Processes & Threads

Lecture Outline

1 Processes

   i. Attributes of a Process

   ii. Process Life Cycle

   iii. Process Scheduling

   iv. Process Times

2 Threads

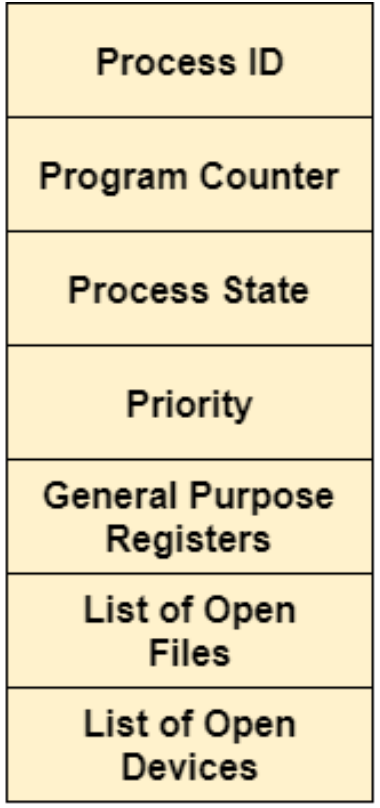
Process

A process is a program in execution

Attributes of a Process

The Attributes of the process are used by the Operating System to create the **process control block (PCB)** for each of them.

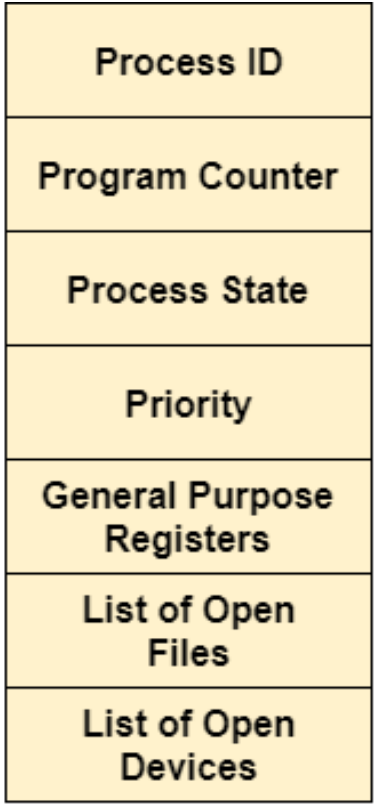
This is also called context of the process.



Attributes of a Process

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.

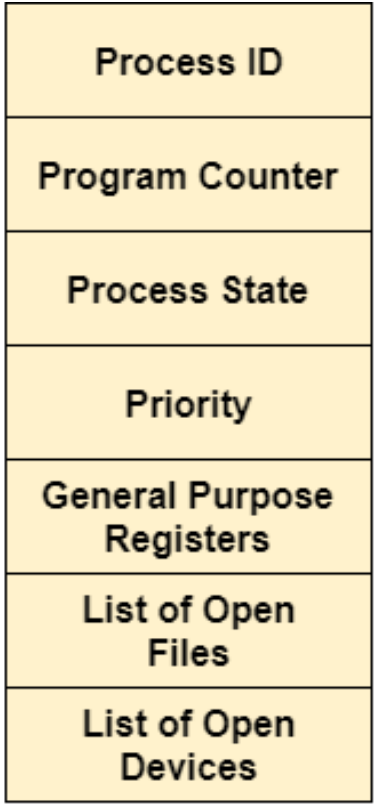


Attributes of a Process

Program Counter

A program counter stores the address of the last instruction of the process on which the process was suspended.

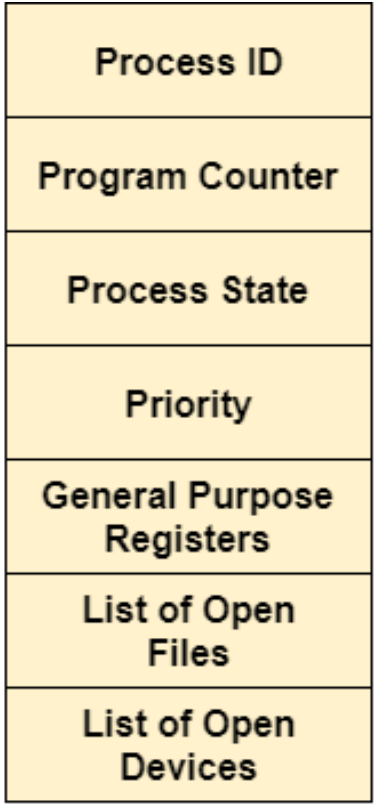
The CPU uses this address when the execution of this process is resumed.



Attributes of a Process

Process State

The process, from its creation to the completion, goes through various states which are new, ready, running and waiting.

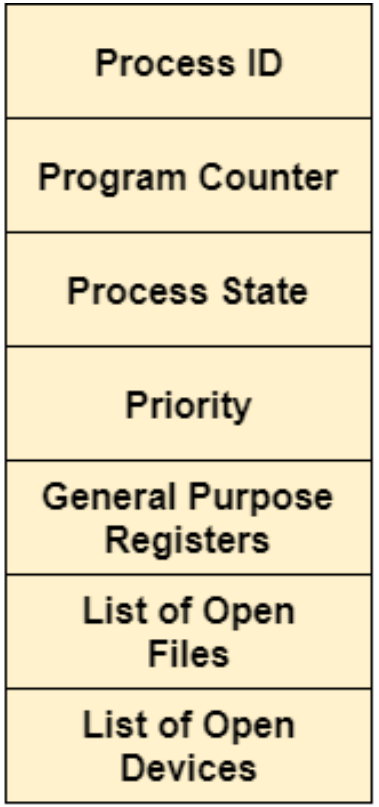


Attributes of a Process

Priority

Every process has its own priority.

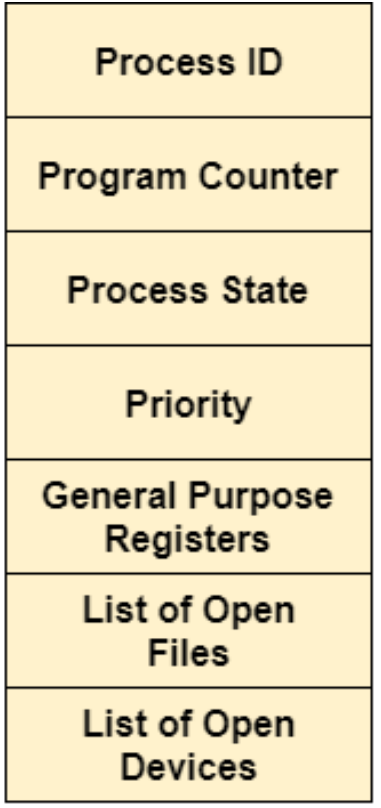
The process with the highest priority among the processes gets the CPU first.



Attributes of a Process

General Purpose Registers

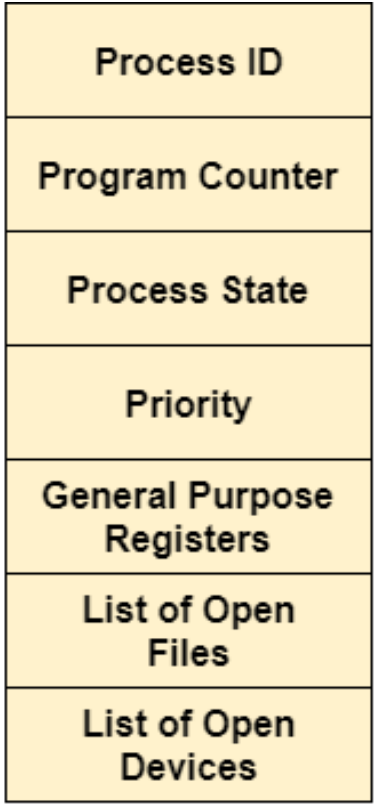
Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.



Attributes of a Process

List of Open Files

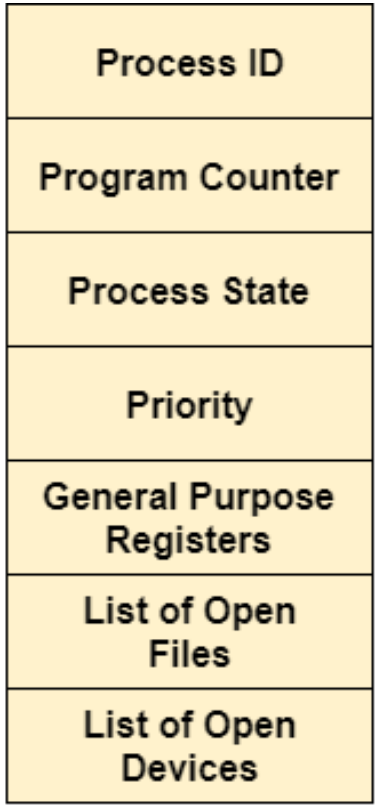
During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.



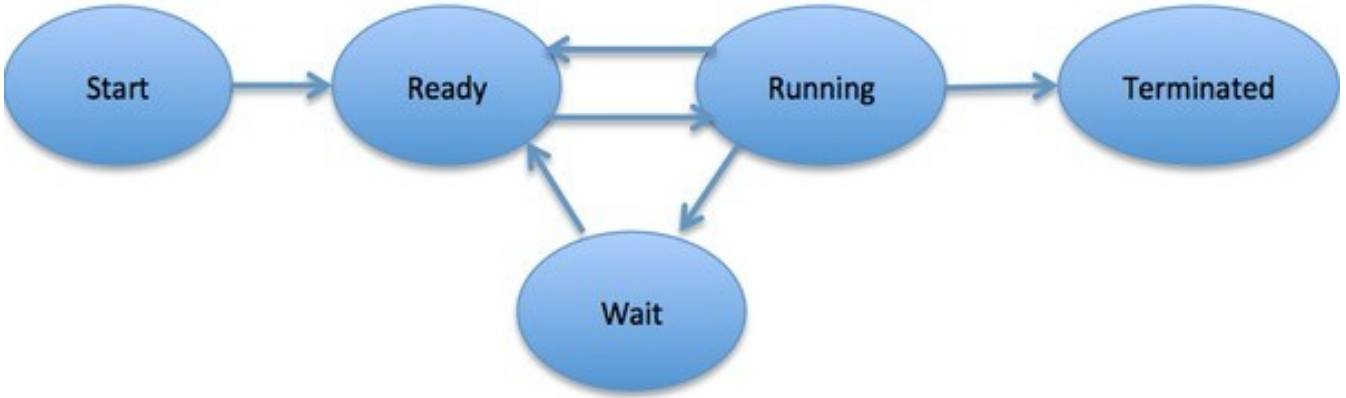
Attributes of a Process

List of Open Devices

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



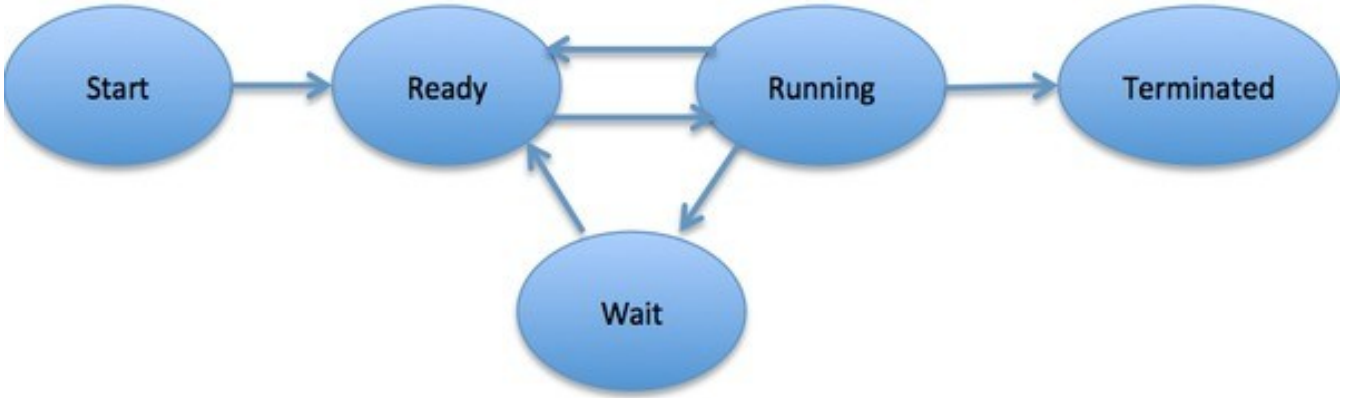
Process Life Cycle



Start

This is the initial state when a process is first started/created.

Process Life Cycle

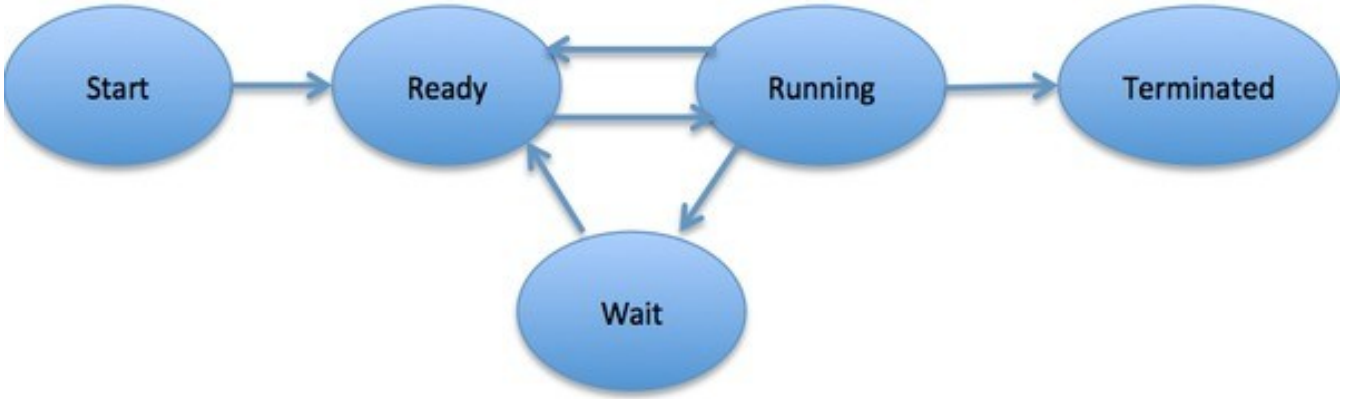


Ready

Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.

Process may come into this state after Start state or while running by being interrupted by the scheduler to assign CPU to some other process.

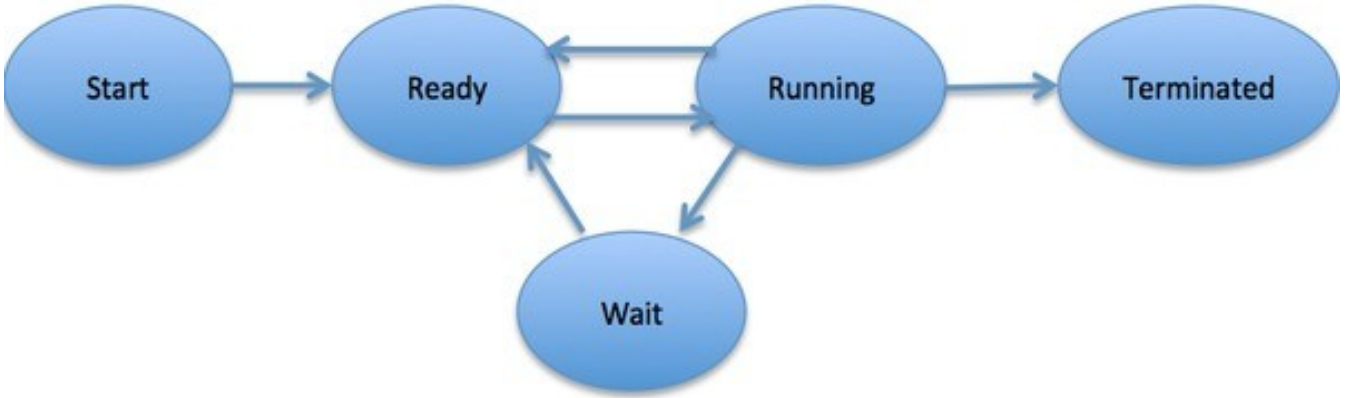
Process Life Cycle



Running

Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

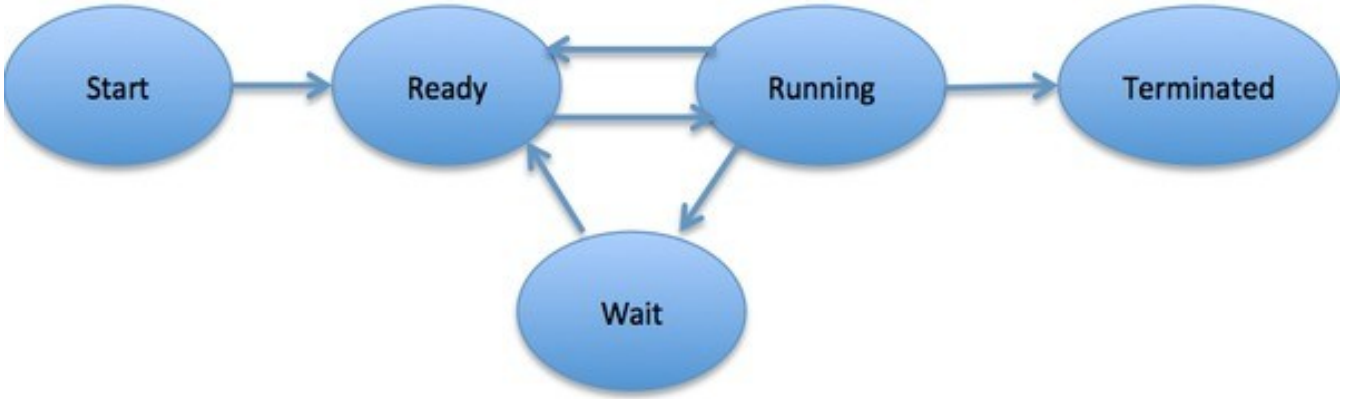
Process Life Cycle



Waiting

Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

Process Life Cycle



Terminated

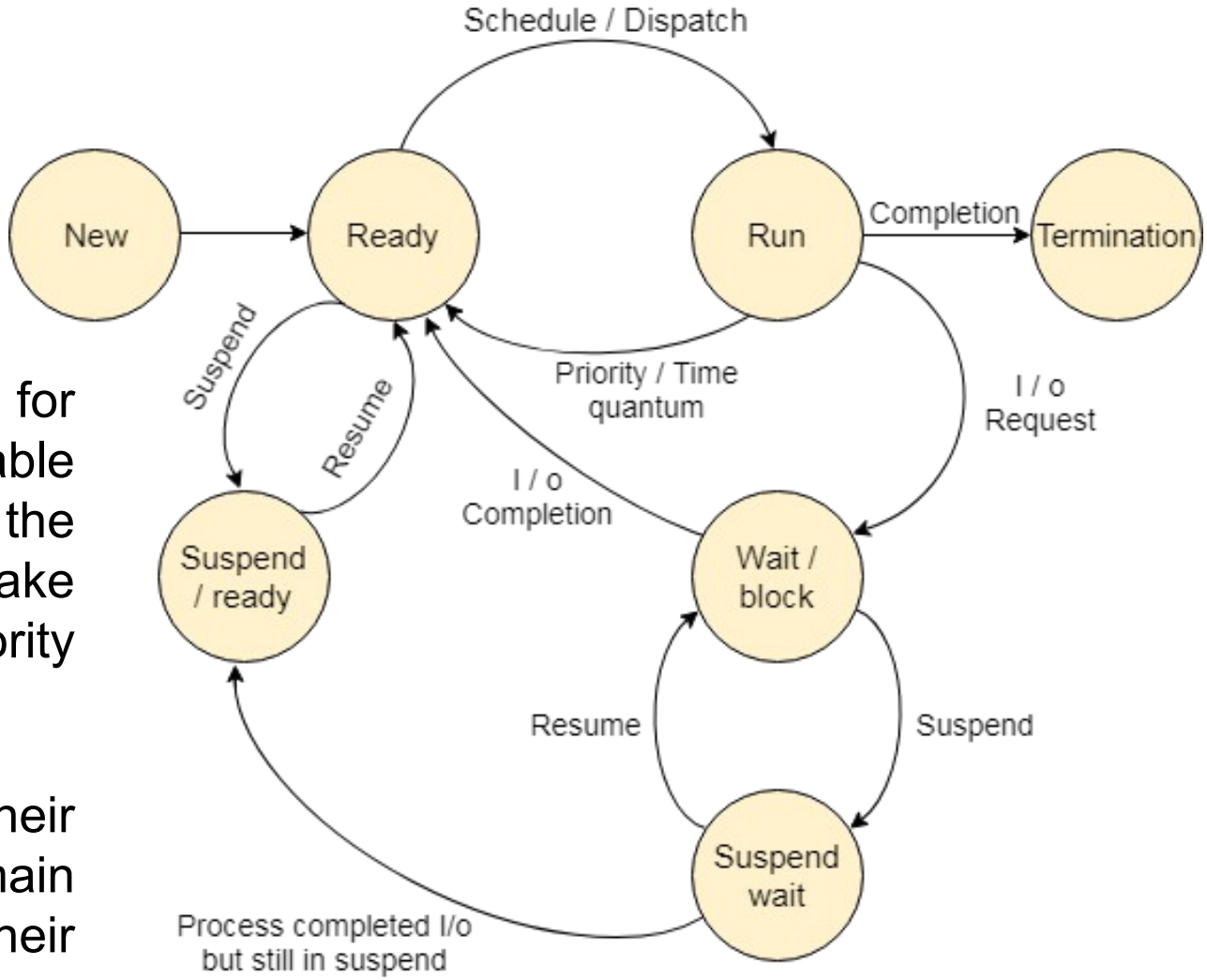
Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

Process Life Cycle

Suspend wait

When a process is waiting for some resource to get available it is better if it waits in the secondary memory and make room for the higher priority process.

These processes complete their execution once the main memory gets available and their wait is finished.

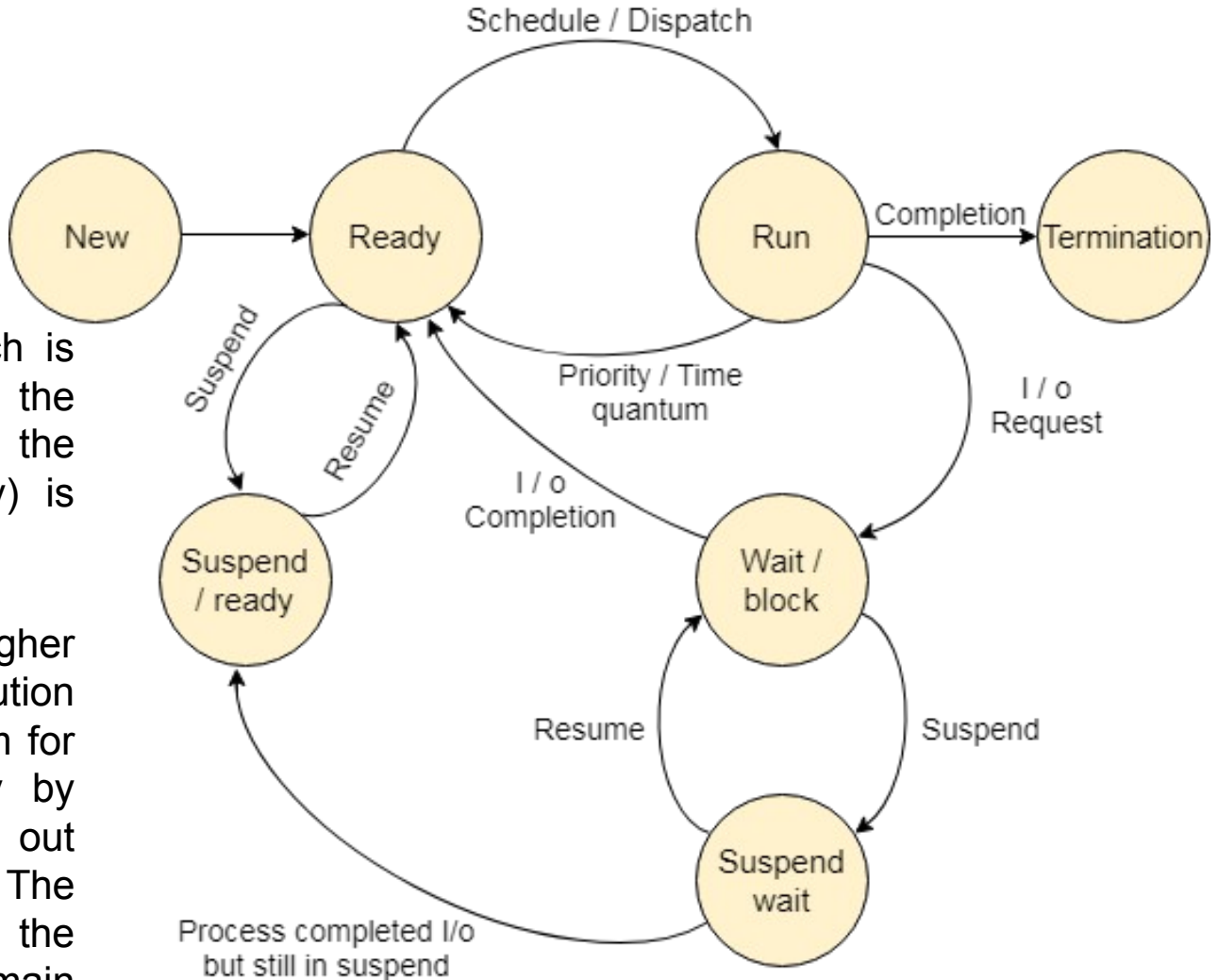


Process Life Cycle

Suspend ready

A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.

If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority process out into the secondary memory. The suspend ready processes remain in the secondary memory until the main memory gets available.

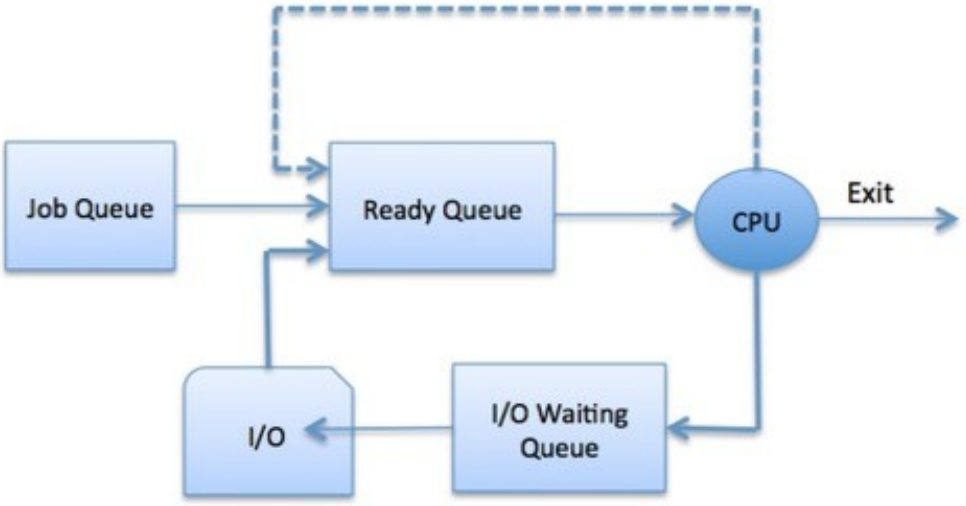


Process Scheduling

- 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

- **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 Schedulers

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.

Process Schedulers

Long term scheduler

- If the job scheduler chooses more I/O 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.

Process Schedulers

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.

Process Schedulers

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.

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.

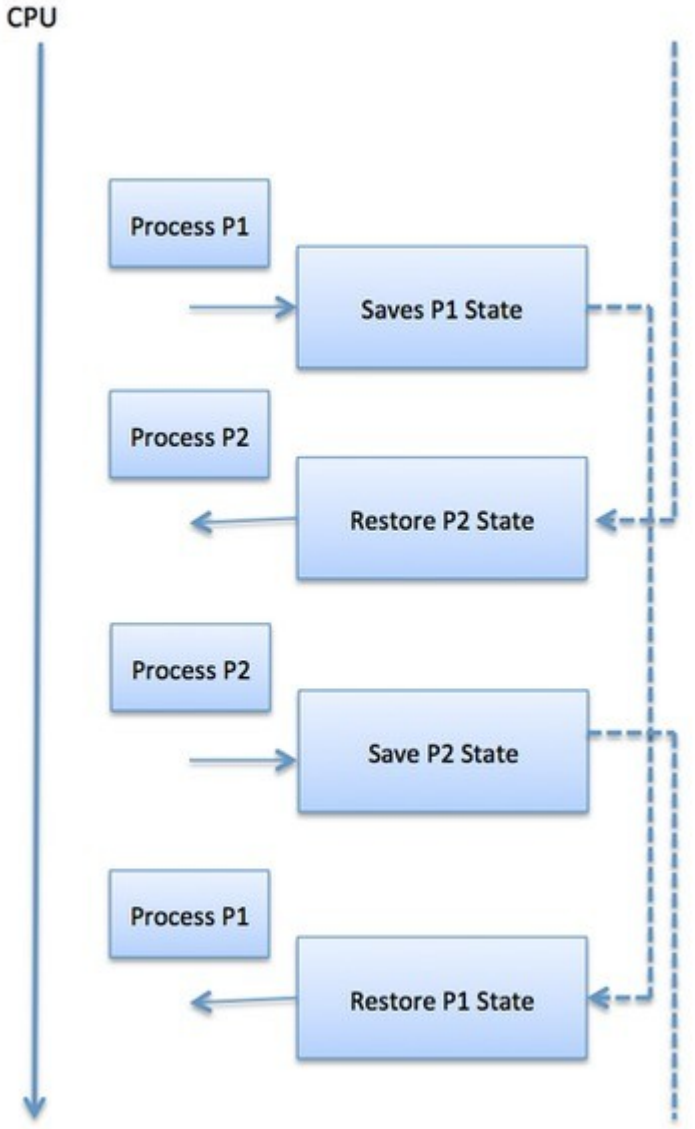
Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

Context Switch

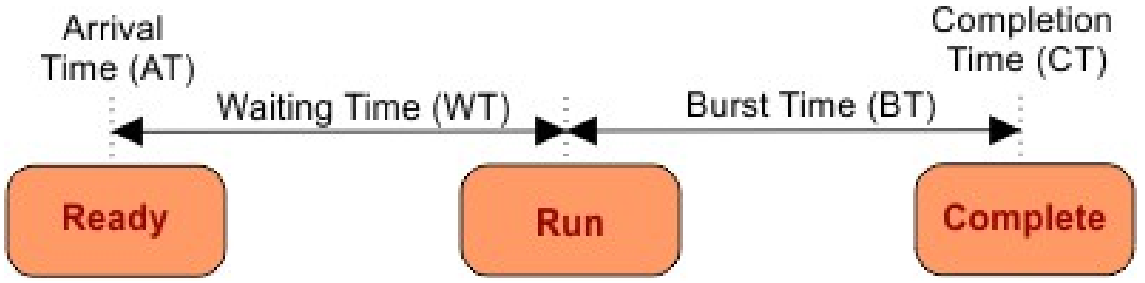
When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block.

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 Switch

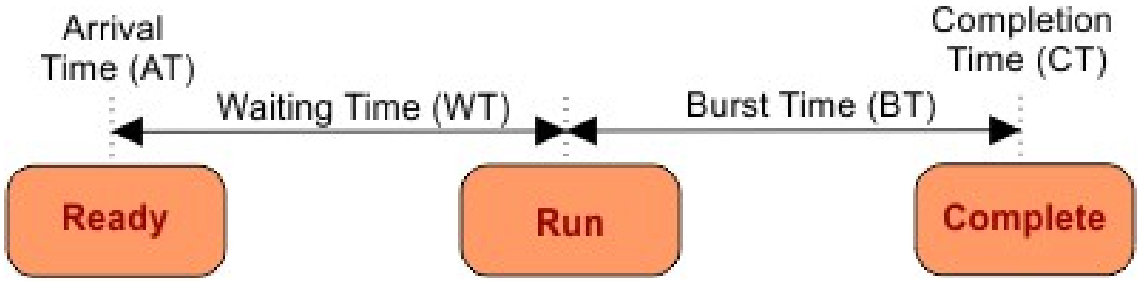


Process Times



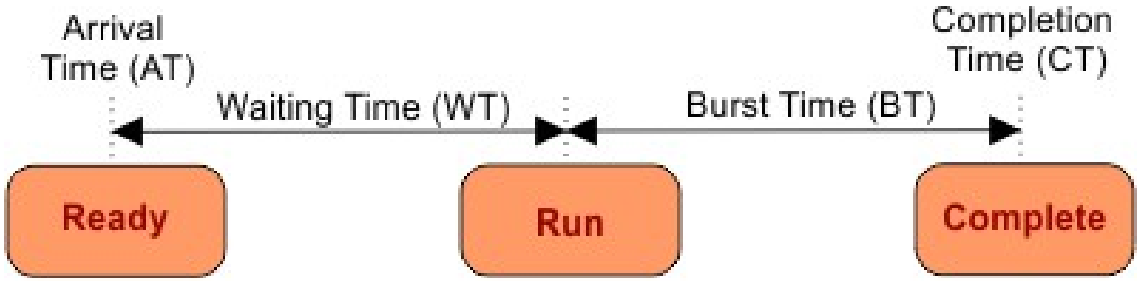
- **Arrival Time:** The Time at which a process arrives in the ready queue.
- **Completion Time:** The Time at which a process completes its execution.
- **Burst Time or Execution time:** The Total execution Time of a process on CPU. It is running time of a process on CPU. it has no concern with waiting time.

Process Times



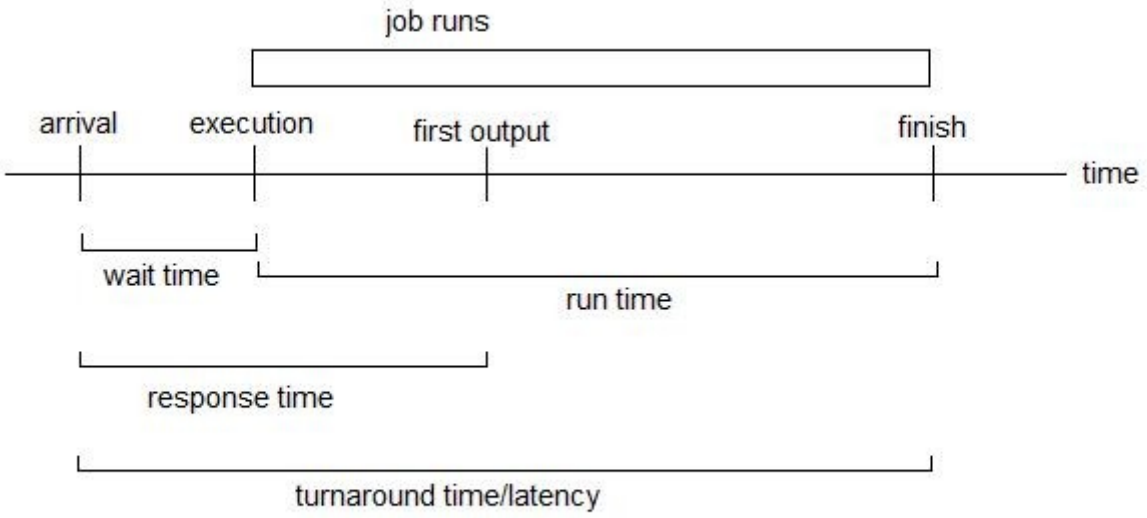
- **Turnaround Time:** = Waiting time + Burst time OR Turnaround Time = completion time – arrival time
- **Waiting Time:** Total time spends by a process when it enters into Ready Queue to completion of that process except the time of CPU execution for that process.

Process Times



- **Response time:** = the time at which a process gets CPU first time – arrival time
- **Throughput:** Number of processes that complete their execution per unit time is called Throughput.

Process Times



- **Arrival** - the first time you know you need to complete a task
- **Execution** - first instruction executed
- **first output** - the first time you know you need to complete a task
- **Execution ends** - last instruction finished
- **Wait time** - the time between the first arrival and when execution begins
- **Response time** - the time between arrival and the first output (useful for interactive applications)
- **Turnaround time (latency)** - the time between arrival and when the execution ends

Threads

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

Threads

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

Threads

A thread is also called a **lightweight process.**

Threads provide a way to improve application performance through parallelism.

Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Threads

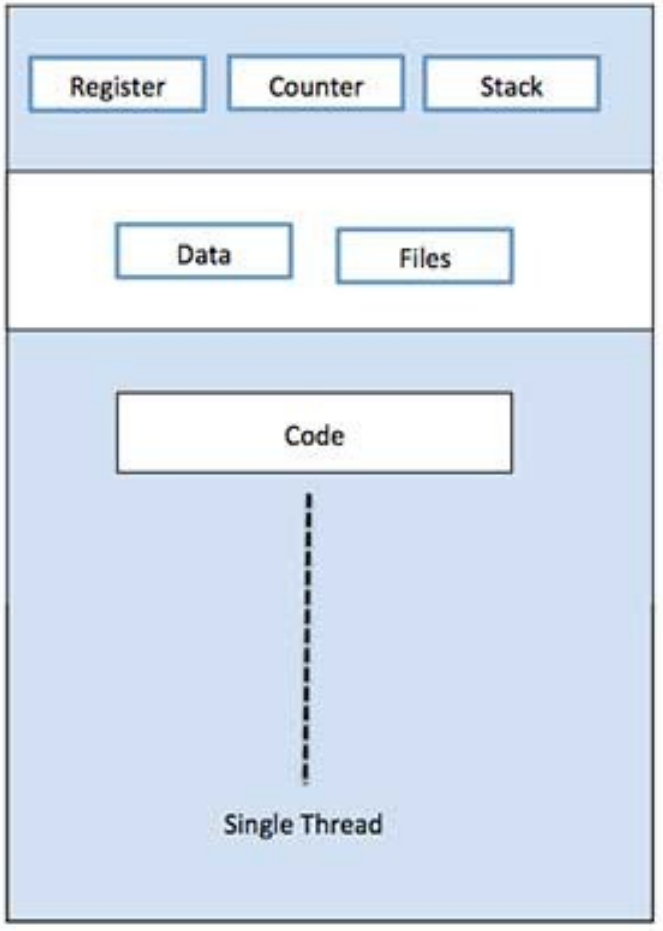
Each thread belongs to exactly one process and no thread can exist outside a process.

Each thread represents a separate flow of control.

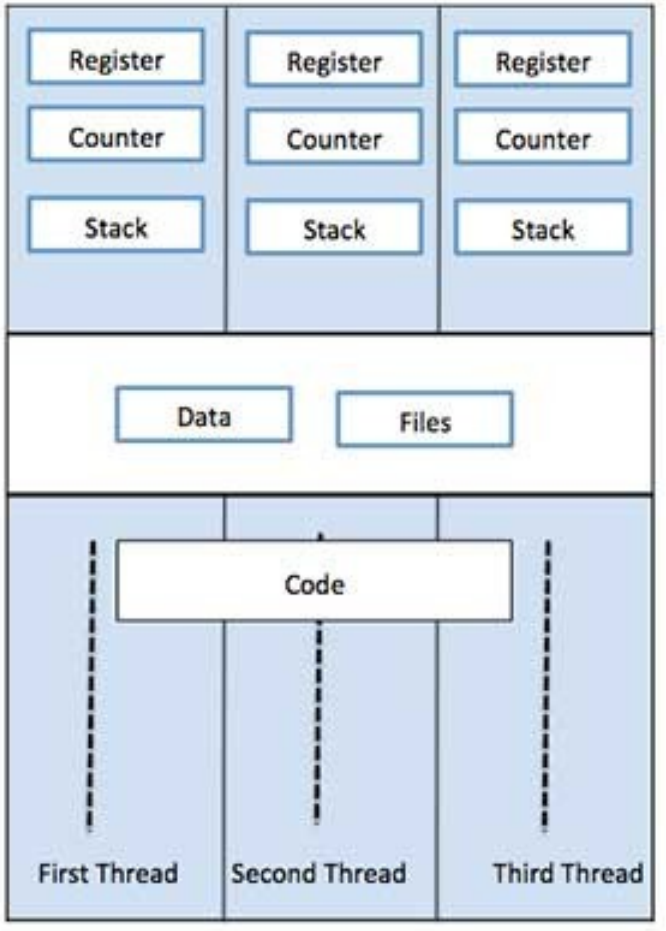
Threads have been successfully used in implementing network servers and web server.

They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.

Threads



Single Process P with single thread



Single Process P with three threads

Differences

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Threads

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Threads

Threads are implemented in following two ways −

- **User Level Threads** − User managed threads.
- **Kernel Level Threads** − Operating System managed threads acting on kernel, an operating system core.

Threads

https://youtu.be/h_HwkHobfs0

To be Continued …

Sources

- https://www.javatpoint.com/os-process-management-introduction
- https://www.tutorialspoint.com/operating_system/os_processes.htm
- http://web.cs.ucla.edu/classes/fall08/cs111/scribe/8/index.html
- https://cstaleem.com/cpu-scheduling-algorithms

Thanks