Hello everyone,

these are my notes for Operating systems, gave a lot of interviews and after multiple updates I can say that almost everything you will be asked in interviews is here.

Please let me know if you come across something that is incorrect or can be improved. You can reach me at

Linkedin: https://www.linkedin.com/in/notabhishek/ Follow for more

Youtube: https://www.youtube.com/channel/UCLPT3aCJwFPWsVjEPqnOYPQ
Please subscribe so that you don't miss the content about to come

The end goal should be to able to revise within 15-20 minutes before an interview

PS: The only thing I can remember I was asked out of these was to implement multi-threading in your language of choice (mostly they want Java but they are fine with any language usually)

Operating Systems (OS)

1. What is Operating System and Its Types (MultiProcessing ,MultiTasking and All)
2. Process and Threads in OS , Process State , Process Control Block and Context Switching (**Threading is Important Topic**)
3. Process Scheduling (**All the Job Scheduling Algorithms**)
4. Important Scheduling Algos (SJF , SRTF , FCFS , LJF(Longest Job First) and **Round Robin Scheduling**)
5. Process Synchronization , Critical Section , Inter Process Communication , Locks for Synchronization (Semaphore and Mutex) and Monitors (**Important**)
6. DeadLock , Characteristics of DeadLock , Handling and Recovery from Deadlock
7. Memory Management in Operating System
8. First Fit , Best Fit, Next Fit and Worst Fit in Operating System (**Important**)
9. Paging in Memory Management (Concept of Virtual memory) (**Important**)
10. **Demand Paging** , Thrashing and Page Replacement Algo (FCFS and **LRU Algorithm**)
11. Segmentation in Memory Management and Translation Lookaside Buffer (TLB)

---

1. What is Operating System and Its Types (MultiProcessing,MultiTasking, and All)

Operating System :

a. Acts as interface between users and computer hardware
b. It is the one program that is always running on the kernel and controls the execution of application programs
c. It is concerned with
    i. Resource management - decides who gets hardware e.g. printer
    ii. Process management - using scheduling
    iii. Storage management - file systems (read more below)
    iv. Memory management - allocates/deallocates primary memory
    v. Security - unauthorized programs can't access programs/data, Windows uses **Kerberos** authentication to prevent unauthorized access to data

Types of Operating Systems :

1. Batch OS :
    a. Similar jobs i.e. those that require similar resources etc are grouped together
2. Multiprogramming OS:
    a. Multiple programs appear to run at the same time
    b. When one program goes for IO, the CPU is allocated to another program
    c. Context switching
3. Multitasking/Time-Sharing OS:
    a. Scheduling is used to share CPU

    b. Context switching + Timesharing
4. Multiprocessing OS:
    a. Multiple CPU in the same computer
    b. If N cores then N processes can run in parallel
5. Distributed OS:
    a. Use many processors located in different machines
6. Network OS:
    a. Network OS runs on a server
    b. We can manage data/users etc in the network
7. Real-Time OS :
    a. Used for performing specific tasks within deadline
    b. Used in military/space systems
8. Mobile OS:
    a. Specially designed for mobiles/tablets etc
    b. E.g Android, iOS, BlackBerry

File Systems: Link

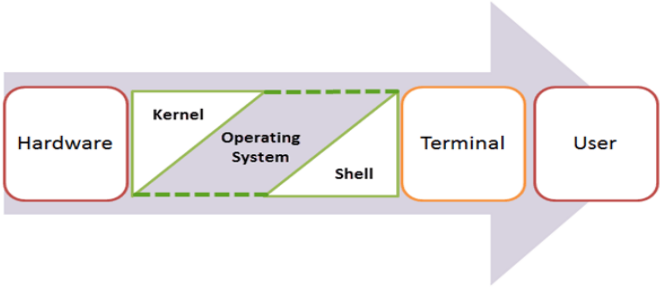
a. Process of managing how and where to store data on a storage disk
b. Windows uses NTFS, Mac uses HFS+
c. Examples :
    i. FAT: File Allocation Table
        1. Was used by older Microsoft Operating Systems
        2. Was used in Cameras, Flash memory
        3. FAT8, FAT12, FAT16, FAT32, [8-32]=bits in each cluster of the file allocation table
    ii. GFS: Global File System
        1. Allows multiple computers over large distances to act as an integrated machine
    iii. HFS: Hierarchical File System
        1. Used in MacOS
        2. Not recognized on windows since they use NTFS

iv. NTFS: New Technology File System
    1. Used in Windows NT and later versions of windows
    2. Better extendability, security, performance
v. UDF: Universal Disk Format
    1. Used in CD/DVDs
    2. Maintains consistency over data on several optical media

---

Kernel: [read more]

a. A central component of OS
b. Manages communication between software and hardware
c. Kernel is innermost part of OS, Shell is outermost



d. Features :
    i. Low-level scheduling
    ii. Interprocess communication
    iii. Process synchronization
    iv. Context switching
e. Types :
    i. Monolithic :
        1. One code block provides all the services of the os
        2. Simplistic
        3. Distinct communication layer between Hardware and Software
    ii. Microlithic :
        1. Microkernel manages all system resources
        2. Services are managed in different address space ie. user services in user address space and kernel services under kernel address space
        3. Helps reduce size of both kernel and os

OS vs FirmWare

Firmware	Operating System
Firmware is one kind of programming that is embedded on a chip in the device which controls that specific device.	OS provides functionality over and above that which is provided by the firmware.
Encoded by the manufacture of the IC or something and cannot be changed.	Can be installed by the user and can be changed
Stored on non-volatile memory.	Stored on the hard drive.

---

32 bit vs 64 bit OS [Read more]

a. 32 bit or 64 bit tells us how much memory can the CPU access from the CPU register
b. One bit in the register can reference an individual byte in memory

32 bit	64 bit
Can access 2^32 bits in register	Can access 2^64 bits in register
Max Ram = 2^32 bits * 8 (1 bit from register ref 1 byte in memory) = 4GB ram (theoretical) In practical ~3.5 GB since register also stores other stuff	Max Ram = 2^64 bits * 8 = 2^34 GB
This means that a 32 bit pc with 8GB ram cannot access ram > 3.5 GB	No limitations unless u have ram > 2^34 GB
Multiprocessing :(	Multiprocessing :) since multiple cores will need RAM

---

2. Process and Threads in OS, Process State, Process Control Block, and Context Switching (**Threading is Important Topic**)

Process Vs Thread **Read more**

Process	**Thread**
A process is any program in execution	A thread is a segment of a process(aka lightweight process)
Takes more time for creation/termination/context switching since interrupt to the kernel is required	Less time required for creation/termination/context switching
Process is isolated, blocking 1 process has no effect on another process	Threads are not isolated, 2nd thread from same parent cannot run if 1st thread is blocked
Process has its own Process Control Block, stack, address space	Threads share the PCB,stack, address space of parent and has its own Thread Control Block
Not good for communication	Better for communication since they have shared memory

---

Thread States

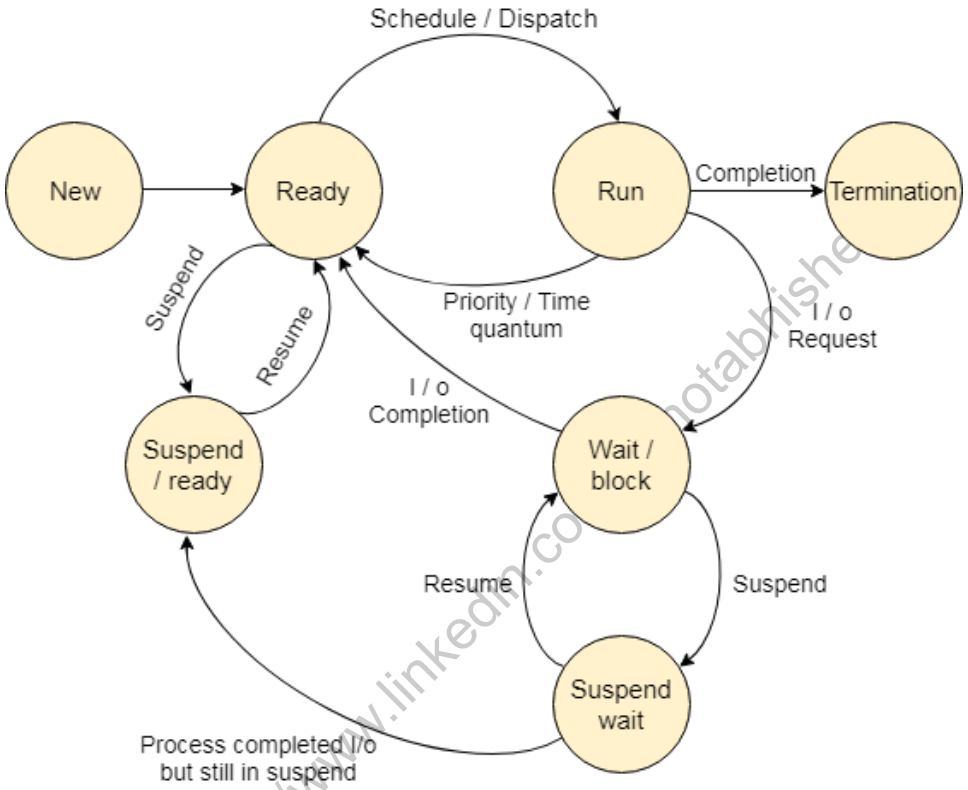
1. Ready
2. Running
3. Blocked

Process States

1. New - going to be picked by OS in main memory
2. Ready - loaded in memory and waiting for CPU
3. Running - running on cpu
4. Block or Wait - when the process is waiting for some resource like IO etc
5. Completion/Termination - all the context, PCB will be deleted and process is

terminated by OS

6. Suspend ready - higher priority process comes -> lower priority process in ready state is moved to secondary memory (suspend ready state) to make room for higher priority process
7. Suspend wait - same as Suspend ready for Block or Wait Queue



Process Control Block [Read more]

a. PCB is a data structure used to store information about a process, used to keep track of the execution status of a process
b. It stores
    i. Stack Pointer
    ii. Process State
    iii. Process Id (PID)
    iv. Program Counter
    v. Register
    vi. Memory Limit
    vii. List of Open Files etc.

c. Whenever the context of the process is switched, the PCB gets updated by the OS
d. It will help suspend and then resume the process

Process Table

a. In the case of multiple processes
b. The process table is an array of PCB for different process

Context Switching

a. It is a technique used by the OS to switch a process from one state to another
b. To perform context switching, status of the process is saved to PCB and later loaded back to resume
c. Triggers
    i. Interrupts -
        1. Interrupts are high priority requests
        2. In case of interrupts automatically context of required process is switched
    ii. Multitasking - scheduling
    iii. Kernel/User Switch - switching between Kernel and User mode

---

3. Process Scheduling (All the Job Scheduling Algorithms)

a. The OS maintains PCB Queues for process scheduling
b. There are separate queues for each process state, and processes in same state are in same queue
c. In case of context switching, process is removed from current queue and moved to queue of its new state (queues are implemented using Linked Lists)
d. Important Process scheduling queues
    i. Job queue - contains all jobs
    ii. Ready queue - jobs in ready state
    iii. Device queue - jobs waiting for resources/IO etc
e. Two-state process model
    i. Has only 2 states
    ii. Running and Not running state
f. Schedulers Read more
    i. Long Term Scheduler, Short Term Scheduler, Medium Term Scheduler
    ii.

Long Term Scheduler	Short Term Scheduler	Medium Term Scheduler
Job scheduler	CPU Scheduler	Process swapping scheduler
Decides which process	Decides which process	Removes processes from

to move from new->ready state	to move from ready->running state	suspend wait states to make room for other process
Controls degree of multiprogramming	Does not control degree of multiprogramming	Reduces degree of multiprogramming
Absent/minimal in time sharing	Absent/minimal in time sharing	There in time sharing OS

---

4. Important Scheduling Algos (SJF , SRTF , FCFS , LJF(Longest Job First) and **Round Robin Scheduling**)

Scheduling Algorithms Read more

a. Used to decide which process should be given CPU
b. Two categories
    i. **Preemptive** - the process can be preempted before execution completes
    ii. **Non-preemptive** - the process cannot be preempted before execution completes

---

c. Scheduling algorithms
    i. **FCFS**, First come first serve
    ii. **SJF**, Shortest job first, impossible in systems where time required for job is not known
    iii. **LJF**, Longest job first
    iv. **SRTF**, Shortest remaining time first (preemptive version of SJB)
    v. **Round Robin**, time quantum + context switching is used
    vi. Priority-based
    vii. **Multilevel Queues** - each queue has a priority and separate scheduling algo
    viii. **Feedback queues** - multilevel queues + feedback i.e process can move between queues(solves the starvation problem for low priority process in multilevel queues)

---

5. Process Synchronization , Critical Section , Inter Process Communication , Locks

for Synchronization (Semaphore and Mutex) and Monitors (Important)

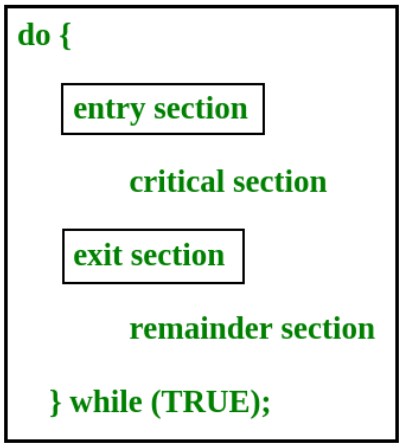
Process Synchronization Read more

a. Two types of processes based on synchronization
    i. Independent - no sync required
    ii. Cooperative - shared resources => sync required

Race condition

a. When 2 or more processes work on same shared variable/memory then the value depends on the order of execution of processes
b. Thus the processes are in a race to modify the shared variable, so the name Race condition
c. Can be avoided by treating the Critical Section as an atomic instruction

Critical Section

a. Code segment that can be assessed by only one process at a time
b. Contains shared variables/memory
c. Synchronization is required to maintain consistency of data variables
d. 

e. The solution to the critical section problem must satisfy
    i. **Mutual Exclusion**: one process at a time in CS
    ii. **Progress**: if no process is in CS, only processes not executing in the remainder section participate in deciding who goes in CS and it cannot be postponed indefinitely
    iii. **Bounded Wait**: A bound on exist before which any process will get CS after requesting to enter CS

Inter Process Communication gfg javatpoint

a. Used for cooperating processes
b. Can be achieved using 2 ways :
    i. **Shared Memory** - using shared variable/memory

ii. **Message Passing** - by passing messages through communication link (Message Queue is maintained)
    1. They provide 2 operations
        a. Send(pid/port , mssg)
        b. Receive(pid/port, mssg)
    2. Communication links
        a. Direct - via specific PID
        b. In-direct - via shared mailbox(PORT) sender puts mail , receiver picks them up

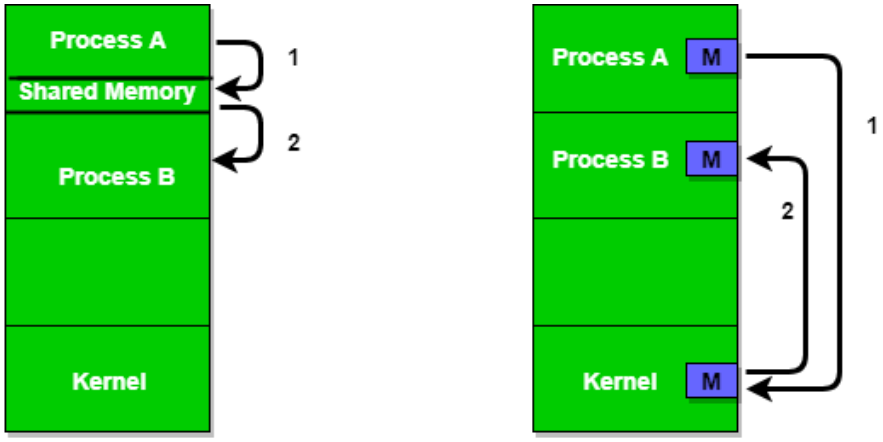


Figure 1 - Shared Memory and Message Passing

c.

d. Examples of IPC systems
    i. Posix uses Shared Memory
    ii. Mach uses Message Passing
    iii. Windows uses Message Passing using local procedural calls
e. Communication in Client/Server Architecture -
    i. **Pipe** -
        1. unidirectional data channel , 2 are used for 2 way communication,
        2. Only work for processes in same hierarchy
        3. Can be used to communicate between parent and child process
    ii. **FIFO** - Video
        1. Aka “Named Pipe”
        2. Fifo are special file types used for communication, processes can rw to fifo for communication
        3. full duplex general communication between 2 unrelated processes
        4. Can be used to communicate between unrelated processes
    iii. **Socket** - Video
        1. acts as an endpoint for communication in a network
        2. A pair of processes communicating over a network employ a pair of sockets, one for each process
        3. A socket is identified by an ip address concatenated with a port number
    iv. **RPCs (Remote Procedural Calls)**

1. Query response model
2. Thread of execution jumps back and forth, client requests, server receives, server sends, client receives
3. Extension of local procedural calls

f. Benefits of IPC
   i. Computational
   ii. Privilege separation
   iii. Convenience
   iv. Helps operating system to communicate with each other and synchronize their actions as well

---

Locks for Synchronization (Semaphores vs Mutex) Read more gfg

a. **Semaphores**
   i. A variable that is non-negative and is shared between threads
   ii. Semaphore is a **signaling mechanism**, uses two atomic operations
      1. wait()
      2. signal()
   iii. 2 types
      1. Binary semaphore (similar to a mutex), allows only 1 thread in CS
      2. Counting semaphore allows n threads in CS

b. **Mutex**
   i. Full form: Mutual exclusion objects
   ii. Mutex is a **locking mechanism**
      1. lock(), acquire resource
      2. unlock() , release resource
   iii. Similar to a binary semaphore but mutex has ownership associated with it
   iv. When a thread enters CS it acquires ownership of the lock/locks for multiple resources, there is no concept of ownership in semaphores
   v. There are recursive mutex that can be locked more than once, have to be unlocked N times if locked N times
   vi. If a thread locks a non-recursive mutex and then tries to lock it again, a **deadlock** occurs since it will wait for the thread to unlock but no one can unlock it other than the same thread

c. **Monitors Read more Video**
   i. Similar to class provides high-level of synchronization
   ii. At a time only one process can enter into the monitor

iii.

```
Monitor Demo // Name of the Monitor
{
variables;
condition variables;
procedure p1 {.....}
procedure p2 {.....}
}
```

iv. **Condition variables** work like semaphores, i.e. condition a; then you can perform
    1. a.wait()
    2. a.signal()

v. If thread T1 enters a monitor and does a.wait() and then thread T2 tries to enter the monitor then T2 is sent to **Block queue/Monitor Entry Queue**

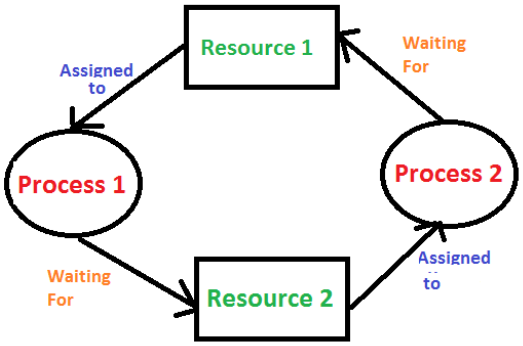
vi. Monitors have 4 parts
    1. Initialization (runs only once)
    2. Private data (not accessible directly, only procedures can access)
    3. Procedures (methods/functions)
    4. Monitor Entry Queue (threads waiting to enter the monitor)

---

6. DeadLock , Characteristics of DeadLock , Handling and Recovery from Deadlock

a. Deadlock

i. Deadlock is a situation when a set of processes are blocked because each process is holding some resources and waiting for resources acquired by other processes



ii.

iii. **4 necessary conditions for Deadlock/deadlock** characteristics

1. **Mutual Exclusion** - one or more resources are non-shareable
2. **Hold and Wait** - the process is holding at least one res and waiting for more
3. **No Preemption** - a res cannot be preempted from a process by force
4. **Circular wait** - set of processes are waiting for each other in circular form

---

b. **Handling Deadlock**

i. **Deadlock prevention**

1. Prevention is done by negating one of the necessary conditions for deadlock
2. **Bankers Algo** Video
   a. Based on Dijkstra's algorithm
   b. We need to have information about the maximum demand of each process (futuristic in nature)
   c. We try to run processes assuming worst case (maximum demand of each process) and try to avoid deadlock
   d. If there is a sequence in which processes can be run successfully without having deadlock it is known as **Safe Sequence**
   e. Safe sequence exists -> no possibility of deadlock, safe sequence does not exist -> there may be deadlock, not necessary though

ii. **Deadlock detection and recovery**

1. Let deadlock occur
2. Do preemption and recover

iii. **Ignore and reboot**

1. Both windows and Unix do this since deadlock is very rare

---

7. Memory Management in Operating System [Read more]

1. We maintain a hierarchy of memory i.e. CPU - cache - main - secondary memory
2. Access time in secondary memory is high (large size) so we use **Locality of reference** i.e. prefetch data in the locality of currently executing instruction to main memory (lower access time) since they have a high probability of getting executed next
3. Using this average access time can be reduced, avgT = hit% * (mainT) + miss% * (mainT+secT)
4. CPU works with **Logical Address**(secondary memory) but we have prefetched data in main memory **Physical Address**(primary memory) and so we need to translate from one address to another
5. **Memory Allocation**
    a. **Contiguous**
        i. The fetched process is allocated contiguous memory in the primary memory
        ii. Advantage: very fast access time and address translation
        iii. Disadvantage: **External Fragmentation** i.e the required space is available but not in a contiguous way, thus even after having space you cannot load the process in the main memory
    b. **Non Contiguous**
        i. Slow access time, like linked lists, each block store pointer to next block
        ii. Free from External fragmentation
6. **Fragmentation** [Read more]
    a. **Internal Fragmentation**
        i. When fixed-size blocks are allocated to process, there may be internal space left i.e. a block of 10kb allocated to a process requiring 9kb so 1kb hole, this is internal fragmentation
    b. **External Fragmentation**
        i. The required amount of memory exists but not in a contiguous manner
        ii. Caused by best fit, first fit, etc.

---

8. First Fit , Best Fit, Next Fit and Worst Fit in Operating System (Important)

1. **Memory Allocation**
    a. **Single contiguous allocation**
    b. **Partitioned allocation** - parts of contiguous memory
        i. Paging
            1. Fixed size pages / page frames
            2. Every page is mapped to some frame in secondary memory
        ii. Segmentation
            1. Only memory management that does not provide the user with a contiguous address space

2. **Segment Table** is maintained
3. Segment table stores the physical address in secondary memory, size of segment, access protection bits etc

c. The main memory is divided into two parts
   i. Low memory - os resides here
   ii. High memory - user processes are held here

2. Allocation schemes

a. When a new process is to be fetched it is allocated memory by using one of the following schemes
   i. **First fit** - first sufficient block of memory is allocated
   ii. **Best fit** - first smallest block of memory that is sufficient
   iii. **Next fit** - like first fit, but searches after the last allocated block
   iv. **Worst fit** - largest block is allocated

---

9. Paging in Memory Management (Concept of Virtual memory) (Important)

a. Paging Read more

i. Address
1. Logical/Virtual address - address generated by CPU
2. Logical/Virtual address space - set of all addresses gen by a program
3. Physical address - address actually available on the memory unit
4. Physical address space - set of all physical addresses corresponding to logical addresses

ii. Mapping from Logical to Physical address is done by MMU(Memory management unit) which is a hardware device and this is known as the paging technique

iii. The **physical address space** is conceptually divided into fixed-size **frames**

iv. The **logical address space** is divided into fixed-size **pages**

v. **Page size = Frame size**

vi. Logical address
1. Page Number
2. Page Offset - word number on the page

vii. Physical address
1. Frame Number
2. Frame Offset - word number on the frame

---

10. Demand Paging, Thrashing, and Page Replacement Algo (FCFS and LRU Algorithm)

a. **Virtual Memory**
   i. When a program is executing using locality of reference only a few pages are loaded on the main memory
   ii. A **Page table** is maintained which stores which page is stored on which frame of main memory
   iii. If CPU requests for a page that is not present on main memory, a **page fault** occurs, and then it will be loaded into main memory
   iv. This is known as **Demand Paging** i.e. pages are loaded on demand
   v. If we have no free frames on main memory then we have to replace a page with the new page, aka **Lazy swapping**(wait for demand and then swap), for this we use page replacement algorithms
   vi. Because by using Demand Paging the user feels as if the whole process is loaded in main memory and feels as if he is working with large amount of memory, this is the concept of **Virtual Memory**
   vii. Demand paging helps **increase the degree of multiprogramming** since multiple programs can be loaded into the main memory

b. **Thrashing**
   i. As we go on increasing the degree of multiprogramming, the new programs will get lesser and lesser frames on the main memory
   ii. The probability of page fault occurrence is very high
   iii. The CPU is busy resolving page faults and not executing the programs
   iv. Therefore there is a drastic increase in page faults and decrease in CPU utilization while increasing the degree of multiprogramming, due to unavailability of sufficient frames on main memory, this is known as **Thrashing**

c. **Page Replacement Algorithms**
   i. **Page Fault** :
      1. A page fault occurs when a running program accesses a page that is mapped into virtual address space but not loaded in physical memory
   ii. Replacement algorithms
      1. **FIFO/FCFS** -
         a. The oldest page in front of the queue is removed
         b. **Belady's anomaly -** in the FIFO algorithm, if you increase the number of page slots, the number of page faults might still increase even when we have more slots so it should decrease
      2. **Optimal** -
         a. The page which will not be used for the longest duration in future is removed
         b. Requires info from future, not possible in real-world since we don't know which requests might come in future
      3. **LRU** (Least recently used)
         a. The least recently used page is removed

11. Segmentation in Memory Management and Translation Lookaside Buffer (TLB)

a. Segmentation

i. A process is divided into segments which are not necessarily of the same size
ii. Segmentation gives user's view of the process which paging does not give
iii. 2 types
    1. **Simple segmentation**
        a. All segments are loaded into physical memory at runtime, not necessarily contiguously
    2. **Virtual memory segmentation**
        a. Non all the segments of process are loaded into physical memory
iv. **Segment Table** is maintained for translation of logical address to physical address
    1. Address from CPU i.e. Logical/Virtual address is divided into
        a. Segment Number
        b. **Segment Offset** - word number in the segment
v. **Advantages**
    1. **No Internal Fragmentation**
    2. Segment table takes less space in comparison to page table
vi. **Disadvantage**
    1. **External Fragmentation** occurs since memory has holes when segments are removed.

---

b. Translation Lookaside Buffer (TLB)

i. Initially, registers were used to store Page Tables since they were fast
ii. But registers have a small capacity, hence page tables are stored in main memory, thus access time increases
iii. To solve this problem, we use TLB, it is a **special high-speed cache memory**
iv. It **stores recently used transaction**
v. When a running process requests a page, CPU generates a logical address and we search for the mapped physical address in TLB
    1. **TLB hit** - we know the physical address of the page and directly access it
    2. **TLB miss** - the corresponding physical address is searched in Page Table stored in main memory and then the page is loaded, and then TLB is updated
vi. TLB helps **reduce EMAT** (Effective Memory Access Time)

1. **EMAT = h*(c+m) + (1-h)*(c+2m)**
2. Here, h = hit ratio
3. C = TLB access time
4. M = Memory access time
