**CS Fundamental -> OS**

# SYLLABUS

1-UNIT 1 **Introduction to OS:**

**U1.1.**

Operating Systems Objectives and functions, Components of OS, OS Structure, Evolution of Operating Systems - Simple Batch, Multiprogramming, Time shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, Operating System services, System Calls.

**U1.2. Self Study:** System Programs, System structure, Virtual Machines, Dual Mode Operation.

2. UNIT 2 **Process Management:**

**U2.1**

Process and CPU Scheduling - Process concepts – Process and Process States, Process Control Block, Cooperating Processes, Inter-process Communication, Process Scheduling - Scheduling Queues, Schedulers, Context Switch, Non Pre-emptive and Pre-emptive Scheduling, Dispatcher, Schedulability Criteria, Scheduling algorithms.

Process Coordination: Process Synchronization, The Critical section Problem, Peterson's solution, Synchronization Hardware, Semaphores, and Classical Problems of Synchronization, Monitors. Threads and its type.

**U2.2. Self Study:** Process creation mechanism and scheduling algorithms used in Linux and Windows.

3. UNIT 3 **Deadlock:**

**U3.1**

Deadlocks: System model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock avoidance, Deadlock Detection, recovery from Deadlock.

**U3.2.** **Self Study**: Thread creation and Thread scheduling in Linux and Windows

4. UNIT 4 **File System & Memory Management**

**U4.1**

Memory Management strategies, Background, Logical versus Physical Address space, MMU, Address Translation, Swapping, Contiguous Allocation, Paging, Segmentation.

Virtual Memory: Background, Demand paging, Page Replacement concepts, Page Replacement Algorithms. Allocation of frames, Thrashing, Segmentation with Paging,

**U4.2. Self Study:** Multilevel Paging, Inverted Page Table, Demand Segmentation, Case study using Linux and Windows.

5. UNIT 5 **File System & Storage Management**

**U5.1.**

File System Interface - The Concept of a File, Access methods, Directory Structure, File System Mounting, File Sharing, Protection, File System Structure, File System Implementation, Allocation methods.

I/O Systems- Overview of Mass Storage Structure, Device Drivers, Disk Structure, Disk Scheduling, Disk Management, and Swap space Management, Free-space Management, Directory Implementation, RAID Structure.

**U5.2. Self Study:** Disk Attachment, Stable Storage Implementation, Case studies on File system: LINUX and Windows.

**(\*) DON’T MAKE NOTES OF OS BECAUSE ITS VERY TIME CONSUMING AND NOT NECESSARY INSTEAD OF THIS YOU CAN LEARN INTERVIEW QUESTIONS AND MOST ASKED QUESTIONS OR HANDWRITTEN**

1. What is the main purpose of an operating system? Discuss different types?

2. What is a socket, kernel and monolithic kernel ?

3. Difference between process and program and thread? Different types of process.

4. Define virtual memory, thrashing, threads.

5. What is RAID ? Different types.

6. What is a deadlock ? Different conditions to achieve a deadlock.

7. What is fragmentation? Types of fragmentation. 8. What is spooling ?

9. What is semaphore and mutex (Differences might be asked)? Define Binary semaphore.

10. Belady’s Anomaly

11. Starving and Aging in OS

12. Why does trashing occur?

13. What is paging and why do we need it?

14. Demand Paging, Segmentation

15. Real Time Operating System, types of RTOS.

16. Difference between main memory and secondary memory.

17. Dynamic Binding

18. FCFS Scheduling

19. SJF Scheduling

20. SRTF Scheduling

21. LRTF Scheduling

22. Priority Scheduling

23. Round Robin Scheduling

24. Producer Consumer Problem

25. Banker’s Algorithm

26. Explain Cache

27. Diff between direct mapping and associative mapping

28. Diff between multitasking and multiprocessing

Deadlock 32 page

# 

# UNIT 1 Introduction to OS:

### **Operating System**

An operating system is a program that manages the computer hardware.

An operating system can be defined as an interface between user and hardware the purpose of operating system is to provide an environment in which a user can execute program in a convenient and efficient manner

### Objective of operating system

**1) Convenience** -An OS makes and computer more convenient to the user for using GUI

**2)Efficiency** - An OS allow the computer system resource to be used in any efficient manner to ensure good resource utilization efficiency

**3)Ability to evolve** -An OS should be constructed in such a way as to permit the effective development testing and introduction of new system functions without interfering the service

### Function of Operating system services [function](https://www.youtube.com/watch?v=at2oc4AYsTQ&ab_channel=ashakhilrani)

* 1. Resource management - decides who gets hardware e.g. printer
  2. Process management - using scheduling
  3. Storage management - file systems ( read more below )
  4. Memory management - allocates/deallocates primary memory
  5. Security - unauthorized programs can’t access programs/data, Windows uses **Kerberos** authentication to prevent unauthorized access to data

1. Process Management => The OS decides the order in which processes have access to the processor, and how much processing time each process has. This function of OS is called process scheduling When CPU is free operating system select process from job queue and allocate in the CPU to the process when process execution will complete operating system free the processor and again select another process for execution

2. File Management => file is stored in secondary storage

3. Memory management =>

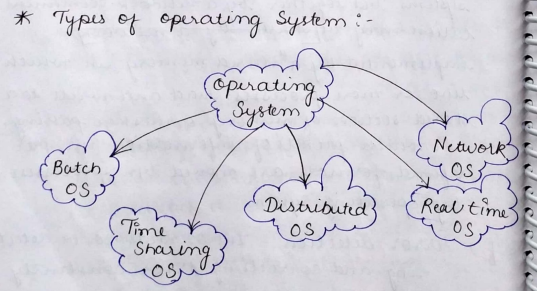
4. security management =>

5. input output device management =>

# 

# 

### Types of Operating System



# 

**1.Batch OS** – The user of this OS does not interact with the computer directly.

Users prepare job in punch card and submit to operator.

Job is an unit of work that contains program input data and control instructions

Operator collect the jobs from user short the jobs and group similar jobs into batches

Batches are inserted into the OS and then executed only when the execution of the previous job completed

Disadvantage of batch OS

CPU is often idol because the speed of input output mechanical device is slower than CPU

lack of interaction between the user and the job

**2. Multiprogramming OS** – The main memory consists of jobs waiting for CPU time. The OS selects one of the processes and assigns it to the CPU. Whenever the executing process needs to wait for any other operation (like I/O), the OS selects another process from the job queue and assigns it to the CPU. This way, the CPU is never kept idle and the user gets the flavor of getting multiple tasks done at once.

**3. Multitasking OS** – Multitasking OS combines the benefits of Multiprogramming OS and CPU scheduling to perform quick switches between jobs. The switch is so quick that the user can interact with each program as it runs.

**4. Time Sharing OS** – Time-sharing systems require interaction with the user to instruct the OS to perform various tasks. The OS responds with an output. The instructions are usually given through an input device like the keyboard.

**Advantage**

- quick response -avoid lubrication of software -reduce CPU idle time

**Disadvantages**

Problem of reliability - problem of data communication

**5. Real Time OS** – Real-Time OS are usually built for dedicated systems to accomplish a specific set of tasks within deadlines or with time constraints.

**6. Distributed operating system**

It use multiple Central processor to serve multiple real time application and multiple user

Data processing jobs are distributed among the processor according

**Function of distributed OS**

load balancing

computer science speed up

access to different hardware configuration

**7. Network OS**- a network OS run on a server and provide the server capability to manage data user group security application and other networking application

**Advantage**

highly stable centralized server

security concern are through service

New technology and hardware upgradation are easily integrated to the system

**Disadvantage**

Service are costly maintenance and updates are required regularly

### System call

A system call is a method for a computer program to request a service from the kernel of the [operating system](https://www.javatpoint.com/os-tutorial) on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.

# 

# 

# 

# UNIT 2 Process Management:

Process => A program in execution is called a process.

Program contains the instructions to be executed by processor

Following activities of Process management

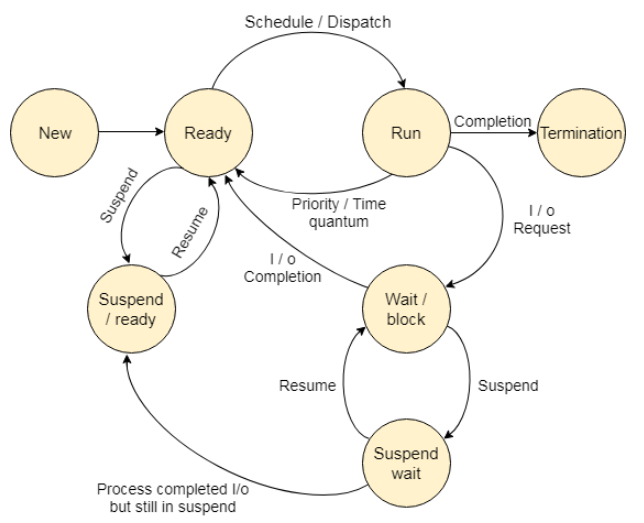
1. Scheduling processes and threads on the CPUs.
2. Creating and deleting both user and system processes.
3. Suspending and resuming processes.

**Process Vs Thread** [**Read more**](https://www.geeksforgeeks.org/difference-between-process-and-thread/)

| **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 |

### Process States

The process, from its creation to completion, passes through various states. The minimum number of states is five.



1.New - The process is being created.

2. Ready - The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.

3. Running - Process instructions are being executed (i.e. The process that is currently being executed).

4. Waiting - The process is waiting for some event to occur (such as the completion of an I/O operation).

5. Terminated - The process has finished execution.

### Process Control Block [Read more](https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/)

1. PCB is a data structure used to store information about a process, used to keep track of the execution status of a process - Each process in the Os is represented by PCB. It also known as Task control block.
2. It stores
   1. Stack Pointer
   2. Process State
   3. Process Id (PID)
   4. Program Counter
   5. Register
   6. Memory Limit
   7. List of Open Files etc.
3. Whenever the context of the process is switched, the PCB gets updated by the OS
4. It will help suspend and then resume the process

**Context Switching**

Switching CPU from one process to another process requires performing a state safe of the current process and a state is stored of the different process this task is known as context switching.

When context switch occurs,

- the kernel save the context of the old process in its PCB and

- load the saved context of the next process schedule to run

-context switch time depends machine to machine depending on the memory Speed the number of registers at must be copy it and the existence of special instruction

### Thread in OS

A thread is a path of execution within a process a process can contain multiple threats it is also known as light weight process

Multi threading.

The idea is to achieve parallelism by dividing a process into multiple threads for example in a browser multiple tabs can be different thread MS word uses multiple threads 1 tab to format the text another thread to process input etc

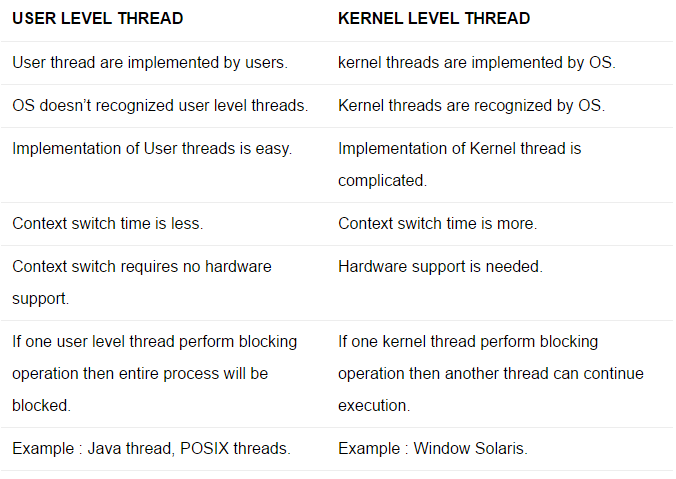
**Advantage**

Thread minimizes context switching time.

Use of threads provides concurrency within a process.

Efficient communication

\* Types of Threads:-



### **Process scheduling**

the activity of the process manager that handles the removal of the running process from the CPU and the selection of under process on the basis of a particular strategy

Main objective of scheduling

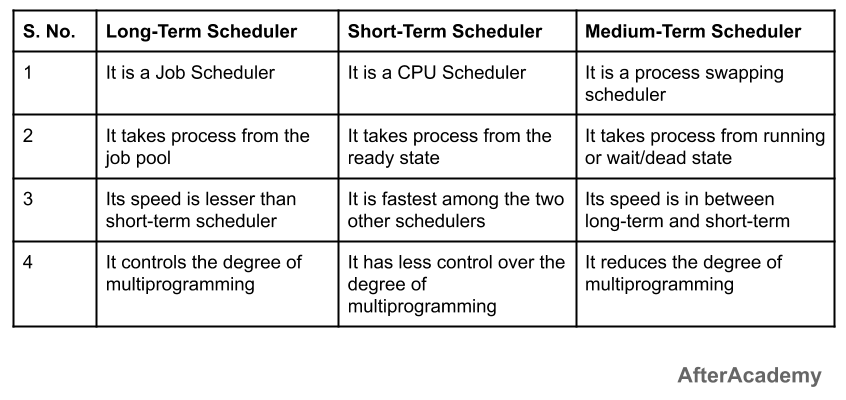
1. Make the system fast

2. Maximize throughput

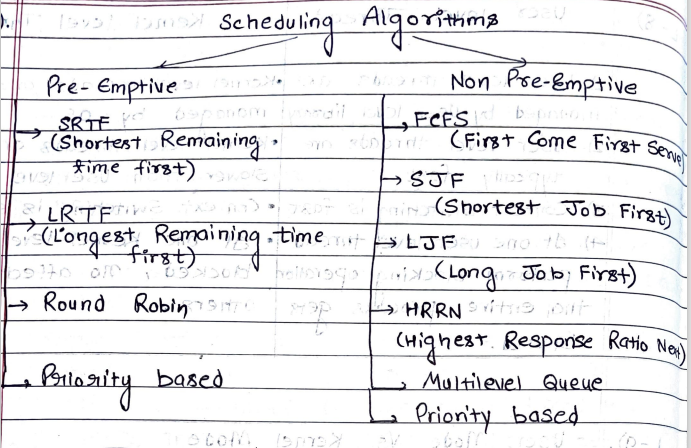
3. Increase the output

**Scheduler** the main task is to select the job to be submitted into the system and to decide which process to run

It has three types long term, short term, medium term



**In Multiprogramming systems**, the Operating system schedules the processes on the CPU to have the maximum utilization of it and this procedure is called **CPU scheduling**. The Operating System uses various scheduling algorithm to schedule the processes



● Process Scheduling:

1. Arrival Time – Time at which the process arrives in the ready queue.

2. Completion Time – Time at which process completes its execution.

3. Burst Time – Time required by a process for CPU execution.

4. Turn Around Time – Time Difference between completion time and arrival time. Turn Around Time = Completion Time - Arrival Time

5. Waiting Time (WT) – Time Difference between turn around time and burst time. Waiting Time = Turnaround Time - Burst Time

**Scheduling Algorithms**[Read more](https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm)

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

● Scheduling Algorithms :

1. **First Come First Serve (FCFS) :** Simplest scheduling algorithm that schedules according to arrival times of processes.

2. **Shortest Job First (SJF):** Processes which have the shortest burst time are scheduled first.

3. **Shortest Remaining Time First (SRTF):** It is a preemptive mode of SJF algorithm in which jobs are scheduled according to the shortest remaining time.

4. **Round Robin (RR) Scheduling:** Each process is assigned a fixed time, in a cyclic way.

5. **Priority Based scheduling (Non Preemptive):** In this scheduling, processes are scheduled according to their priorities, i.e. The highest priority process is scheduled first. If priorities of two processes match, then scheduling is according to the arrival time.

6. **Highest Response Ratio Next (HRRN):** In this scheduling, processes with the highest response ratio are scheduled. This algorithm avoids starvation. Response Ratio = (Waiting Time + Burst time) / Burst time

7. **Multilevel Queue Scheduling (MLQ):** According to the priority of the process, processes are placed in the different queues. Generally high priority processes are placed in the top level queue. Only after completion of processes from the top level queue, lower level queued processes are scheduled.

8. **Multilevel Feedback Queue (MLFQ) Scheduling:** It allows the process to move in between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it is moved to a lower-priority queue.

### Process Synchronization

When two or more process cooperates with each other, their order of execution must be preserved otherwise there can be conflicts in their execution and inappropriate outputs can be produced.

Such processes need to be synchronized so that their order of execution can be guaranteed.

The procedure involved in preserving the appropriate order of execution of cooperative processes is known as Process Synchronization. There are various synchronization mechanisms that are used to synchronize the processes.

1. Two types of processes based on synchronization
   1. Independent - no sync required
   2. Cooperative - shared resources => sync required

**Critical section**

Critical the segment of code or the program which tries to access or modify the value of the variables in a shared resource.

The regions of a program that try to access shared resources and may cause race conditions are called critical section. To avoid race condition among the processes, we need to assure that only one process at a time can execute within the critical section.

**Race around section**

A Race Condition typically occurs when two or more threads try to read, write and possibly make the decisions based on the memory that they are accessing concurrently.

A solution for the critical section problem must satisfy the following three conditions:

**1. Mutual Exclusion** – If a process Pi is executing in its critical section, then no other process is allowed to enter into the critical section.

**2. Progress** – If no process is executing in the critical section, then the decision of a process to enter a critical section cannot be made by any other process that is executing in its remainder section. The selection of the process cannot be postponed indefinitely.

**3. Bounded Waiting** – There exists a bound on the number of times other processes can enter into the critical section after a process has made a request to access the critical section and before the request is granted.

**Inter Process Communication**

1. Used for cooperating processes
2. Can be achieved using 2 ways :
   1. **Shared Memory** - using shared variable/memory
   2. **Message Passing** - by passing messages through communication link ( Message Queue is maintained)

### Synchronization Tools:

**1. Semaphore :** Semaphore is a protected variable or abstract data type that is used to lock the resource being used. The value of the semaphore indicates the status of a common resource.

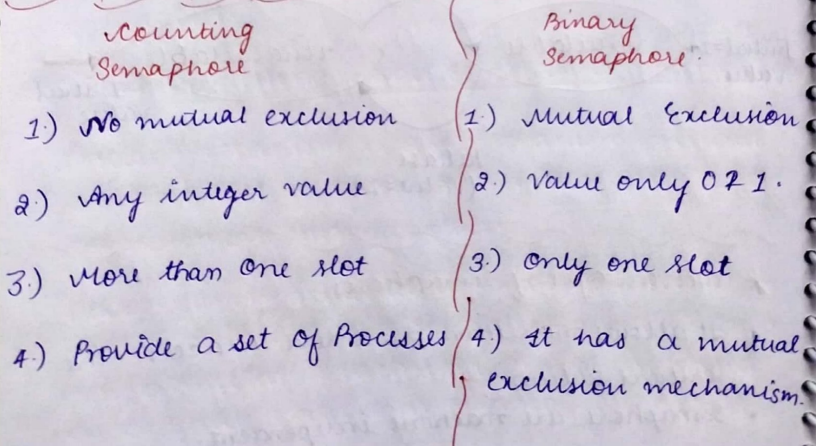
Semaphore is a **signaling mechanism**, uses two atomic operations

* + 1. wait()
    2. signal()

There are two types of semaphores:

-Binary semaphores (Binary semaphores take only 0 and 1 as value and are used to implement mutual exclusion and synchronize concurrent processes.)

-Counting semaphores (A counting semaphore is an integer variable whose value can range over an unrestricted domain.)

****

**2)Mutex**

-Full form: Mutual exclusion objects

-Mutex is a **locking mechanism**

* + 1. lock(), acquire resource
    2. unlock() , release resource

-Similar to a binary semaphore but mutex has ownership associated with it

-When a thread enters CS it acquires ownership of the lock/locks for multiple resources, there is no concept of ownership in semaphores

-There are recursive mutex that can be locked more than once, have to be unlocked N times if locked N times

### 

### Monitors [Read more](https://www.tutorialandexample.com/monitors-in-operating-system/) [Video](https://www.youtube.com/watch?v=rz5CKJHhalU)

Similar to class provides high-level of synchronization

At a time only one process can enter into the monitor

# UNIT 3 Deadlock:

A deadlock is **a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function**.

### Deadlock can arise if following four conditions hold simultaneously

1. **Mutual Exclusion –** One or more than one resource is non-sharable (Only one process can use at a time).

2. **Hold and Wait –** A process is holding at least one resource and waiting for resources.

3. **No Preemption –** A resource cannot be taken from a process unless the process releases the resource.

4. **Circular Wait –** A set of processes are waiting for each other in circular form.

### Methods for handling deadlock:

There are three ways to handle deadlock

1. **Deadlock prevention or avoidance :** The idea is to not let the system into a deadlock state.

2. **Deadlock detection and recovery :** Let deadlock occur, then do preemption to handle it once occurred.

3. **Ignore the problem all together :** If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

### Banker's algorithm

is used to avoid deadlock. It is one of the deadlock-avoidance methods. It is named as Banker's algorithm on the banking system where a bank never allocates available cash in such a manner that it can no longer satisfy the requirements of all of its customers.

# UNIT 4 Memory Management

Memory management is a functionality of OS to manage the various kind of memory like Register- case-main memory - secondary memory and keep track of process moving form SM to MM and vice versa is called memory management.

Functionalities

1. Allocate and deallocate memory when needed and when they not required.

2. Managing the transfer of “memory “ RAM & Disk.

Goal - Maximum CPU utilization - Minimize the response time

Cpu is use of run the process, before this program must be store somewhere we can keep our program in **Register** because its the access time is less but register size is small so cant store big program so for this there is **Secondary memory** in it there is lots of space but access time is high so its not suitable to use

At last **Main memory** is the best solution for us because it can store the program and also less access time

- we fetch the Program/data from secondary memory and insert into main memory and this will help in execution of the process.

1. 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

### 

### Memory Allocation

* 1. **Contiguous**
     1. The fetched process is allocated contiguous memory in the primary memory
     2. Advantage: very fast access time and address translation
     3. 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
  2. **Non Contiguous**
     1. Slow access time, like linked lists, each block store pointer to next block
     2. **Free from External fragmentation**

### 

### Fragmentation [Read more](https://www.geeksforgeeks.org/difference-between-internal-and-external-fragmentation/)

Fragmentation is a phenomenon of memory wastage. It reduces the capacity and performance because space is used inefficiently.

**1. Internal fragmentation:** It occurs when we deal with the systems that have fixed size allocation units.

Ex- 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

**2.External fragmentation:** It occurs when we deal with systems that have variable-size allocation units.

* The required amount of memory exists but not in a contiguous manner
* Caused by best fit, first fit, etc.

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

1. **Memory Allocation**
   1. **Single contiguous allocation**
   2. **Partitioned allocation** - parts of contiguous memory
      1. **Paging**
         1. Fixed size pages / Fixed size of frames
         2. Every page is mapped to some frame in secondary memory
      2. **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

* 1. The main memory is divided into two parts
     1. Low memory - os resides here
     2. High memory - user processes are held here

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

### Paging [Read more](https://www.geeksforgeeks.org/paging-in-operating-system/)

Paging can be defined as, The Process is divided into fixed size of pages and The Main memory is divided into fixed size of frames, every Pages mapped into the main memory frames by memory management unit. Is called paging technique

OR

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

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

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

**3. Page size = Frame size.**

| **Logical Address** | **Physical Address** |
| --- | --- |
| 1. Address generated by the CPU.  2.User program deal with the logical address directly.  3.The set of all logical address generated by a program are known as logical address space.  4.The logical address space is divided into fixed-size pages. | 1. Address actually available on the memory unit.  2. User program cannot deal with physical address directly.  3. The set of all physical address is known as the physical address space.  4.The **physical address** **space** is conceptually divided into fixed-size **frames.** |

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

# 

# 

# UNIT 5 **Storage Management**

### Virtual Memory

* 1. When a program is executing using locality of reference only a few pages are loaded on the main memory
  2. A **Page table** is maintained which stores which page is stored on which frame of main memory
  3. 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
  4. This is known as **Demand Paging** i.e. pages are loaded on demand
  5. 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
  6. 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**
  7. Demand paging helps **increase the degree of multiprogramming** since multiple programs can be loaded into the main memory

### Thrashing

* 1. As we go on increasing the degree of multiprogramming, the new programs will get lesser and lesser frames on the main memory
  2. The probability of page fault occurrence is very high
  3. The CPU is busy resolving page faults and not executing the programs
  4. 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**

### Page Replacement Algorithms

* 1. **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
  2. Replacement algorithms
     1. **FIFO/FCFS** -
        1. The oldest page in front of the queue is removed
        2. **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** -
        1. The page which will not be used for the longest duration in future is removed
        2. 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)
        1. The least recently used page is removed

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

### Segmentation

* 1. A process is divided into segments which are not necessarily of the same size
  2. Segmentation gives user’s view of the process which paging does not give
  3. 2 types
     1. **Simple segmentation**
        1. All segments are loaded into physical memory at runtime, not necessarily contiguously
     2. **Virtual memory segmentation**
        1. Non all the segments of process are loaded into physical memory
  4. **Segment Table** is maintained for translation of logical address to physical address
     1. Address from CPU i.e. Logical/Virtual address is divided into
        1. **Segment Number**
        2. **Segment Offset** - word number in the segment
  5. **Advantages**
     1. **No Internal Fragmentation**
     2. Segment table takes less space in comparison to page table
  6. **Disadvantage**
     1. **External Fragmentation** occurs since memory has holes when segments are removed.

### Translation Lookaside Buffer (TLB)

* 1. Initially, registers were used to store Page Tables since they were fast
  2. But registers have a small capacity, hence page tables are stored in main memory, thus access time increases
  3. To solve this problem, we use TLB, it is a **special high-speed cache memory**
  4. It **stores recently used transaction**
  5. 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
  6. TLB helps **reduce EMAT** (Effective Memory Access Time)

**1.** **EMAT = h\*(c+m) + (1-h)\*(c+2m)**

* + 1. Here, h = hit ratio
    2. C = TLB access time
    3. M = Memory access time