OPERATING SYSTEM

Definition: An operating system (OS) is system software that manages computer hardware, and software resources, and provides common services for computer programs

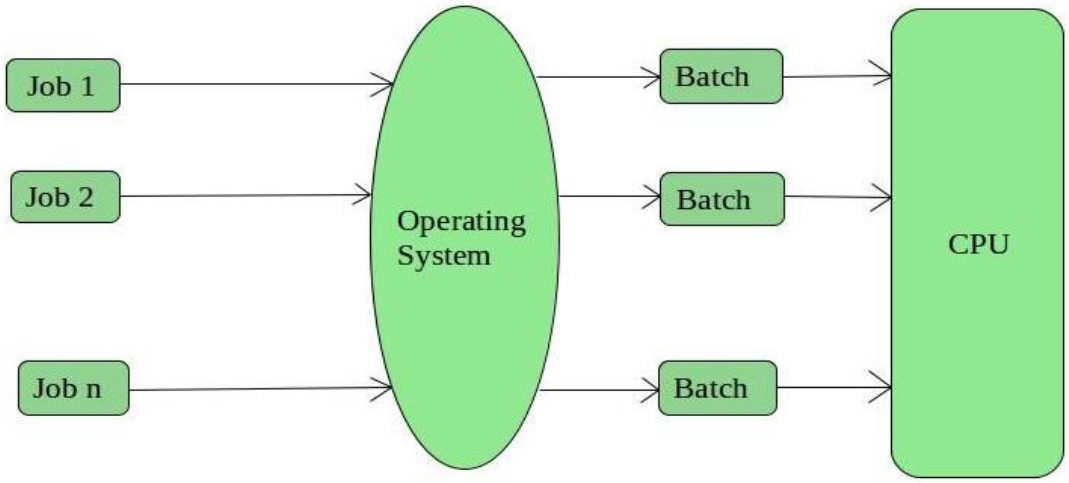
Types of Operating Systems

An Operating System performs all the basic tasks like managing files, processes, and memory. Thus operating system acts as the manager of all the resources, i.e. resource manager. Thus the operating system becomes an interface between the user and the machine.

Types of Operating Systems: Some of the widely used operating systems are as follows-

a. Batch Operating System:

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and groups them into batches. It is the responsibility of the operator to sort the jobs with similar needs.

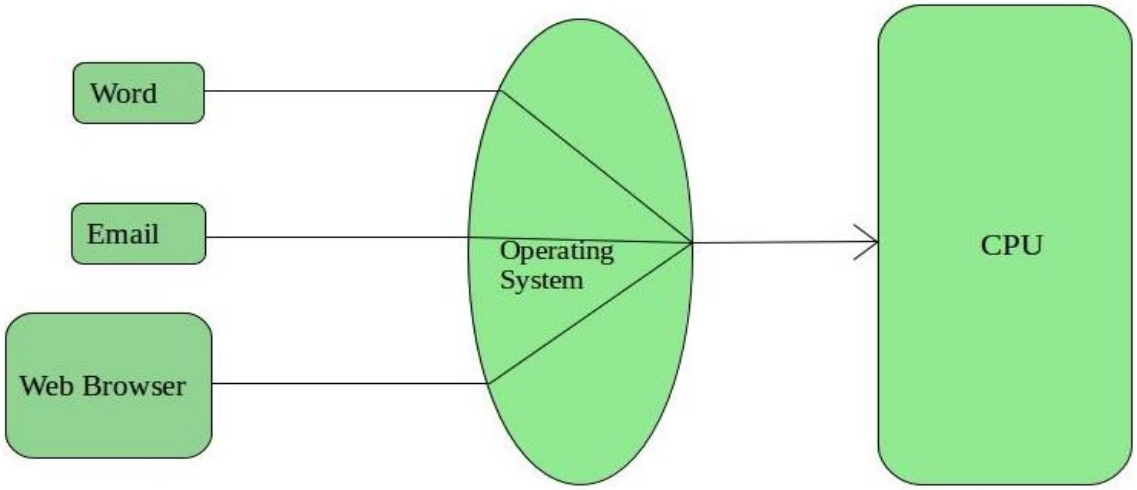


TANJO PARKSON

Examples of Batch based Operating Systems: Payroll System, Bank Statements etc.

b. Time-Sharing Operating Systems

Each task is given some time to execute so that all the tasks work smoothly. Each user gets a time of CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or from different users. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.

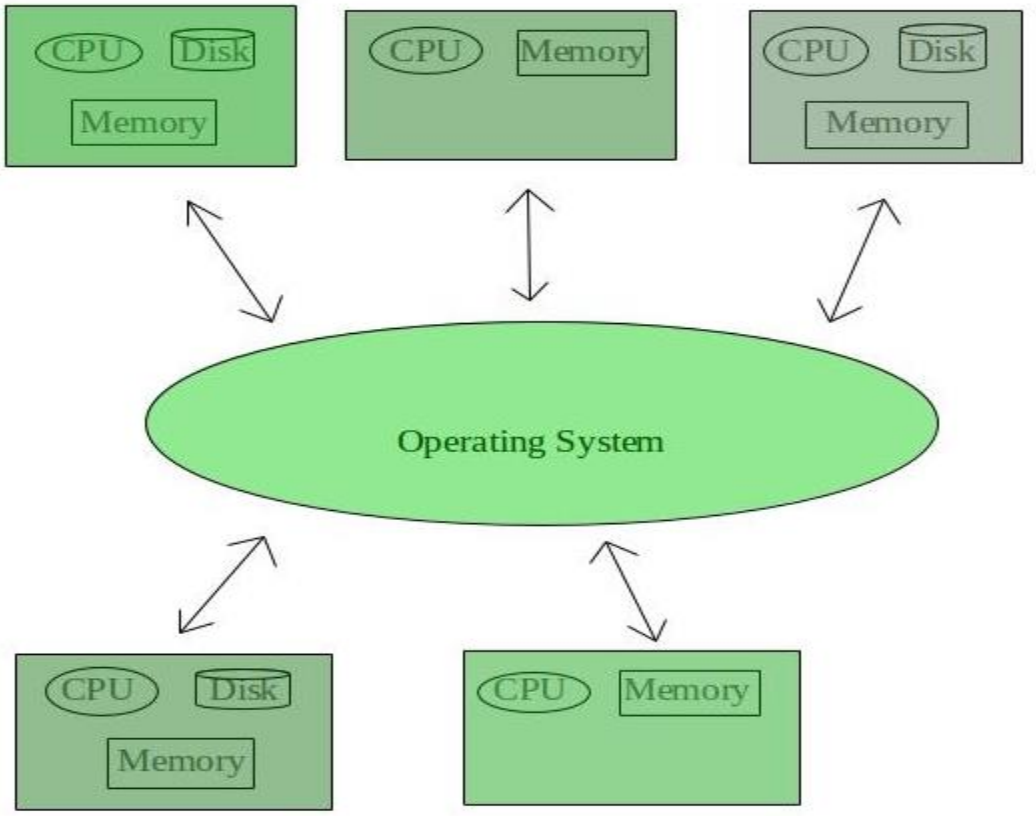


Examples of Time-Sharing OSs are: Multics, Unix, etc.

c. Distributed Operating System

Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred as loosely coupled systems or distributed systems. These systems' processors differ in size and function. The major benefit of working with these types of an operating systems is that it is always possible that one user can access the files or software which are not actually present on his system but on

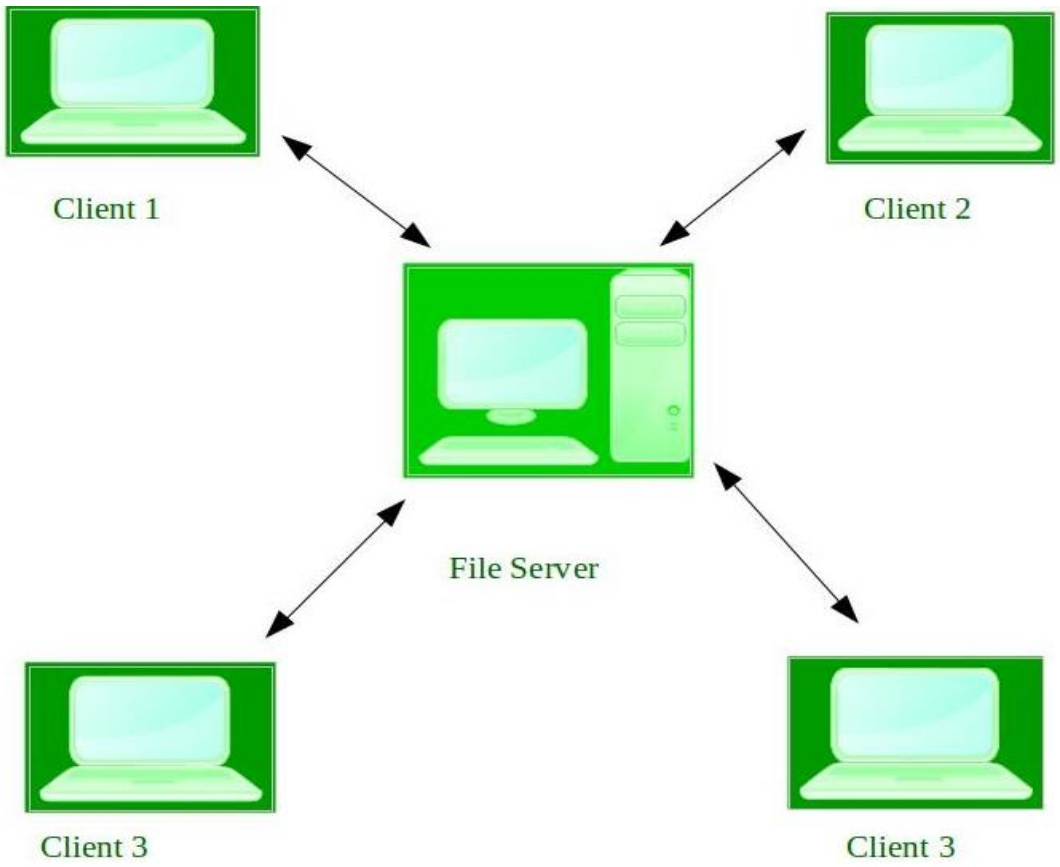
some other system connected within this network i.e., remote access is enabled within the devices connected in that network.



Examples of Distributed Operating Systems are- LOCUS etc.

d. Network Operating System

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access to files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as tightly coupled systems.



Examples of Network Operating Systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD

e. Real-Time Operating System

These types of OS serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called response time.

Real-time systems are used when there are time requirements are very strict like missile systems, air traffic control systems, robots, etc.

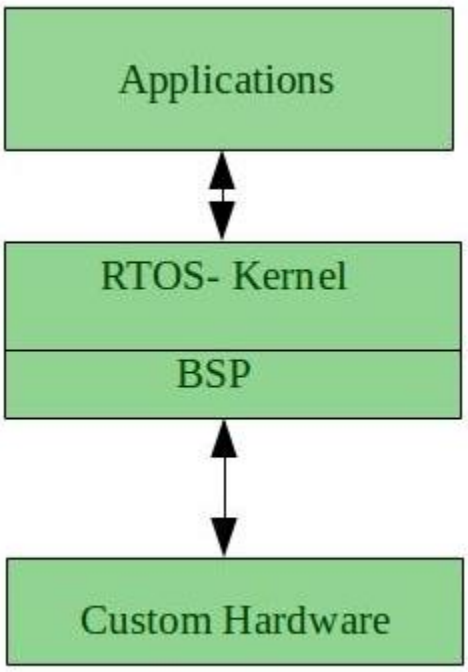
Two types of Real-Time Operating System, which are as follows:

- **Hard Real-Time Systems:**

These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags, which are required to be readily available in case of an accident. Virtual memory is almost never found in these systems.

- **Soft Real-Time Systems:**

These OSs are for applications where for time-constraint is less strict.



Properties of Operating system

a. **Batch Processing:** Batch processing is a technique in which an Operating System collects the programs and data together in a batch before processing starts.
b. **Multitasking**: Multitasking is when multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running. Each user has at least one separate program in memory.
c. **Multiprogramming**: Sharing the processor, when two or more programs reside in memory at the same time, is referred to as multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.
d. **Spooling**: Spooling is an acronym for simultaneous peripheral operations on-line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.
e. **Distributed Environment:** A distributed environment refers to multiple independent CPUs or processors in a computer system.

TANO PARKSON

THE FUNCTION OF THE OS

- **Memory Management** − Keeps track of the primary memory, i.e. what part of it is in use by whom, what part is not in use, etc., and allocates the memory when a process or program requests it.
- **Processor Management** − Allocates the processor (CPU) to a process and deallocates the processor when it is no longer required.
- **Device Management** − Keeps track of all the devices. This is also called I/O controller that decides which process gets the device, when, and for how much time.
- **File Management** − Allocates and de-allocates the resources and decides who gets the resources.
- **Security** − Prevents unauthorized access to programs and data by means of passwords and other similar techniques.
- **Job Accounting** − Keeps track of time and resources used by various jobs and/or users.
- **Control Over System Performance** − Records delays between the request for a service and from the system.

User Interfaces

The user interface is the manner in which the user interacts with the computer software and hardware.

There are two basic types of UI:

i. **Graphical User Interface (GUI):** Graphical User Interface provides the user graphical means to interact with the system. GUI can be a combination of both hardware and software. Using GUI, the user interprets the software.

   Every graphical component provides a way to work with the system. A GUI system has the following elements such as:

TANO PARKSON

- **Window** - An area where the contents of the application are displayed
- **Icon** - An icon is a small picture representing an associated application.
- **Cursor** - Interacting devices such as a mouse, touchpad, and digital pen are represented in GUI as cursors
- **Other elements include Tabs and Menu**

ii. Command Line Interface (CLI):

CLI provides the user with terminals where they type in commands to control and manage the system. CLI is the first choice of many technical users and programmers.

CLI has the following elements:

- **Command Prompt**: It is a text-based notifier that mostly shows the context in which the user is working. It is generated by the software system.
- **Cursor**: The cursor is mostly found in a blinking state. It moves as the user writes or deletes something.
- **Command**: A command is a text-based reference to a set of instructions, which are expected to be executed by the system.

The advantages of a command line interface are:

- granular control of an OS or application
- faster management of a large number of operating systems
- ability to store scripts to automate regular tasks
- Basic command line interface knowledge to help with troubleshooting, such as network connection issues.

The disadvantages of a command line interface are:

- GUI is more user-friendly
- steeper learning curve associated with memorizing commands and complex syntax/arguments
- Different commands used in different shells.

PROCESS MANAGEMENT

Introduction

A Program does nothing unless a CPU executes its instructions. A program in execution is called a process. In order to accomplish its task, the process needs computer resources.

The operating system is responsible for the following activities in connection with Process Management

- Scheduling processes and threads on the CPUs.
- Creating and deleting both user and system processes.
- Suspending and resuming processes.
- Providing mechanisms for process synchronization.
- Providing mechanisms for process communication

Attributes of a process

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them.

1. Process ID

When a process is created, a unique id is assigned to the process, which is used for the unique identification of the process in the system.

2. Program counter

A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed.

3. Process State

The Process, from its creation to its completion, goes through various states, which are new, ready, running, and waiting. We will discuss about it later in detail.

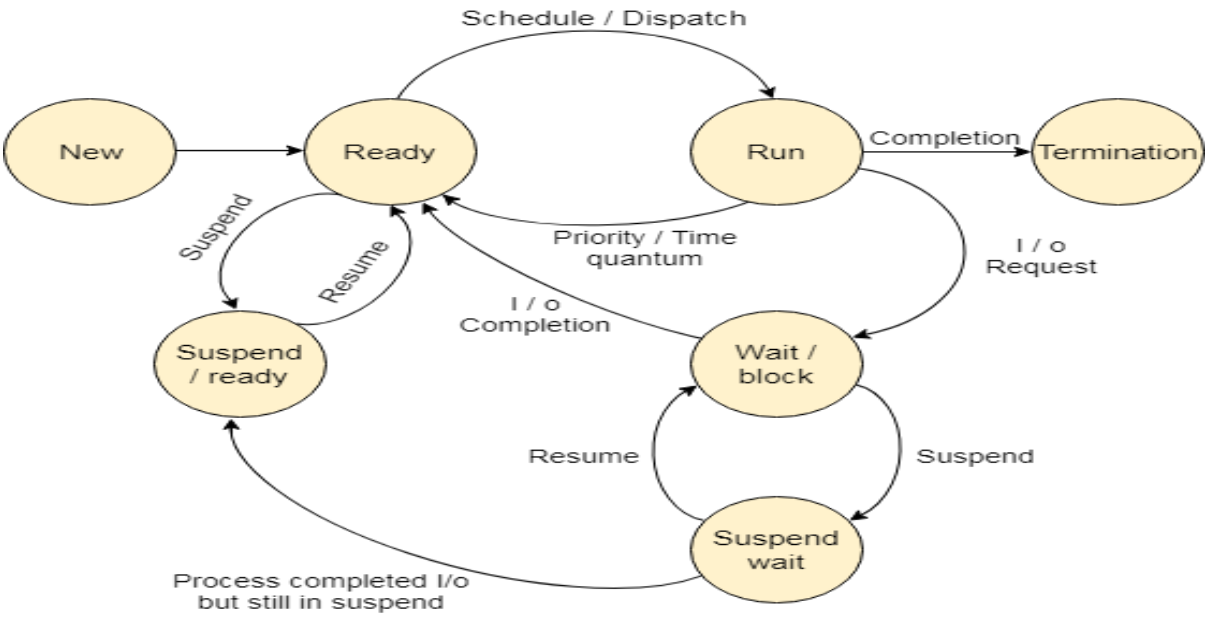
4. Priority

Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

Process State

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:

- New: In this state, the process is being created.
- Running: In this state, instructions are being executed.
- Waiting: In this state, the process is waiting for a different event to occur like I/O completion or treatment of a signal.
- Ready: In this state, the process waits to assign a processor.
- Terminated: In this state, the process has finished executing.
- Suspend ready: A process in the ready state, which is moved to secondary memory from the main memory due to lack of resources is called in the suspend ready state.
- Suspend Wait: Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory.



Operations on the Process

1. Creation

Once the process is created, it will be ready and come into the ready queue (main memory) and will be ready for execution.

2. Scheduling

Selecting the process which is to be executed next, is known as scheduling.

3. Execution

Once the process is scheduled for execution, the processor starts executing it.

4. Deletion/killing

Once the purpose of the process gets over then the OS will kill the process.

SCHEDULING ALGORITHM

DEFINITIONS:

Process Scheduling: This is the act of determining which process is in the **ready** state, and should be moved to the **running** state

The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs

CPU scheduling: This process allows one process to use the CPU while the execution of another process is on hold (in a waiting state) due to the unavailability of any resource like I/O etc, thereby making full use of the CPU.

The aim of CPU scheduling is to make the system efficient, fast, and fair.

Categories of Scheduler

- **Long-Term Scheduler:** The long-term scheduler selects the job from the job pool and loads it into the ready line. It controls the degree of multiprogramming. It is also known as JOB Scheduler because it manages the jobs.
- **Short-Term Scheduler:** Short-term Scheduler select the process from the line and Assign CPU for performance. It is also known as a CPU Scheduler because it manages CPU utilization.
- **Mid-Term Scheduler:** Mid-term scheduler removes those processes from the main memory and places them in secondary memories, which are not active for some time. Primarily Scheduler is used for swap out or swap in.

Categories of Scheduling Algorithm

- **Preemptive Scheduling:** If the CPU has been allocated to a process, the process will switch from a running state to a ready state after entering into a new process.

- **Non-Preemption:** A running process cannot be interrupted or released CPU after entering a new process into the ready queue.

Some Key Definitions

- **CPU Utilization:** CPU utilization refers to the use of the CPU for maximum time. So that it will not remain, free.
- **Throughput:** Throughput is the measure of completing the number of processes per unit of time.
- **Turnaround Time:** Time interval from submission of a process to completion of a process. Turnaround time is the sum of periods spent waiting to get into time, waiting time in the ready queue, I/O processes, or execution time.

  Turnaround time = Completion Time - Arrival Time
- **Waiting Time:** Waiting time is the sum of time when a process waits for the CPU in the ready queue. It may be in the continue slot or maybe a non-continue slot.

  Waiting time for Process = Completion time - Arrival time – Execution time
- **Response Time:** Time taken by the CPU to respond to a process.
- **CPU Burst:** The time required by the Process for execution.

Different Scheduling Algorithm

I. **First Come First Serve:** In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first. A perfect real-life example of FCFS scheduling is **buying tickets at the ticket counter.** The FCFS is non-preemptive.

   a. **Calculating Average waiting tine**

For every scheduling algorithm, **Average waiting time** is a crucial parameter to judge its performance.

AWT or Average waiting time is the average of the waiting times of the processes in the queue, waiting for the scheduler to pick them for execution.

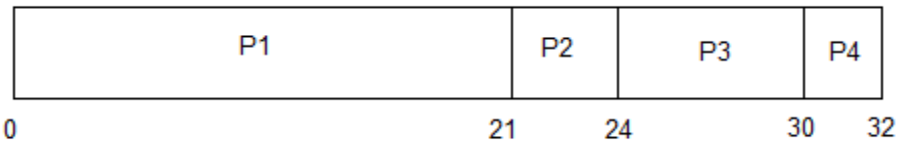
Lower the Average Waiting Time, the better the scheduling algorithm.

Example:

Considering the processes P1, P2, P3, and P4 given in the below table, arrive for execution in the same order, with **Arrival Time** 0, and given **Burst Time**, let's find the average waiting time using the FCFS scheduling algorithm.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The average waiting time will be = (0 + 21 + 24 + 30)/4 = 18.75 ms



This is the GANTT chart for the above processes

Advantages of FCFS scheduling Algorithm

- FCFS algorithm doesn't include any complex logic, it just puts the process requests in a queue and executes it one by one.
- Hence, FCFS is pretty simple and easy to implement.

- Eventually, every process will get a chance to run, so starvation does not occur.

Disadvantages of FCFS Scheduling Algorithm

- There is no option for pre-emption of a process. If a process is started, then the CPU executes the process until it ends.
- Because there is no pre-emption, if a process executes for a long time, the processes in the back of the queue will have to wait for a long time before they get a chance to be executed.
- Resources utilization in parallel is not possible, which leads to **Convoy Effect**, and hence poor resource (CPU, I/O, etc.) utilization

What is Convoy Effect?

Convoy Effect is a situation where many processes that need to use a resource for short time are blocked by one process holding that resource for a long time. This essentially leads to poor utilization of resources and hence poor performance

II. Shortest Job First (SJF)

The process, which has the shortest burst time, is scheduled first. If two processes have the same bust time then FCFS is used to break the tie. It is a non-preemptive scheduling algorithm.

Consider the below processes available in the ready queue for execution, with **arrival time** as 0 for all and given **burst times**.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

In Shortest Job First Scheduling, the shortest Process is executed first.

Hence the GANTT chart will be following :

Now, the average waiting time will be = (0 + 2 + 5 + 11)/4 = 4.5 ms

As you can see above, the process **P4** will be picked up first as it has the shortest burst time, then **P2**, followed by **P3**, and at last **P1**.

We scheduled the same set of processes using the FCFS algorithm in the previous example and got the average waiting time to be 18.75ms, whereas, with SJF, the average waiting time comes out to 4.5ms.

Advantages of SJF scheduling Algorithm

- According to the definition, short processes are executed first and then followed by longer processes.
- The throughput is increased because more processes can be executed in less amount of time.

Disadvantages:

- The CPU beforehand must know the time taken by a process, which is not possible.
- Longer processes will have more waiting time, eventually they'll suffer starvation

III. Priority Scheduling Algorithm

Processes are executed according to their priorities, i.e., the highest priority process is executed first. If the priorities of the two processes match, then schedule according to arrival time. Here starvation of process is possible.

Example: Consider the below table fo processes with their respective CPU burst times and the priorities

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,

The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

The disadvantage of Priority Scheduling Algorithm

In the priority scheduling algorithm, the chances of **indefinite blocking** or **starvation**.

A process is considered blocked when it is ready to run but has to wait for the CPU as some other process is running currently.

However, in the case of priority scheduling if new higher-priority processes keep coming in the ready queue then the processes waiting in the ready queue with lower priority may have to wait for long durations before getting the CPU for execution.

In 1973, when the IBM 7904 machine was shut down at MIT, a low-priority process was found which was submitted in 1967 and had not yet been run.

Aging Technique:

To prevent starvation of any process, we can use the concept of **aging** where we keep on increasing the priority of low-priority processes based on their waiting time.

For example, if we decide the aging factor to be **0.5** for each day of waiting, then if a process with priority **20**(which is comparatively low priority) comes in the ready queue. After one day of waiting, its priority is increased to **19.5**, and so on.

By doing so, we can ensure that no process will have to wait for an indefinite time for getting CPU time for processing

IV. Round Robin Scheduling Algorithm

- Fixed time is allotted to each process, called a **quantum**, for execution.
- Once a process is executed for the given time period that process is preempted and another process executes for the given time period.
- Context switching is used to save states of preempted processes.

Advantages:

- The CPU for a fixed time quantum serves each process, so all processes are given the same priority.

TANO PARKSON

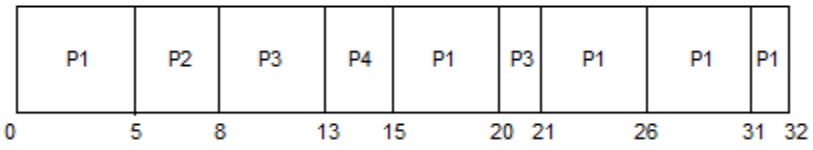
- Starvation does not occur because, for each round-robin cycle, every process is given a fixed time to execute. No process is left behind.

Disadvantages:

- The throughput in RR largely depends on the choice of the length of the time quantum. If the time quantum is longer than needed, it tends to exhibit the same behavior as FCFS.
- If the time quantum is shorter than needed, the number of times that CPU switches from one process to another process, increases. This leads to decrease in CPU efficiency.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The GANTT chart for round robin scheduling will be,



The average waiting time will be, 11 ms.

Problems encounter in Process Management

DEADLOCK

A process in operating systems uses different resources and uses resources in the following way.

- Requests a resource
- Use the resource
- Releases the resource

A deadlock is 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.

Characteristics of Deadlock

Four conditions must be present simultaneously for the deadlock to occur

- ***Mutual Exclusion:*** One or more than one resource is non-sharable (Only one process can use at a time)
- ***Hold and Wait:*** A process is holding at least one resource and waiting for resources.
- ***No Preemption:*** A resource cannot be taken from a process unless the process releases the resource.
- ***Circular Wait:*** A set of processes are waiting for each other in circular form.

How to avoid Deadlocks

Deadlocks can be avoided by avoiding at least one of the four conditions because all these four conditions are required simultaneously to cause deadlock.

1. **Mutual Exclusion**

Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, require exclusive access by a single process.

2. **Hold and Wait**

   In this condition, processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

3. **No Preemption**

   Preemption of process resource allocations can avoid the condition of deadlocks, wherever possible.

4. **Circular Wait**

   The circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing (or decreasing) order.

Handling Deadlock

The above points focus on preventing deadlocks. But what to do once a deadlock has occurred? The following three strategies can be used to remove deadlock after its occurrence.

1. **Preemption**

   We can take a resource from one process and give it to another. This will resolve the deadlock situation, but sometimes it does cause problems.

2. **Rollback**

   In situations where deadlock is a real possibility, the system can periodically make a record of the state of each process and when a deadlock occurs, roll everything back to the last checkpoint, and restart, but allocating resources differently so that deadlock does not occur.

3. **Kill one or more processes**

   This is the simplest way, but it works.

TANO PARKSON

Critical Section

The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

The solution to the Critical Section Problem

The critical section problem needs a solution to synchronize the different processes. The solution to the critical section problem must satisfy the following conditions −

a. Mutual Exclusion

Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

b. Progress

Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.

c. Bounded Waiting

Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

Semaphore

A semaphore is a signaling mechanism and another thread can signal a thread that is waiting on a semaphore. This is different from a mutex, as the mutex can be signaled only by the thread that called the wait function.

A semaphore uses two atomic operations, wait and signal for process synchronization.

TANO PARKSON

Race Condition

When two or more concurrently running threads/processes access a shared data item or resources and the results depend on the order of execution, we have a race condition.

Avoiding Race Conditions:

To avoid race condition, we need Mutual Exclusion. Mutual Exclusion is some way of making sure that if one process is using a shared variable or file, the other processes will be excluded from doing the same things.0

TANO PARKSON

MEMORY MANAGEMENT

1. Memory Management and Memory Management Unit (MMU)

Definitions:

- **Memory management** is the functionality of an operating system, which manages primary memory and moves processes back and forth between main memory and disk during execution.
- **A memory management unit (MMU)** is a computer hardware component that handles all memory and caching operations associated with the processor. Memory-Management Unit is used for mapping logical addresses to their corresponding physical address

The work of the MMU can be divided into three major categories:

- Hardware memory management, which oversees and regulates the processor's use of RAM (random accewwacftr5rss memory) and cache memory.
- OS (operating system) memory management, which ensures the availability of adequate memory resources for the objects and data structure of each running program at all times.
- Application memory management, which allocates each individual program's required memory, and then recycles freed-up memory space when the operation concludes.

Importance of Memory Management

- It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.
- Tracks whenever inventory gets freed or unallocated. According to it will update the status.

- It allocates the space to application routines.
- It applications do not interfere with each other.
- It helps protect different processes from each other
- It places the programs in memory so that memory is utilized to its full extent.

2. **Memory address**

- **Logical Address** is generated by the CPU while a program is running. The logical address is a virtual address as it does not exist physically, therefore, it is also known as Virtual Address. CPU uses this address as a reference to access the physical memory location. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective.
- **Physical Address** identifies a physical location of required data in memory. The user never directly deals with the physical address but can access it by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.

3. **Memory management techniques**

Memory management function of operating system helps in allocating the main memory space to the processes and their data at the time of their execution.

The following are there key memory management techniques used by an operating system:

- **Contiguous memory allocation**

In **contiguous memory allocation**, all the available memory space remains together in one place. It means freely available memory partitions are not scattered here and there across the whole memory space.

- **Partitioned allocation:**

Memory is divided into different blocks or partitions. Each process is allocated according to the requirement.

- **Segmentation**

Segmentation refers to the technique of dividing the physical memory space into multiple blocks. Each block has a specific length and is known as a segment.

- **Paging**

Paging is a technique in which the main memory of computer system is organized in the form of equal-sized blocks called pages. In this technique, the address of occupied pages of physical memory are stored in a table, which is known as a page table.

- **Swapping**

Swapping is the technique used by an operating system for efficient management of the memory space of a computer system. Swapping involves performing two tasks called swapping in and swapping out. The task of placing the pages or blocks of data from the hard disk to the main memory is called swapping in. On the other hand, the task of removing pages or blocks of data from main memory to the hard disk is called swapping out. The swapping technique is useful when larger program is to be executed or some operations have to performed on a large file.

TANO PARKSON

4. Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. After a period, processes cannot be allocated to memory blocks considering their small size and memory blocks remain unused. This problem is known as Fragmentation.

- **Type of Fragmentation**

There are two types of fragmentation

i. **Internal Fragmentation**:

The memory block assigned to the process is bigger. Some portion of memory is left unused, as another process cannot use it.

Internal fragmentation occurs when the memory is divided into fixed-sized blocks. Whenever a process requests memory, the fixed-sized block is allocated to the process. In case the memory assigned to the process is somewhat larger than the memory requested, then the difference between assigned and requested memory is internal fragmentation.

Internal fragmentation can be solved by partitioning the memory into variable-sized blocks and assigning the best fit block to the requesting process

ii. **External Fragmentation:**

Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.

External fragmentation occurs when there is a sufficient amount of space in the memory to satisfy the memory request of a process. However, the process's memory request cannot be satisfied as the memory available is in a non-contiguous manner.

TANO PARKSON

Whether you apply a first-fit or best-fit memory allocation strategy it will cause external fragmentation.

External fragmentation can be solved by compaction, but it is expensive to implement, so the processes must be allowed to acquire physical memory in a non-contiguous manner, to achieve this the technique of paging and segmentation is introduced.

5. **Memory Allocation**

Memory allocation is a process by which computer programs are assigned memory or space. It is of three types:

- **Best Fit:**

The smallest block that is big enough is allocated to the program.

For example, suppose a process requests 12KB of memory and the memory manager currently has a list of unallocated blocks of 6KB, 14KB, 19KB, 11KB, and 13KB blocks. The best-fit strategy will allocate 12KB of the 13KB block to the process.

- **First Fit:**

The first block that is big enough is allocated to the program.

Using the same example as above, the first fit will allocate 12KB of the 14KB block to the process.

- **Worst Fit:**

The largest block that is big, enough is allocated to program.

Using the same example as above, worst fit will allocate 12KB of the 19KB block to the process, leaving a 7KB block for future use.

TANO PARKSON

6. Virtual Memory

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to function as the computer's RAM.

- **Benefits of having Virtual Memory**
    1. Large programs can be written, as the virtual space available is huge compared to physical memory.
    2. Less I/O required, leads to faster and easy swapping of processes.
    3. More physical memory is available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

TANO PARKSON