

OPERATING SYSTEM

ULEMU MPONELA

DMI ST JOHN THE BAPTIST UNIVERSITY : UNIT-I

Table of Contents

Introduction to operating system	2
• Concurrent Systems.....	9
• Resource Sharing	10
• Changes in Hardware and Software.....	10
• Portable Operating Systems.....	10
• Backward Compatibility.....	10
• Program Execution.....	10
Operating System Activities	10
Program.....	12
Process Life Cycle	12
Process Control Block (PCB).....	13
Process Scheduling Queues	15
Two-State Process Model.....	15
Schedulers.....	16
Long Term Scheduler	16
Short Term Scheduler	16
Medium Term Scheduler.....	17
Comparison among Scheduler	17
Process Creation	18
Process Preemption.....	18
Process Blocking	18
Process Termination.....	18
Reasons for needing cooperating processes	18
Methods of Cooperation	19
Synchronization in Inter-process Communication	19
Approaches to Inter-process Communication	20
Types of Threads.....	21
• User-level thread	21
• Kernel level thread.....	22
Benefits of Threads	22

Unit I Introduction:

Views - Goals - OS Structure - Components - Services – system calls - System Structure - Virtual Machines - System Design and Implementation. Process Management: Introduction - Process - Process Scheduling – Operations on processes - Cooperating Process - Inter-process Communication. – Threads.

INTRODUCTION TO OPERATING SYSTEM

An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a computer system.

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Main Goals of Operating Systems

Below are some of the goals of the operating system

- To make the computer system convenient to use
- To facilitate the use of computer hardware in efficient way
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.

Main Components of Operating System

Following are some of important components of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

SERVICES PROVIDED BY OPERATING SYSTEM

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system –

- Program execution
- I/O operations
- File System manipulation

- Communication
- Error Detection
- Resource Allocation
- Protection

1. Program Execution

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management –

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

2. I/O Operation

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

3. File System Manipulation

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management –

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.

- Operating System provides an interface to create the backup of file system.

4. Communication

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication –

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

5. Error Handling

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling –

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

6. Resource Management

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management –

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

7. Protection

Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection –

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

SYSTEM CALLS

A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System call provides the services of the operating system to the user programs via Application Program Interface(API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Services Provided by System Calls

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking

Types of System Calls

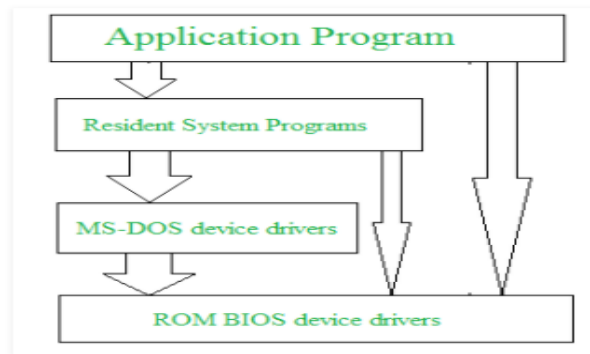
1. Process control: end, abort, create, terminate, allocate and free memory.
2. File management: create, open, close, delete, read file etc.
3. Device management
4. Information maintenance
5. Communication

SYSTEM STRUCTURE IN OPERATING SYSTEM

Operating system can be implemented with the help of various structures. The structure of the OS depends mainly on how the various common components of the operating system are interconnected and melded into the kernel. Depending on this we have following structures of the operating system:

Simple Structure

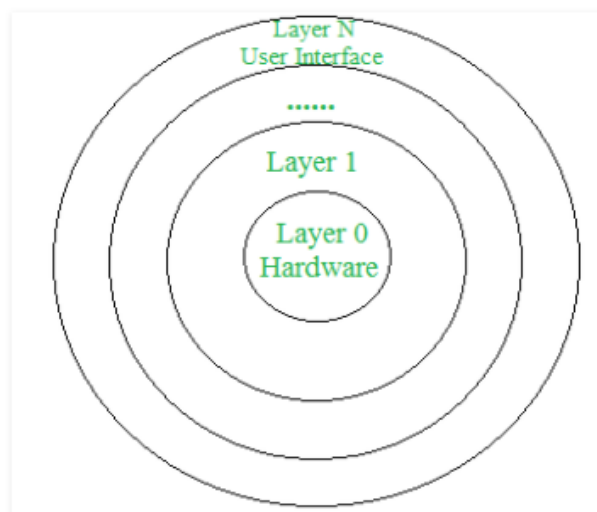
Such operating systems do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. MS-DOS is an example of such operating system. In MS-DOS application programs are able to access the basic I/O routines. These types of operating system cause the entire system to crash if one of the user programs fails. Diagram of the structure of MS-DOS is shown below.



Layered Structure

An OS can be broken into pieces and retain much more control on system. In this structure the OS is broken into number of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower level layers only. This simplifies the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Careful planning of the layers is necessary as a layer can use only lower level layers. UNIX is an example of this structure.



Micro-Kernel Structure

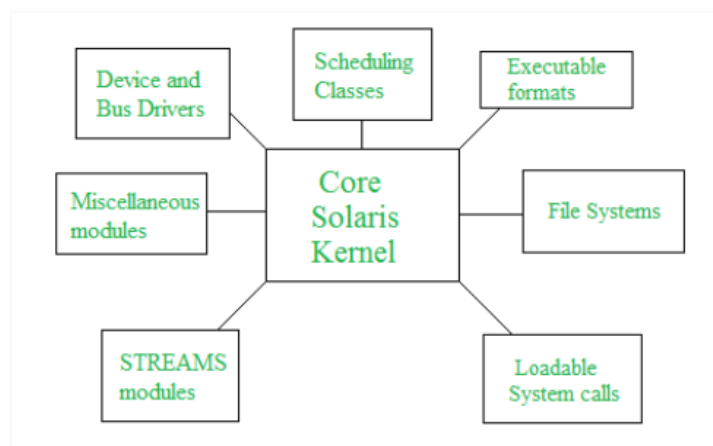
This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This result in a smaller kernel called the micro-kernel.

Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails then rest of the operating system remains untouched. Mac OS is an example of this type of OS.

Modular Structure

It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces but it is more flexible than the layered structure as a module can call any other module.

For example Solaris OS is organized as shown in the figure



VIRTUAL MACHINES

Virtual Machine abstracts the hardware of our personal computer such as CPU, disk drives, memory, NIC (Network Interface Card) into many different execution environments as per our requirements, hence giving us a feel that each execution environment is a single computer.

When we run different processes on an operating system, it creates an illusion that each process is running on a different processor having its own virtual memory, with the help of CPU scheduling and virtual-memory techniques. There are additional features of a process that cannot be provided by the hardware alone like system calls and a file system. The virtual machine approach does not provide these additional functionalities but it only provides an interface that is same as basic hardware. Each process is provided with a virtual copy of the underlying computer system.

We can create a virtual machine for several reasons, all of which are fundamentally related to the ability to share the same basic hardware yet can also support different execution environments, i.e., different operating systems simultaneously.

The main drawback with the virtual-machine approach involves disk systems. Let us suppose that the physical machine has only three disk drives but wants to support seven virtual machines. Obviously, it cannot allocate a disk drive to each virtual machine, because virtual-machine software itself will need substantial disk space to provide virtual memory and spooling. The solution is to provide virtual disks.

Users are thus given their own virtual machines. After which they can run any of the operating systems or software packages that are available on the underlying machine. The virtual-machine software is concerned with multi-programming multiple virtual machines onto a physical machine, but it does not need to consider any user-support software. This arrangement can provide a useful way to divide the problem of designing a multi-user interactive system, into two smaller pieces.

Advantages:

1. There are no protection problems because each virtual machine is completely isolated from all other virtual machines.
2. Virtual machine can provide an instruction set architecture that differs from real computers.
3. Easy maintenance, availability and convenient recovery.

Disadvantages:

1. When multiple virtual machines are simultaneously running on a host computer, one virtual machine can be affected by other running virtual machines, depending on the workload.
2. Virtual machines are not as efficient as a real one when accessing the hardware.

SYSTEM DESIGN AND IMPLEMENTATION

The operating system is needed to design and implement because without proper design and implementation any system cannot work properly, for every aspect or for any development a proper design and implementation should be necessary so that it can work in good manner and we can easily debug if any failures occur.

So, design and implementation is a necessary part of an operating system and this technique can be used by every user who uses a computer.

There are different types of techniques to design and implement the operating system.

- Design goals
- Mechanism
- Implementation

Let us discuss each technique in detail.

a) Design Goals

Let us understand the features of the design goals.

- **Concurrent Systems**

Operating systems must handle multiple devices as well as multiple users concurrently. It is a must for modern multiple core architectures. Due to these features the design of the operating system is complex and very difficult to make.

- **Security and Privacy**

Operating systems must provide security and privacy to a system. It is important to prevent the malicious user from accessing your system and to prevent the stealing of the user programs.

- **Resource Sharing**

Operating system ensures that the resources of the system must be shared in a correct fashion in between multiple user processes. It becomes more complex when multiple users use the same device.

- **Changes in Hardware and Software**

Operating systems must be flexible in order to accommodate any change to the hardware and software of the system, It should not be obsolete. It is necessary as it is costly to change the operating system again and again on any change to the software or hardware.

- **Portable Operating Systems**

Operating system that is able to work with different hardware and systems is called a portable operating system and it is a very important design goal.

- **Backward Compatibility**

Any upgrade to the current operating system could not hinder its compatibility with the machine i.e. if the previous version of the operating system is compatible with the system then the newer or upgraded version should also be compatible with the system this is called backward compatibility.

b) Mechanism

In an operating system there is a particular mechanism to be followed and this mechanism is responsible for the entire task performed by the operating system.

When a task is performed in the operating system then we follow a particular mechanism for input, store, process, and output and by using this process we can define memory to the different tasks performed by the computer.

An Operating system provides the services to users and programs. Like I/O operations, program execution, file system manipulation, Resource allocation, Protection.

- **Program Execution**

OS handles many activities from user programs to system programs like printer spooler, name servers, file servers etc. Each of these activities is encapsulated as a process. A process includes the complete execution context.

Operating System Activities

The operating system activities are as follows –

- Loads a program into memory

- Execute the program
- Handle's program's execution
- Provides a mechanism for process synchronization
- Provides a mechanism for process communication
- Provides a mechanism for deadlock handling

c) Implementation

Once the operating system is designed it must be implemented because it is a collection of many programs written by many people over a long period of time.

Implementation is a process which is most important for the operating system.

An operating system needs to be implemented because when it is implemented then new tasks can be performed and new application software can also install in your computer and run your computer smoothly for that we need implementation of operating system.

PROCESS MANAGEMENT

Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems. The OS must allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other methods and allows synchronization among processes.

It is the job of OS to manage all the running processes of the system. It handles operations by performing tasks like process scheduling and such as resource allocation.

Process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections — stack, heap, text and data. The following image shows a simplified layout of a process inside main memory:



S.N.	Component & Description
1	Stack The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	Heap This is dynamically allocated memory to a process during its run time.
3	Text This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	Data This section contains the global and static variables.

Program

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C programming language.

```
#include <stdio.h>

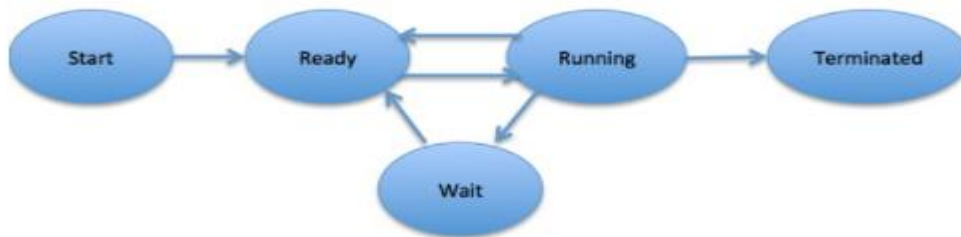
int main() {
    printf("Hello, World! \n");
    return 0;
}
```

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

In general, a process can have one of the following five states at a time.



S.N.	State & Description
1	Start This is the initial state when a process is first started/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. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.
3	Running Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
4	Waiting Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.
5	Terminated or Exit Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table

S.N.	Information & Description
1	Process State The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	Process privileges This is required to allow/disallow access to system resources.
3	Process ID Unique identification for each of the process in the operating system.
4	Pointer A pointer to parent process.
5	Program Counter Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	CPU registers Various CPU registers where process need to be stored for execution for running state.
7	CPU Scheduling Information Process priority and other scheduling information which is required to schedule the process.
8	Memory management information This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	Accounting information This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	IO status information This includes a list of I/O devices allocated to the process.

The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

PROCESS SCHEDULING

Process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

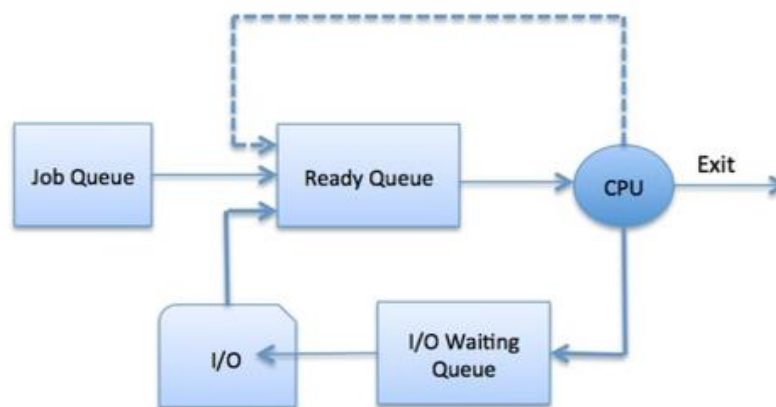
Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Process Scheduling Queues

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues –

- **Job queue** – This queue keeps all the processes in the system.
- **Ready queue** – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** – The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below:

S.N.	State & Description
1	Running When a new process is created, it enters into the system as in the running state.
2	Not Running Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types –

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

OPERATIONS ON PROCESSES

There are many operations that can be performed on processes. Some of these are process creation, process preemption, process blocking, and process termination. These are given in detail as follows:

Process Creation

Processes need to be created in the system for different operations. This can be done by the following events –

- User request for process creation
- System initialization
- Execution of a process creation system call by a running process
- Batch job initialization

A process may be created by another. The creating process is called the parent process and the created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent and child processes have the same memory image, open files, and environment strings. However, they have distinct address spaces.

Process Preemption

An interrupt mechanism is used in preemption that suspends the process executing currently and the next process to execute is determined by the short-term scheduler. Preemption makes sure that all processes get some CPU time for execution

Process Blocking

The process is blocked if it is waiting for some event to occur. This event may be I/O as the I/O events are executed in the main memory and don't require the processor. After the event is complete, the process again goes to the ready state.

Process Termination

After the process has completed the execution of its last instruction, it is terminated. The resources held by a process are released after it is terminated.

A child process can be terminated by its parent process if its task is no longer relevant. The child process sends its status information to the parent process before it terminates. Also, when a parent process is terminated, its child processes are terminated as well as the child processes cannot run if the parent processes are terminated.

COOPERATING PROCESSES

Cooperating processes are those that can affect or are affected by other processes running on the system. Cooperating processes may share data with each other.

Reasons for needing cooperating processes

There may be many reasons for the requirement of cooperating processes. Some of these are given as follows –

- **Modularity**

Modularity involves dividing complicated tasks into smaller subtasks. These subtasks can be completed by different cooperating processes. This leads to faster and more efficient completion of the required tasks.

- **Information Sharing**

Sharing of information between multiple processes can be accomplished using cooperating processes. This may include access to the same files. A mechanism is required so that the processes can access the files in parallel to each other.

- **Convenience**

There are many tasks that a user needs to do such as compiling, printing, editing etc. It is convenient if these tasks can be managed by cooperating processes.

- **Computation Speedup**

Subtasks of a single task can be performed parallel using cooperating processes. This increases the computation speedup as the task can be executed faster. However, this is only possible if the system has multiple processing elements.

Methods of Cooperation

Cooperating processes can coordinate with each other using shared data or messages. Details about these are given as follows –

- **Cooperation by Sharing**

The cooperating processes can cooperate with each other using shared data such as memory, variables, files, databases etc. Critical section is used to provide data integrity and writing is mutually exclusive to prevent inconsistent data.

- **Cooperation by Communication**

The cooperating processes can cooperate with each other using messages. This may lead to deadlock if each process is waiting for a message from the other to perform an operation. Starvation is also possible if a process never receives a message.

INTER-PROCESS COMMUNICATION

Inter-process communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

Synchronization in Inter-process Communication

Synchronization is a necessary part of inter-process communication. It is either provided by the inter-process control mechanism or handled by the communicating processes. Some of the methods to provide synchronization are as follows –

- **Semaphore**

A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.

- **Mutual Exclusion**

Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.

- **Barrier**

A barrier does not allow individual processes to proceed until all the processes reach it. Many parallel languages and collective routines impose barriers.

- **Spinlock**

This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

Approaches to Inter-process Communication

The different approaches to implement inter-process communication are given as follows:

- **Pipe**

A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.

- **Socket**

The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for inter-process communication.

- **File**

A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.

- **Signal**

Signals are useful in inter-process communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

- **Shared Memory**

Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

- **Message Queue**

Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for inter-process communication and are used by most operating systems.

THREADS

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.

Types of Threads

In the operating system there are two types of threads.

1. Kernel level thread.
2. User-level thread.

- **User-level thread**

The operating system does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes. examples: Java thread.

- **Kernel level thread**

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.

Benefits of Threads

- **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
- **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
- **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
- **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
- **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.