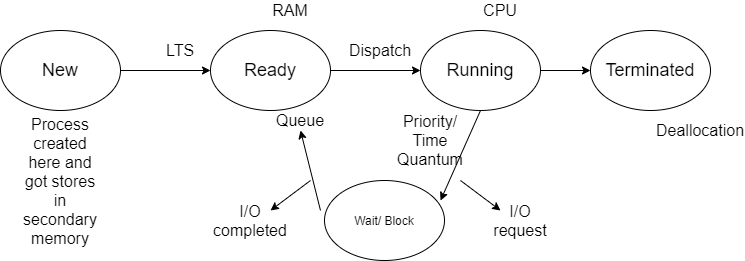
Process State in Operating System | Operating System – M01 P05

This is a multipart blog article series, and in this series I am going to explain you the concepts of operating system. This article series is divided into multiple modules and this is the first module which consists of 12 articles.

In this article we will learn about the “process state” in operating system. We will discuss about many topics such as “Long Term Scheduler”, “Short term scheduler”, “Primitive” and “Medium term scheduler”

**Process State:**

* Before “new” state there was no process, we had just created a process, or it may happen that when we open (switch on) our system many process get started automatically.
* When we initialize the process to start, first it get stored in RAM and the process get queued, and by this now process is in “Ready State”. Before initializing the process was in “new state” in secondary memory.



**Long term scheduler:**

* The main task of LTS is to send maximum number of process from “new state” (secondary memory) to “Ready state” (RAM) and it is done by using the concept of multiprogramming.
* Now from RAM the process will be transferred to CPU, which means from ready state to “running state”. If we have one CPU then uni-processing will occur, while if we have multiple CPU then parallel processing/ multiprocessing will occur. The process is still in RAM but now at different location/ address and it’s state is also changed from queue to executing.
* When the process is completed then it is “terminated” and then dislocation will happen, that means in starting some resources in the form of (RAM) was allocated to that process, which will be taken back from it as we have limited size of RAM. Now the state of that process is “completed”.
* There is a scenario, suppose CPU is executing a process and while the execution is going on another process with high priority came, so in that case the CPU will return the process to RAM and start executing the high priority process.

**Short term scheduler:**

* There is a concept of “Time sharing” this is the concept of multitasking. In this the CPU will execute a process for a fixed time interval and if the process get completed then it get terminated, otherwise it will send back that process to RAM and start executing another process for that time interval. This is done repeatedly due to which multitasking is achieved.

**Non Pre-emptive:** Non pre-emptive means that if CPU execute a single process at a time from start to finish without starting another process in between, then it is considered as non pre-emptive.



**Pre-emptive:**

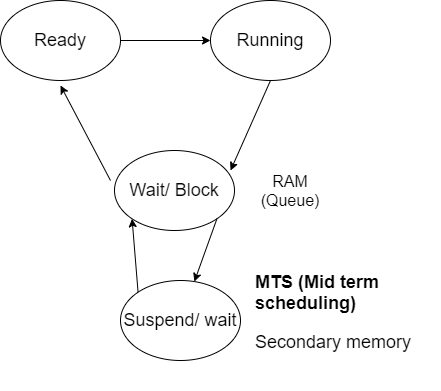
* When we stop a running process and start another process at its place due to high priority process or if time quantum is achieved then this is considered as pre-emptive.



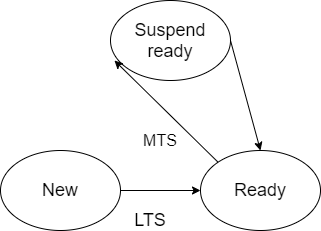
* There is another case, if CPU is running and the process demands for I/O operation, so in this case CPU can’t help process so it send the process to state which is known as “wait/ block” and CPU is known as executing another process. This is done because CPU runs fast while the process want to do some I/O operation, which will be done by the help of secondary memory and we know that secondary memory is slow. Hence the process gets shifted to another place in RAM because CPU has stop executing it. When I/O request of process is completed then it does not go to “running state” rather it goes to “ready state” and when that process chance will come CPU will start executing it.

**Medium term scheduler:**

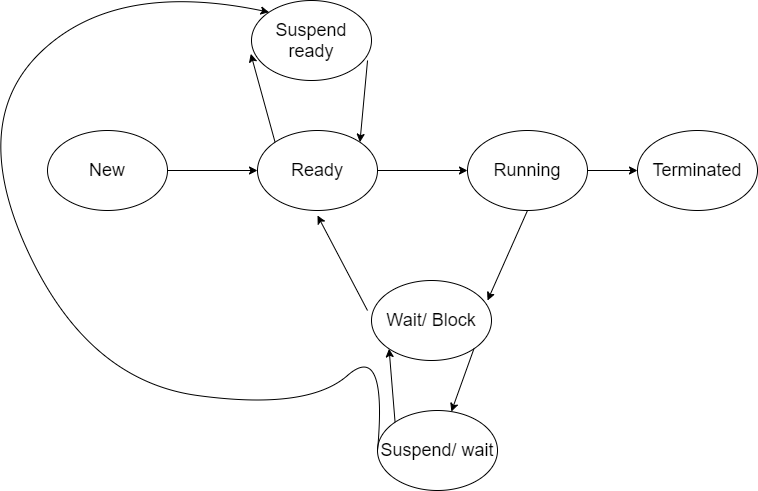
* There is a worst case scenario, in which suppose we have 10 process and all of them want to perform I/O operation, so in that case as size of RAM is limited some process will be sent to “suspend/ wait” state which is in secondary memory, and these process will complete their I/O operation there, and once there I/O request is completed, they came back to “wait/block” state which is situated in RAM and from there it will be sent to “ready” state which is also in RAM.



* There is another case that suppose “ready state” is filled with process, then a process came to it which is VVIP and have very high priority, then in that case some process from ready state will be sent to “ready suspend” state in order to make free space for VVIP process, and when there is space available in “ready state” the process came back to it.



* There is another exception you can understand it from the diagram below.



* When the process has completed its I/O operation but it is not getting chance to go back to “wait/block” state because it is already filled, then it can also go to “suspend ready” state and from there it can go to “ready state” which is the ultimate goal of the process.

**Note:**  You can get info of all process from “Task Manager” in windows, “PS command” in Linux (process state command)

Hope you like this article, and this article helped you, if you have any doubt, question, query related to this topic or wants to share something, then please feel free to contact me.