Introduction of Process Management

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

There may exist more than one process in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

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

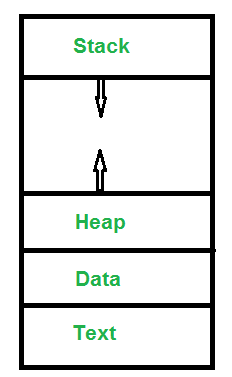
1. Scheduling processes and threads on the CPUs.
2. Creating and deleting both user and system processes.
3. Suspending and resuming processes.
4. Providing mechanisms for process synchronization.
5. Providing mechanisms for process communication.

**Program vs Process**

A process is defined as an entity which represents the basic unit of work to be implemented in the system.  
A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

A process is an ‘active’ entity, as opposed to a program, which is considered to be a ‘passive’ entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

**What does a process look like in memory?**   
 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 −



***Text Section****:*A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the ***Program Counter***.   
***Stack****:* The Stack contains the temporary data, such as function parameters, returns addresses, and local variables.   
***Data Section****:* Contains the global variable.   
***Heap Section****:* Dynamically allocated memory to process during its run time. 

**Attributes or Characteristics of a Process**   
A process has following attributes. 

**1. Process Id:** A unique identifier assigned by the operating system

**2. Process State:** Can be ready, running, etc.

**3. CPU registers:** Like the Program Counter (CPU registers must be saved and

restored when a process is swapped in and out of CPU)

**5. Accounts information:**

**6. I/O status information:** For example, devices allocated to the process,

open files, etc

**8. CPU scheduling information:** For example, Priority (Different processes

may have different priorities, for example

a short process may be assigned a low priority

in the shortest job first scheduling)

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below.

1. Process ID

When a process is created, a unique id is assigned to the process which is used for 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 the completion, goes through various states which are new, ready, running and waiting. We will discuss about them 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.

### 5. General Purpose Registers

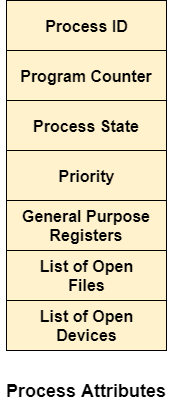
Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

### 6. List of open files

During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

### 7. List of open devices

OS also maintain the list of all open devices which are used during the execution of the process.



All of the above attributes of a process are also known as the***context of the process***.   
Every process has its own [process control block](http://en.wikipedia.org/wiki/Process_control_block)(PCB), i.e each process will have a unique PCB. All of the above attributes are part of the PCB.

**States of Process:**   
A process is in one of the following states: 

**1. New:** Newly Created Process (or) being-created process.

**2. Ready:** After creation process moves to Ready state, i.e. the

process is ready for execution.

**3. Run:** Currently running process in CPU (only one process at

a time can be under execution in a single processor).

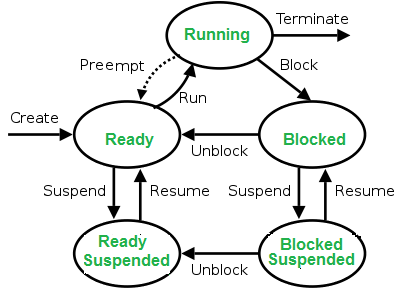
**4. Wait (or Block):** When a process requests I/O access.

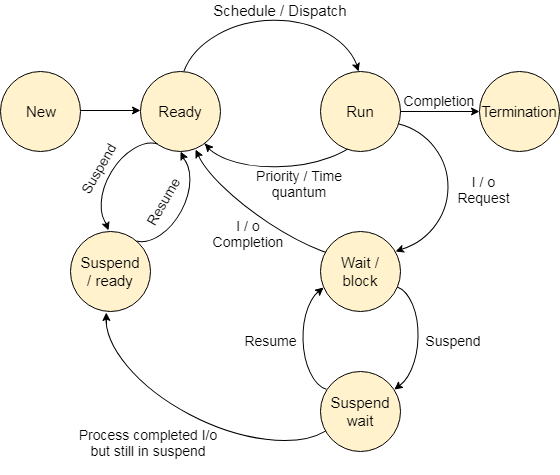
**5. Complete (or Terminated):** The process completed its execution.

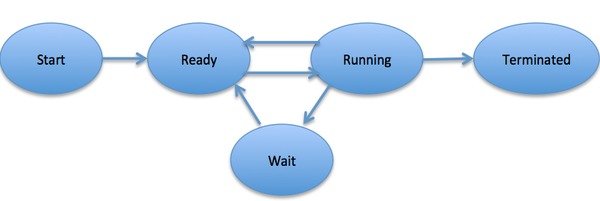
**6. Suspended Ready:** When the ready queue becomes full, some processes

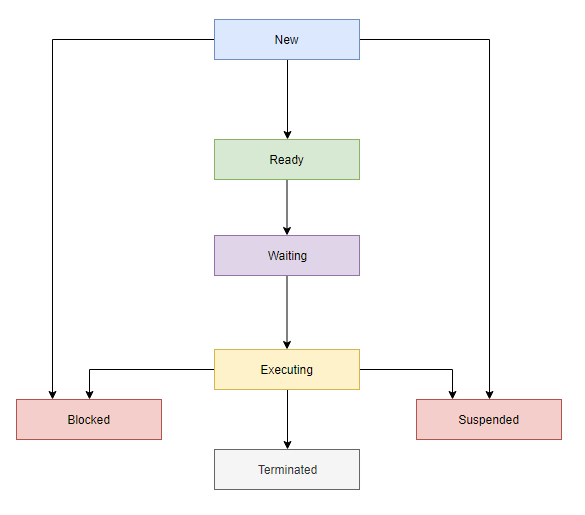
are moved to suspended ready state

**7. Suspended Block:** When waiting queue becomes full.









**Context Switching**   
The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from running state to ready state.

## 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 the execution.

### 2. Scheduling

Out of the many processes present in the ready queue, the Operating system chooses one process and start executing it. Selecting the process which is to be executed next, is known as scheduling.

### 3. Execution

Once the process is scheduled for the execution, the processor starts executing it. Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.

### 4. Deletion/killing

Once the purpose of the process gets over then the OS will kill the process. The Context of the process (PCB) will be deleted and the process gets terminated by the Operating system.

## Definition

The 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.

# Process Scheduling Queuing

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.

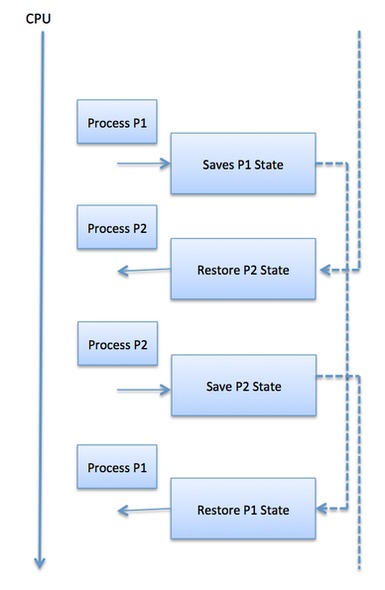
## Comparison among Scheduler

|  |  |  |  |
| --- | --- | --- | --- |
| **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. |

## Context Switch

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. **Using this technique, a context switcher enables multiple processes to share a single CPU.** Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

* Program Counter
* Scheduling information
* Base and limit register value
* Currently used register
* Changed State
* I/O State information
* Accounting information

# Process Schedulers

Operating system uses various schedulers for the process scheduling described below.

### 1. Long term scheduler

Long term scheduler is also known as job scheduler. It chooses the processes from the pool (secondary memory) and keeps them in the ready queue maintained in the primary memory.

Long Term scheduler mainly controls the degree of Multiprogramming. The purpose of long term scheduler is to choose a perfect mix of IO bound and CPU bound processes among the jobs present in the pool.

If the job scheduler chooses more IO bound processes then all of the jobs may reside in the blocked state all the time and the CPU will remain idle most of the time. This will reduce the degree of Multiprogramming. Therefore, the Job of long term scheduler is very critical and may affect the system for a very long time.

### 2. Short term scheduler

Short term scheduler is also known as CPU scheduler. It selects one of the Jobs from the ready queue and dispatch to the CPU for the execution.

A scheduling algorithm is used to select which job is going to be dispatched for the execution. The Job of the short term scheduler can be very critical in the sense that if it selects job whose CPU burst time is very high then all the jobs after that, will have to wait in the ready queue for a very long time.

This problem is called starvation which may arise if the short term scheduler makes some mistakes while selecting the job.

### 3. Medium term scheduler

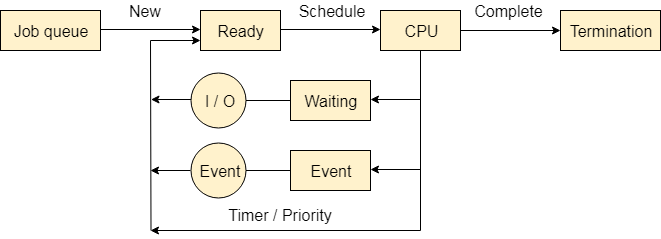
Medium term scheduler takes care of the swapped out processes.If the running state processes needs some IO time for the completion then there is a need to change its state from running to waiting.

Medium term scheduler is used for this purpose. It removes the process from the running state to make room for the other processes. Such processes are the swapped out processes and this procedure is called swapping. The medium term scheduler is responsible for suspending and resuming the processes.

It reduces the degree of multiprogramming. The swapping is necessary to have a perfect mix of processes in the ready queue.

# Process Queues

The Operating system manages various types of queues for each of the process states. The PCB related to the process is also stored in the queue of the same state. If the Process is moved from one state to another state then its PCB is also unlinked from the corresponding queue and added to the other state queue in which the transition is made.



There are the following queues maintained by the Operating system.

### 1. Job Queue

In starting, all the processes get stored in the job queue. It is maintained in the secondary memory. The long term scheduler (Job scheduler) picks some of the jobs and put them in the primary memory.

### 2. Ready Queue

Ready queue is maintained in primary memory. The short term scheduler picks the job from the ready queue and dispatch to the CPU for the execution.

### 3. Waiting Queue

When the process needs some IO operation in order to complete its execution, OS changes the state of the process from running to waiting. The context (PCB) associated with the process gets stored on the waiting queue which will be used by the Processor when the process finishes the IO.

## ****Process Hierarchy****

In a computer system, we require to run many processes at a time and some processes need to create other processes whilst their execution. When a process creates another process, then the parent and the child processes tend to associate with each other in certain ways and further. The child process can also create other processes if required. This parent-child like structure of processes form a hierarchy, called Process Hierarchy.

Unlike sexual reproduction though, the child processes have only one parent. A process can be a parent by itself and that is usually the case.

In UNIX, a process and all of its children and grandchildren form a process group. When a user sends a signal from the keyboard, the signal is delivered to all members of the process group currently associated with the keyboard. Individually, each process can catch the signal and do as it wishes.

There is no hierarchy system in Windows. Each process is treated equally. The only remote association with the hierarchy that it does is that a parent process gets a handle. But then again, this parent process can pass the handle to some other process thereby invalidating the hierarchy system.

## ****Two-State Process Model****

A Two-State Process Model categorizes a process in two categories:

1. A process running on CPU
2. A process not running on CPU

When a process is created, it is in the not in running state and when the CPU runs the process, it is in the running state. If there is a process in a running state and another process is created with a higher priority, then the newly created process will be run by the CPU. The former process will go in the not running state.

## ****Five-State Process Model****

The Five-State Process Model categorizes a process in five states:

1. **New:** When a process is newly created, it is said to be in the new state.
2. **Ready:** All those processes which are loaded into the primary memory and are waiting for the CPU are said to be in ready state.
3. **Running:** All the processes which are running are said to be in the running state.
4. **Waiting:** All the processes, which leave the CPU because of any reason (I/O or for any high priority process) and wait for their execution, are in waiting state.
5. **Terminated:** A process that exits or terminates from the CPU and the primary memory is said to be in the terminated state.

Concurrency in Operating System

**Concurrency** is the execution of the multiple instruction sequences at the same time. It happens in the operating system when there are several process threads running in parallel. The running process threads always communicate with each other through shared memory or message passing. Concurrency results in sharing of resources result in problems like deadlocks and resources starvation.

It helps in techniques like coordinating execution of processes, memory allocation and execution scheduling for maximizing throughput.

**Principles of Concurrency :**  
Both interleaved and overlapped processes can be viewed as examples of concurrent processes, they both present the same problems.  
The relative speed of execution cannot be predicted. It depends on the following:

* The activities of other processes
* The way operating system handles interrupts
* The scheduling policies of the operating system

**Problems in Concurrency :**

* **Sharing global resources –**  
  Sharing of global resources safely is difficult. If two processes both make use of a global variable and both perform read and write on that variable, then the order iin which various read and write are executed is critical.
* **Optimal allocation of resources –**  
  It is difficult for the operating system to manage the allocation of resources optimally.
* **Locating programming errors –**  
  It is very difficult to locate a programming error because reports are usually not reproducible.
* **Locking the channel –**  
  It may be inefficient for the operating system to simply lock the channel and prevents its use by other processes.

**Advantages of Concurrency :**

* **Running of multiple applications –**  
  It enable to run multiple applications at the same time.
* **Better resource utilization –**  
  It enables that the resources that are unused by one application can be used for other applications.
* **Better average response time –**  
  Without concurrency, each application has to be run to completion before the next one can be run.
* **Better performance –**  
  It enables the better performance by the operating system. When one application uses only the processor and another application uses only the disk drive then the time to run both applications concurrently to completion will be shorter than the time to run each application consecutively.

**Drawbacks of Concurrency :**

* It is required to protect multiple applications from one another.
* It is required to coordinate multiple applications through additional mechanisms.
* Additional performance overheads and complexities in operating systems are required for switching among applications.
* Sometimes running too many applications concurrently leads to severely degraded performance.

**Issues of Concurrency :**

* **Non-atomic –**  
  Operations that are non-atomic but interruptible by multiple processes can cause problems.
* Race conditions**–**  
  A race condition occurs of the outcome depends on which of several processes gets to a point first.
* **Blocking –**  
  Processes can block waiting for resources. A process could be blocked for long period of time waiting for input from a terminal. If the process is required to periodically update some data, this would be very undesirable.
* Starvation**–**  
  It occurs when a process does not obtain service to progress.
* Deadlock**–**  
  It occurs when two processes are blocked and hence neither can proceed to execute.

**Summary:**

* A process is defined as the execution of a program that performs the actions specified in that program.
* Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock.
* The important elements of Process architecture are 1)Stack 2) Heap 3) Data, and 4) Text
* The PCB is a full form of Process Control Block. It is a data structure that is maintained by the Operating System for every process
* A process state is a condition of the process at a specific instant of time.
* Every process is represented in the operating system by a process control block, which is also called a task control block.

MCQ

**Exercise:**   
**1.** Which of the following need not necessarily be saved on a context switch between processes? (GATE-CS-2000)   
(A) General purpose registers   
(B) Translation lookaside buffer   
(C) Program counter   
(D) All of the above

**Answer (B)**

**Explanation:**   
In a process context switch, the state of the first process must be saved somehow, so that when the scheduler gets back to the execution of the first process, it can restore this state and continue. The state of the process includes all the registers that the process may be using, especially the program counter, plus any other operating system-specific data that may be necessary. A translation look-aside buffer (TLB) is a CPU cache that memory management hardware uses to improve virtual address translation speed. A TLB has a fixed number of slots that contain page table entries, which map virtual addresses to physical addresses. On a context switch, some TLB entries can become invalid, since the virtual-to-physical mapping is different. The simplest strategy to deal with this is to completely flush the TLB.

**2.** The time taken to switch between user and kernel modes of execution is t1 while the time taken to switch between two processes is t2. Which of the following is TRUE? (GATE-CS-2011)   
(A) t1> t2   
(B) t1 = t2   
(C) t1 < t2   
(D) nothing can be said about the relation between t1 and t2.

**Answer: (C)**   
**Explanation:** Process switching involves mode switch. Context switching can occur only in kernel mode.