Contents

[PROCESS CONCEPT 2](#_Toc130499145)

[Process States 2](#_Toc130499146)

[Process Control Block (PCB) 3](#_Toc130499147)

[CPU SCHEDULING 4](#_Toc130499148)

[PROCESS SCHEDULERS 4](#_Toc130499149)

[1. Long Term Scheduler 4](#_Toc130499150)

[2. Short Term Scheduler 4](#_Toc130499151)

[3. Medium Term Scheduler 4](#_Toc130499152)

[PROCESS QUEUES 5](#_Toc130499153)

[1. Job Queue 5](#_Toc130499154)

[2. Ready Queue 5](#_Toc130499155)

[3. Waiting Queue: 5](#_Toc130499156)

[Context Switching 5](#_Toc130499157)

[PROCESS OPERATIONS 6](#_Toc130499158)

[1. Process Creation 6](#_Toc130499159)

[2. Process Termination 6](#_Toc130499160)

[INTERPROCESS COMMUNICATION 8](#_Toc130499161)

[1. Shared Memory 8](#_Toc130499162)

[Producer-Consumer Problem 9](#_Toc130499163)

[a. Bounded-Buffer 9](#_Toc130499164)

[b. Unbounded-Buffer 9](#_Toc130499165)

[2. Message Passing 10](#_Toc130499166)

[TYPES OF MESSAGE PASSING MODEL 10](#_Toc130499167)

[1. Direct or Indirect 10](#_Toc130499168)

[2. Synchronous or Asynchronous 11](#_Toc130499169)

[3. Buffering 11](#_Toc130499170)

# PROCESS CONCEPT

An operating system is responsible for running different types of programs.

In batch systems, these programs are called jobs.

In time-shared systems, they are referred to as user programs or tasks.

The terms job and process are often used interchangeably.

A process is essentially a program in execution.

A program is a passive entity that is stored on a disk in the form of an executable file. When the executable file of a program is loaded into memory and its execution is initiated through various ways such as GUI mouse clicks, command line entry of its name, etc., it becomes a process. A process is an active entity that comprises multiple parts such as:

1. Program Code/Text

This comprises the contents present in the processor’s registers as well as the current activity reflected by the value of the program counter.

1. Stack

Temporary data like method or function parameters, return address, and local variables are stored in the process stack.

1. Data

The global as well as static variables are included in this section.

1. Heap

This is the memory that is dynamically allocated to a process during its execution.

## Process States

Process state refers to the current activity of a process, as it changes from one state to another during execution. The main states are new, running, ready, waiting, and terminated.

1. New: When a new process is created, it is in the new state. In this state, the process is waiting to be admitted to the system.
2. Ready: Once the process is admitted to the system, it moves to the ready state. In this state, the process is waiting for the CPU to be assigned.
3. Running: When the process is assigned to the CPU, it moves to the running state. In this state, the process is actively executing instructions.
4. Waiting: Sometimes a process needs to wait for an event, such as user input or completion of I/O. In this case, the process moves to the waiting state.
5. Terminated: When a process completes its execution, it moves to the terminated state. In this state, the process is no longer running, and its resources are released.

## Process Control Block (PCB)

Process Control Block (PCB) is used to represent a process in the operating system. It contains information on process state, ID, program counter, CPU registers, scheduling info, memory management, accounting info, and input/output status. PCB helps to manage and execute multiple processes in an operating system.

1. Process state: It represents the current activity level of a process (such as running, ready, blocked, suspended, etc.)

2. ID: It is a unique identifier assigned to each process by the operating system.

3. Program counter: It is a pointer that indicates the address of the next instruction to be executed.

4. CPU registers: They are small storage locations of the CPU used to hold intermediate or temporary values during processing.

5. Scheduling info: It contains information about the priority of a process. Additional scheduling information are required for the process to be scheduled.

6. Memory management: It contains information about the process's memory usage, such as the size of the process, its memory allocation, and the location of the memory.

7. Accounting info: It contains process-specific accounting information, such as the amount of CPU time used, the amount of memory used, and the amount of I/O performed by the process.

8. Input/output status: It contains the list I/O devices allocated to the process and list of open files.

The PCB architecture is fully dependent on the operating system, and different operating systems may include different information.

# CPU SCHEDULING

* When there are several or more runnable processes, the operating system chooses which one to run first; this is known as process scheduling.
* A **scheduler** is a program that uses a scheduling algorithm to make choices. The following are characteristics of a good scheduling algorithm:

1. For users, response time should be kept to a bare minimum.

2. The total number of jobs processed every hour should be as high as possible, implying that a good scheduling system should provide the highest possible throughput.

3. The CPU should be used to its full potential.

4. Each process should be given an equal amount of CPU time.

## PROCESS SCHEDULERS

1. Long Term Scheduler: - Long term scheduler is also referred to as Job Scheduler. The task of the Long-term scheduler is to select the process from secondary storage and hold them in the ready queue, which is in the primary storage.
2. Short Term Scheduler: - The task performed by the short-term scheduler is to choose one of the jobs from the ready queue and send it for execution to the CPU. Short Term Scheduler means a CPU scheduler.
3. Medium Term Scheduler: - The Medium-term scheduler is responsible for having the process suspended and resumed. It Decides which processes should be swapped out of the main memory to the secondary memory (disk) and which processes should be brought back into the main memory

## PROCESS QUEUES

The operating system handles a number of queues for a process state.

There are various types of queue:

* Job Queue
* Ready Queue
* Waiting Queue

1. Job Queue: - All processes are placed in the job queue at the beginning. It is kept in secondary storage. The long-term scheduler selects a few of the jobs and places them in the primary storage.
2. Ready Queue: - It is the list of processes that are waiting to be executed. Once a process is loaded into the memory, it is placed in the ready queue until it is given access to the CPU. With the help of the short-term scheduler, we select the job for the execution from the ready queue, and then we will dispatch it to the CPU.
3. Waiting Queue: While running when the process requires doing some operations means any input/output operation to finish its execution, then operating system switches the process state from running to the waiting state. The context (Program Control Block) is stored in the waiting queue that will be used by the processor when the IO is done.

## Context Switching

Context switching is the process of saving the context of one process, so that it can be resumed later, and then loading and executing the context of another process. It is necessary for multitasking and multiprogramming operating systems to give the illusion of executing multiple processes simultaneously on a single CPU

# PROCESS OPERATIONS

Process operations refer to the actions or events that can occur in a process's life cycle

## Process Creation

Process operations are events or actions that can occur during a process's life cycle. One such operation is process creation, which involves creating a new process from an existing process using the fork() system call. The newly created process is called the child process, and the process that created it is called the parent process. The parent and child processes have the same memory image, open files, and environment strings, but they have distinct address spaces.

The parent process can create multiple child processes, forming a tree of processes. Each process is identified and managed by a unique process identifier (pid). Resource sharing options include parent and child processes sharing all resources, children sharing a subset of the parent's resources, or parent and child sharing no resources.

Execution options include parent and children executing concurrently or the parent waiting until children terminate. Address space options include the child having a duplicate of the parent's address space or having a program loaded into it. These process operations are essential for managing the creation and execution of processes in an operating system.

## 2. Process Termination

Process Termination refers to the process of ending the execution of a process. When a process completes its last instruction, it terminates, and the resources held by it are released.

Sometimes, the parent process can terminate a child process if it is no longer needed. In this case, the child process sends its status information to the parent process before termination.

A process can also terminate itself by requesting the operating system to delete it using the exit() system call. When a process terminates, its resources are deallocated by the operating system.

The parent process may terminate the execution of its child processes using the abort() system call for several reasons, such as the child process has exceeded its allocated resources, or the task assigned to it is no longer needed. If a process terminates, then all its children must also be terminated, which is known as cascading termination.

The termination process can be initiated by the operating system, and the parent process can wait for the termination of a child process using the wait() system call. If the parent process does not wait, the process becomes a zombie, and if the parent process terminates without invoking wait, the process becomes an orphan.

# INTERPROCESS COMMUNICATION

IPC stands for Interprocess Communication, which is a mechanism that allows processes in a computer system to communicate with each other and synchronize their actions. Processes may need to exchange data, signals, or even perform operations on behalf of each other.

Processes within a computer system can be either independent or cooperating. Independent processes don't affect or get affected by the execution of other processes. On the other hand, cooperating processes can impact or be impacted by the execution of other processes.

There are several advantages to process cooperation.

One benefit is information sharing where multiple processes can share the same information to perform certain tasks. This sharing can happen through inter-process communication (IPC).

Convenience is another advantage that refers to the ease and simplicity with which cooperating processes can communicate and share resources.

Another benefit is increased computational speed if IPC is used to communicate between processes.

Modularity is yet another benefit of process cooperation where an architecture can be broken down into different modules that cooperate using IPC.

Two common models of IPC are shared memory and message passing.

## Shared Memory

Shared memory is a technique used in inter-process communication (IPC) where multiple processes can share a common memory area. One process will create a memory portion which other processes can access if allowed. It is an efficient way of sharing data between cooperating processes.

**Example:** Process A generate information about certain resources and keeps records in shared memory. When process B needs to use that information, it will check the record stored in shared memory and take note of the information generated by process A and act accordingly. Thus, processes can use shared memory for extracting information as a record from other process as well as for delivering any specific information to other process.

## Producer-Consumer Problem

The producer-consumer problem is a classic example of inter-process communication and synchronization. It involves two processes, the producer and the consumer, which share a common buffer.

The producer generates data and puts it into a shared region, while the consumer takes data out of the region and processes it. The problem arises when the producer tries to put data into a full region, or when the consumer tries to take data out of an empty region.

To solve this problem, we use a buffer of items (in the shared region) that can be filled by the producer and emptied by the consumer.

There are two types of buffers:

### Bounded-Buffer

In the bounded-buffer solution, the buffer has a fixed size and can hold only a limited number of items. When the buffer is full, the producer must wait until the consumer removes an item. Similarly, when the buffer is empty, the consumer must wait until the producer adds an item. This ensures that the producer and consumer do not access the buffer at the same time and avoids any race conditions.

### Unbounded-Buffer

In the unbounded-buffer solution, there is no limit on the size of the buffer. The producer can keep producing items and adding them to the buffer, but the consumer can keep consuming items only they become available.

**TYPES OF MESSAGE PASSING MODEL**

**- Links are established through:**

**1. Direct or Indirect**

1. **Synchronous or Asynchronous**
2. **Buffering**

## Message Passing

Message passing is a way for processes to communicate with each other. When two processes want to talk to each other, they first set up a communication link between them. Once this link is established, they can start sending messages back and forth to each other.

There are two basic actions that are used in message passing: sending and receiving. To send a message, a process specifies the message content and the destination process. To receive a message, a process waits for an incoming message and then reads it.

Unlike shared memory, message passing does not require processes to have a shared memory area to exchange information. Instead, processes communicate by sending messages through the established communication link.

## TYPES OF MESSAGE PASSING MODEL

Links are established through:

* Direct or Indirect
* Synchronous or Asynchronous
* Buffering

### Direct or Indirect

In direct message passing, processes explicitly name each other and there is only one link between two processes and exchange information directly through it. The communication link can be unidirectional or bidirectional.

In indirect message passing, messages are sent to a mailbox or port, which is bound to a receiving process. The sender doesn't know which process will actually receive its message, and multiple processes can send messages to the same mailbox, allowing multi-process links. Mailboxes can be created, messages can be sent and received through the mailbox, and mailboxes can be destroyed.

### Synchronous or Asynchronous

Synchronous (blocking) and asynchronous (non-blocking).

In synchronous message passing, the sender is blocked until the message is received, and the receiver is blocked until a message is available.

In asynchronous message passing, the sender sends the message and continues, and the receiver receives either a valid message or a null message.

### Buffering

Buffering is a technique used to queue messages attached to the link. It can be implemented in three ways:

* Zero capacity (no messages are queued on a link) where sender must wait for receiver.
* Bounded capacity (a finite length of n messages) where Sender must wait if link full.
* Unbounded capacity (an infinite length) where the sender never waits.