Operating Systems

- An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

- ## Types of Operating Systems :

1. **Batch OS –** A set of similar jobs are stored in the main memory for execution. A job gets assigned to the CPU, only when the execution of the previous job completes.
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.
5. **Real Time OS –** Real-Time OS are usually built for dedicated systems to accomplish a specific set of tasks within deadlines.

- **Process** : A process is a program under execution. **The value of the program counter (PC) indicates the address of the next instruction of the process being executed**. Each process is represented by a Process Control Block (PCB).

- **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

- **Thread (Important) :** A thread is a lightweight process and forms the basic unit of CPU utilization. A process can perform more than one task at the same time by including multiple threads.
  - A thread has its own program counter, register set, and stack
  - A thread shares resources with other threads of the same process: the code section, the data section, files and signals.

Note : A new thread, or a child process of a given process, can be introduced by using the fork() system call. A process with n fork() system call generates $2^n – 1$ child processes.

There are two types of threads:

- User threads (User threads are implemented by users)
- Kernel threads (Kernel threads are implemented by OS)

- **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., 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.

- **The Critical Section Problem**:
    1. **Critical Section –** The portion of the code in the program where shared variables are accessed and/or updated.
    2. **Remainder Section –** The remaining portion of the program excluding the Critical Section.
    3. **Race around Condition –** The final output of the code depends on the order in which the variables are accessed. This is termed as the race around condition.

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.

- **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.

       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.)

Mutex (A mutex provides mutual exclusion, either producer or consumer can have the key (mutex) and proceed with their work. As long as the buffer is filled by the producer, the consumer needs to wait, and vice versa.
At any point of time, only one thread can work with the entire buffer. The concept can be generalized using semaphore.)

- **Deadlocks** **(Important)**:

A situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Deadlock can arise if following four conditions hold simultaneously (Necessary Conditions):

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.

- **Memory Management:** These techniques allow the memory to be shared among multiple processes.
  - **Overlays –** The memory should contain only those instructions and data that are required at a given time.
  - **Swapping –** In multiprogramming, the instructions that have used the time slice are swapped out from the memory.
- **Techniques** :

  (a) Single Partition Allocation Schemes – The memory is divided into two parts. One part is kept to be used by the OS and the other is kept to be used by the users.

  (b) Multiple Partition Schemes –

  1. **Fixed Partition –** The memory is divided into fixed size partitions.
  2. **Variable Partition –** The memory is divided into variable sized partitions.

  Note : Variable partition allocation schemes:

  1. **First Fit –** The arriving process is allotted the first hole of memory in which it fits completely.
  2. **Best Fit –** The arriving process is allotted the hole of memory in which it fits the best by leaving the minimum memory empty.
  3. **Worst Fit –** The arriving process is allotted the hole of memory in which it leaves the maximum gap.

Note:

- Best fit does not necessarily give the best results for memory allocation.
- The cause of external fragmentation is the condition in Fixed partitioning and Variable partitioning saying that the entire process should be allocated in a contiguous memory location.Therefore **Paging** is used.

1. **Paging –** The physical memory is divided into equal sized frames. The main memory is divided into fixed size pages. The size of a physical memory frame is equal to the size of a virtual memory frame.
2. **Segmentation –** Segmentation is implemented to give users a view of memory. The logical address space is a collection of segments. Segmentation can be implemented with or without the use of paging.

- **Page Fault:**

A page fault is a type of interrupt, raised by the hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Page Replacement Algorithms (Important):

1. **First In First Out (FIFO) –**

   This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced, the page in the front of the queue is selected for removal.
   For example, consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots. Initially, all slots are empty, so when 1, 3, 0 come they are allocated to the empty slots —> 3 Page Faults. When 3 comes, it is already in memory so —> 0 Page Faults. Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —> 1 Page Fault. Finally, 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —> 1 Page Fault.

   Belady's anomaly:

Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string (3 2 1 0 3 2 4 3 2 1 0 4) and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

2. **Optimal Page replacement –**

   In this algorithm, pages are replaced which are not used for the longest duration of time in the future.

   Let us consider page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 and 4 page slots. Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults. 0 is already there so —> 0 Page fault. When 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—> 1 Page fault. 0 is already there so —> 0 Page fault. 4 will takes place of 1 —> 1 Page Fault. Now for the further page reference string —> 0 Page fault because they are already available in the memory.
   Optimal page replacement is perfect, but not possible in practice as an operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

3. **Least Recently Used (LRU) –**

   In this algorithm, the page will be replaced with the one which is least recently used. Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially, we had 4-page slots empty. Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults. 0 is already there so —> 0 Page fault. When 3 comes it will take the place of 7 because it is least recently used —> 1 Page fault. 0 is already in memory so —> 0 Page fault. 4 will take place of 1 —> 1 Page Fault. Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

- **Disk Scheduling**: Disk scheduling is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.
    1. **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written.
    2. **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.
    3. **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
    4. **Disk Access Time:** Seek Time + Rotational Latency + Transfer Time
    5. **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of all requests.

- **Disk Scheduling Algorithms** **(Important)**:
    1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.
    2. **SSTF:** In SSTF (Shortest Seek Time First), requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in a queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first.
    3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence is also known as **elevator algorithm.**
    4. **CSCAN:** In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.
6. **CLOOK:** As LOOK is similar to SCAN algorithm, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Key Terms

- **Real-time system** is used in the case when rigid-time requirements have been placed on the operation of a processor. It contains well defined and fixed time constraints.
- **A monolithic kernel** is a kernel which includes all operating system code in a single executable image.
- **Micro kernel:** Microkernel is the kernel which runs minimal performance affecting services for the operating system. In the microkernel operating system all other operations are performed by the processor.

  Macro Kernel: Macro Kernel is a combination of micro and monolithic kernel.

- **Re-entrancy :** It is a very useful memory saving technique that is used for multi-programmed time sharing systems. It provides functionality that multiple users can share a single copy of a program during the same period. It has two key aspects:The program code cannot modify itself and the local data for each user process must be stored separately.
- **Demand paging** specifies that if an area of memory is not currently being used, it is swapped to disk to make room for an application's need.
- **Virtual memory (Imp)** is a very useful memory management technique which enables processes to execute outside of memory. This technique is especially used when an executing program cannot fit in the physical memory.
- **RAID** stands for Redundant Array of Independent Disks. It is used to store the same data redundantly to improve the overall performance. There are 7 RAID levels.
- **Logical address space** specifies the address that is generated by the CPU. On the other hand, **physical address space** specifies the address that is seen by the memory unit.

- **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.
    2. **External fragmentation**: It occurs when we deal with systems that have variable-size allocation units.
- **Spooling** is a process in which data is temporarily gathered to be used and executed by a device, program or the system. It is associated with printing. When different applications send output to the printer at the same time, spooling keeps these all jobs into a disk file and queues them accordingly to the printer.
- **Starvation** is Resource management problem. In this problem, a waiting process does not get the resources it needs for a long time because the resources are being allocated to other processes.
- **Aging** is a technique used to avoid starvation in the resource scheduling system.
- **Advantages of multithreaded programming**:
    1. Enhance the responsiveness to the users.
    2. Resource sharing within the process.
    3. Economical
    4. Completely utilize the multiprocessing architecture.
- **Thrashing** is a phenomenon in virtual memory schemes when the processor spends most of its time in swapping pages, rather than executing instructions.