Contents

[Difference between Program and Process 2](#_Toc107228418)

[Program 2](#_Toc107228419)

[Process: 2](#_Toc107228420)

[Program vs Process 2](#_Toc107228421)

[Type s of Processes 2](#_Toc107228422)

[Different States of a Process 3](#_Toc107228423)

[Different Types of Scheduler 3](#_Toc107228424)

[PCB (Process Control Block) 4](#_Toc107228425)

[How Process looks like inside the main memory 4](#_Toc107228426)

[Difference between Thread and Process 5](#_Toc107228427)

[Process Scheduling 5](#_Toc107228428)

[Context Switching 6](#_Toc107228429)

[Zombie and Orphan Process 6](#_Toc107228430)

Process Concept

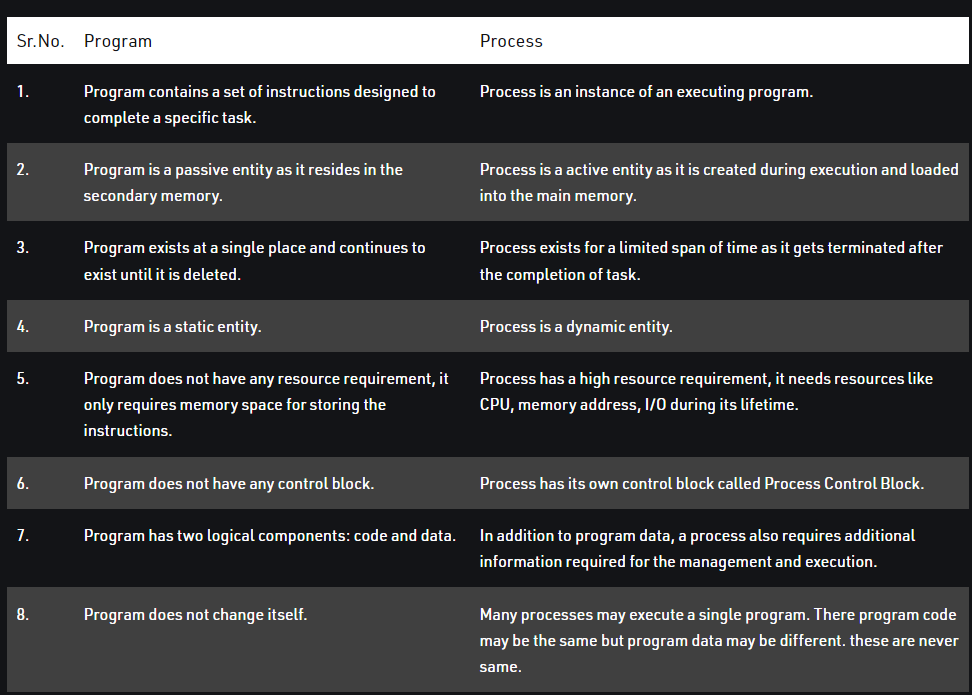
# Difference between Program and Process

Program : We open text editor and write a code and save that code in our secondary storage with a name and an extension. That file is known as program. For example fibonacci.c is a program file.

A program is a passive entity as it resides in the secondary memory, such as the contents of a file stored on disk. One program can have several processes.

Process: A program in execution is known as Process.

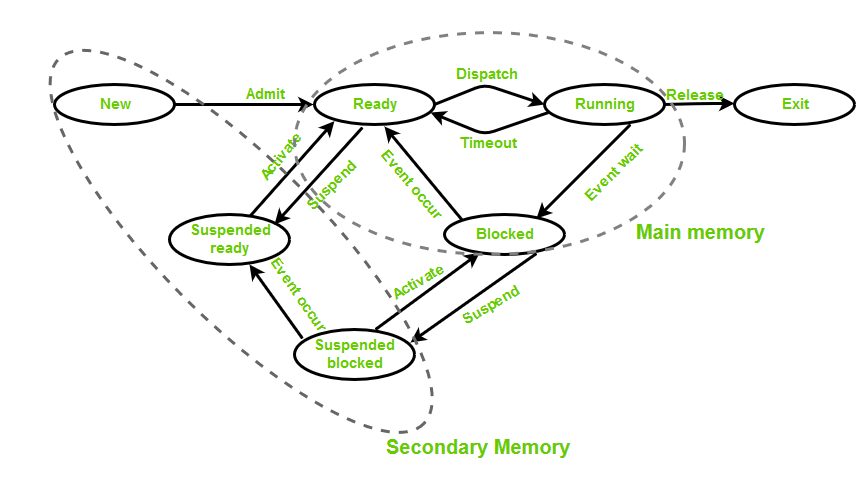
## Program vs Process



## Type s of Processes

* CPU bound: Most instructions are arithmetic instruction.
  + Sum a+b.
* I/O bound: Most of the instructions are input output instruction.
  + Take values from user
  + Show output to the User.

# Different States of a Process



* New (Create) – In this step, the process is about to be created but not yet created, it is the program which is present in secondary memory that will be picked up by OS to create the process.
* Ready – New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue for ready processes.
* Run – The process is chosen by CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
* Blocked or wait – Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or wait state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
* Terminated or completed – Process is killed as well as PCB is deleted.
* Suspend ready – Process that was initially in the ready state but was swapped out of main memory(refer Virtual Memory topic) and placed onto external storage by scheduler is said to be in suspend ready state. The process will transition back to ready state whenever the process is again brought onto the main memory.
* Suspend wait or suspend blocked – Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.

# Different Types of Scheduler

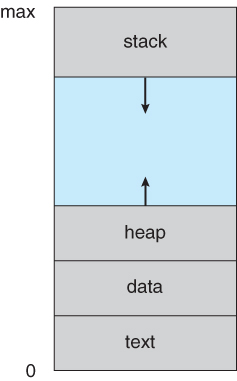
* Schedulers are special system software which handle process scheduling in various ways.
* Long term – performance – Makes a decision about how many processes should be made to stay in the ready state, this decides the degree of multiprogramming. Once a decision is taken it lasts for a long time hence called long term scheduler.
* Short term – Context switching time – Short term scheduler will decide which process to be executed next and then it will call dispatcher. A dispatcher is a software that moves process from ready to run and vice versa. In other words, it is context switching.
* Medium term – Swapping time – Suspension decision is taken by medium term scheduler. Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.

# PCB (Process Control Block)

Operating System maintains a data structure to store the regarding a process, that’s known as a Process Control Block. Process Control Block store remaining information regarding a Process:

* Process State: Whether process is ready, running, terminated, wait or in suspended ready state.
* Process Id : Process id is the unique identifier of a Process which is also known as a PID.
* Program Counter: It point to the instruction which will be executed by the CPU very next.
* Register: Store process data.
* Memory
* Open files

# How Process looks like inside the main memory

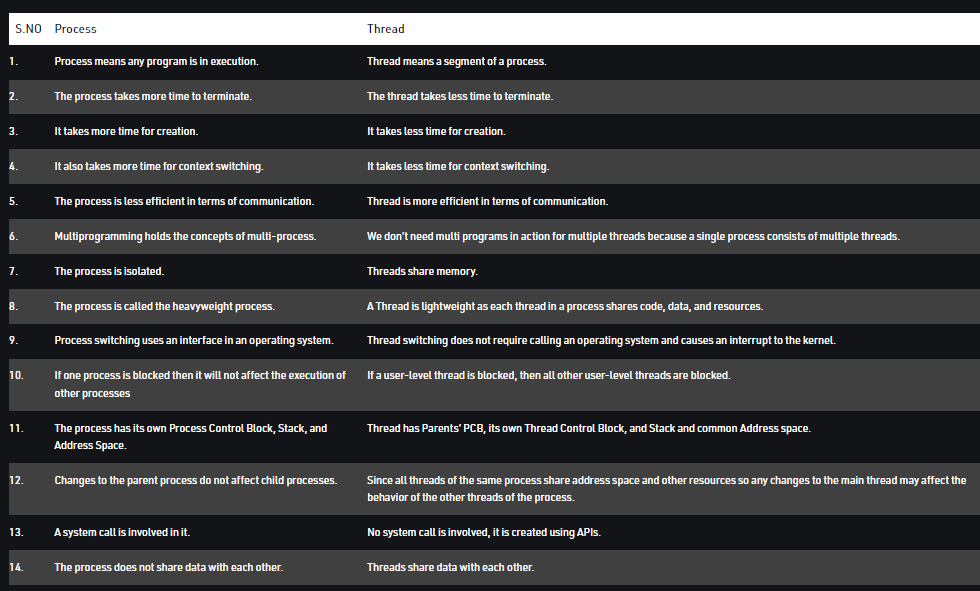


* Stack : It contains all local variable or functional variable and function calls. Create multiple stack spaces for multiple functions.
* Heap: It stores all dynamically created memory which are created using malloc or calloc.
* Data: It is also known as data section which contains global and static variable.
* Text: Text section contains Code as well as all macros.

# Difference between Thread and Process

Thread: Thread is the segment of a process which means a process can have multiple threads and these multiple threads are contained within a process. A thread has three states: Running, Ready, and Blocked.

The thread takes less time to terminate as compared to the process but unlike the process, threads do not isolate.



# Process Scheduling

It is a Procedure of deciding, which process present in the ready queue will be allocated to CPU right now by the short time Scheduler.

Different type of 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.



# Context Switching

When CPU switches its control from one process to another process due to any interrupt or any I/O operation, this is known as Context Switching.

While Context Switching, OS save the state of one process into PCB(p1) and read the state of another process from its PCB(p2).

Context Switching is a Pure overhead process.

# Zombie and Orphan Process

* A process which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from the process table.

The parent process reads the exit status of the child process which reaps off the child process entry from the process table. In the following code, the child finishes its execution using exit() system call while the parent sleeps for 50 seconds, hence doesn’t call wait() and the child process’s entry still exists in the process table.

* A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process. In the following code, parent finishes execution and exits while the child process is still executing and is called an orphan process now.

However, the orphan process is soon adopted by init process, once its parent process dies.