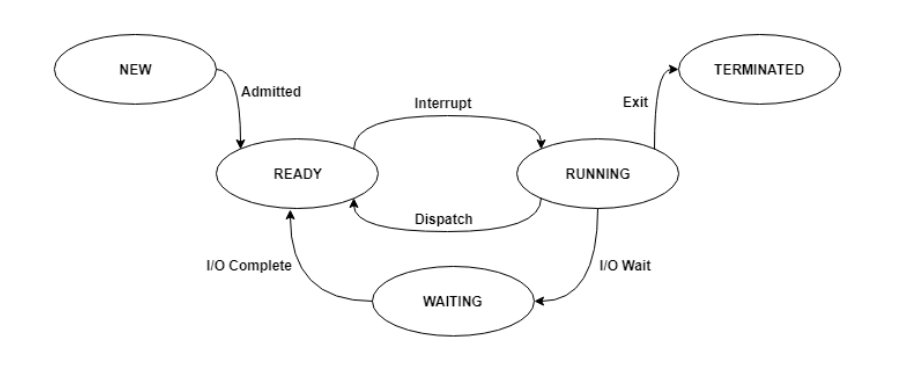
**Process Management**

A process is an active program i.e a program that is under execution. It contains the program code, program counter, process stack, registers etc.

**Process States**

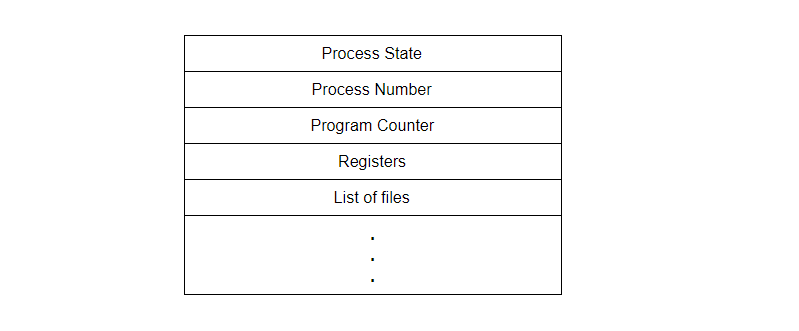
The different states that a process is in during its execution are explained using the following diagram −



* **New**- The process is in new state when it has just been created.<.li>
* **Ready** - The process is waiting to be assigned the processor by the short term scheduler.<.li>
* **Running** - The process instructions are being executed by the processor.<.li>
* **Waiting** - The process is waiting for some event such as I/O to occur.<.li>
* **Terminated** - The process has completed its execution.<.li>

**Process Control Block**

A process control block is associated with each of the processes. It contains important details about that particular process. These are as follows −



* **Process State** - This specifies the process state i.e. new, ready, running, waiting or terminated.
* **Process Number** - This shows the number of the particular process.
* **Program Counter** - This contains the address of the next instruction that needs to be executed in the process.
* **Registers** - This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.
* **List of files** - These are the different files that are associated with the process.

**Process Scheduling**

There are many scheduling queues that are used to handle processes. When the processes enter the system, they are put into the job queue. The processes that are ready to execute in the main memory are kept in the ready queue. The processes that are waiting for the I/O device are kept in the device queue.

The different schedulers that are used for process scheduling are:

**Long Term Scheduler**

The job scheduler or long term scheduler selects processes from the storage pool and loads them into memory for execution. The job scheduler must select a careful mixture of I/O bound and CPU bound processes to yield optimum system throughput. If it selects too many CPU bound processes then the I/O devices are idle and if it selects too many I/O bound processes then the processor has nothing to do.

**Short Term Scheduler**

The short term scheduler selects one of the processes from the ready queue and schedules them for execution. The short term scheduler executes much more frequently than the long term scheduler as a process may execute only for a few milliseconds.

**Medium Term Scheduler**

The medium term scheduler swaps out a process from main memory. It can again swap in the process later from the point it stopped executing. This is helpful in reducing the degree of multiprogramming. Swapping is also useful to improve the mix of I/O bound and CPU bound processes in the memory.

**Context Switching**

Removing a process from a CPU and scheduling another process requires saving the state of the old process and loading the state of the new process. This is known as context switching. The context of a process is stored in the Process Control Block (PCB) and contains the process register information, process state and memory information.

The dispatcher is responsible for context switching. It saves the context of the old process and gives control of the CPU to the process chosen by the short term scheduler

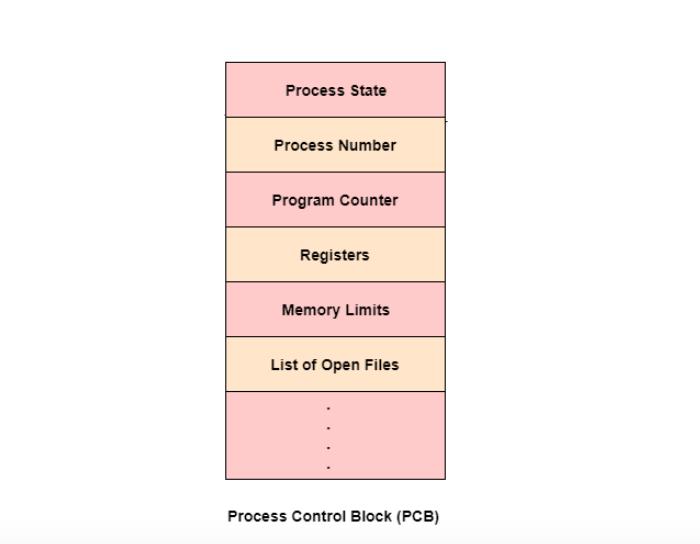
**PCB**

Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

**Structure of the Process Control Block**

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram −



The following are the data items −

**Process State**

This specifies the process state i.e. new, ready, running, waiting or terminated.

**Process Number**

This shows the number of the particular process.

**Program Counter**

This contains the address of the next instruction that needs to be executed in the process.

**Registers**

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

**List of Open Files**

These are the different files that are associated with the process

**CPU Scheduling Information**

The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

**Memory Management Information**

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

**I/O Status Information**

This information includes the list of I/O devices used by the process, the list of files etc.

**Accounting information**

The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

**Location of the Process Control Block**

The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

**Process Scheduling**

nterprocess 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.

A diagram that illustrates interprocess communication is as follows −



**Synchronization in Interprocess Communication**

Synchronization is a necessary part of interprocess communication. It is either provided by the interprocess 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 Interprocess Communication**

The different approaches to implement interprocess 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 interprocess 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 interprocess 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 interprocess communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of interprocess communication is as follows −

