devices** (jaise hard disk, SSD) par manage karta hai. Ye handle karta hai **blocks allocation, disk scheduling, aur data retrieval**. User ko ye details directly nahi dikhte, lekin ye OS ke liye crucial hota hai.

Simple shabdon me, physical file system **files ko hardware level par efficiently store aur access** karne ka kaam karta hai.

37. Allocation Strategy Module

Allocation strategy module decide karta hai ki **file ke liye memory blocks kaise allocate honge**. Common strategies:

1. **Contiguous Allocation** – File ke blocks sequential memory me store hote hain.
2. **Linked Allocation** – File ke blocks scattered hote hain aur pointers ke through linked hote hain.
3. **Indexed Allocation** – Ek index block ke through file ke saare blocks track kiye jaate hain.

Simple shabdon me, allocation strategy module **files ko memory me efficiently store aur manage** karne ka tarika define karta hai.

38. Device Strategy Module

Device strategy module OS ka wo part hai jo **I/O devices ke sath file operations ko manage** karta hai. Ye decide karta hai kaunsa device use hoga, data kaise transfer hoga, aur **read/write requests ko optimize** karta hai.

Simple shabdon me, device strategy module **hardware aur file system ke beech communication aur efficiency** ensure karta hai.

39. I/O Initiators

I/O initiators wo components hote hain jo **input/output operations ko start** karte hain. Ye generally **processes ya OS ke modules** hote hain jo I/O requests generate karte hain. OS fir ye requests device drivers ko forward karta hai for execution.

Simple shabdon me, I/O initiators **I/O activities ko trigger** karte hain taaki data read ya write ho sake.

40. Device Handlers

Device handlers wo modules hote hain jo **specific I/O devices ke liye operations manage** karte hain. Ye read/write requests ko process karte hain, errors handle karte hain, aur device ke sath communication coordinate karte hain.

Simple shabdon me, device handlers **I/O devices aur OS ke beech bridge** ka kaam karte hain.

41. Disk Scheduling

Disk scheduling me OS decide karta hai ki **multiple I/O requests me se kaunsa request pehle execute hoga**. Common algorithms:

1. **FCFS (First-Come, First-Served)** – Pehle aaya, pehle paya.
2. **SSTF (Shortest Seek Time First)** – Jo request disk head ke pass ho, pehle execute.
3. **SCAN / Elevator** – Disk head ek direction me move karte hue requests serve karta hai.
4. **C-SCAN** – SCAN ka circular version.

Simple shabdon me, disk scheduling **disk access ko fast aur efficient** banata hai.