Operating System – Processes

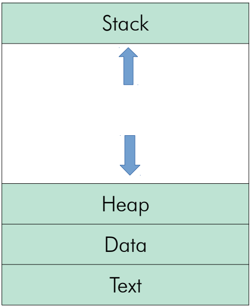
Process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

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:



Component	Description
Stack	The process Stack contains the temporary data such as method/function parameters, return address, and local variables.
Heap	This is a dynamically allocated memory to a process during its runtime.
Text	This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
Data	This section contains the global and static variables.

Program

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C & Java programming language:

C

```
#include <stdio.h>

int main() {
      printf("Hello, World! \n");
      return 0;
}
```

Java

```java
package test;

public class Test_Main {
       public static void main(String[] args){
          System.out.println ("Hello, World!");
     }
}
```

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

A part of a computer program that performs a well-defined task is known as an algorithm.
A collection of computer programs, libraries and related data are referred to as a software.

Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.
In general, a process can have one of the following five states at a time.

State	Description
Start	This is the initial state when a process is first

	started/created.
Ready	The process is waiting to be assigned to a processor. 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 it by but interrupted by the scheduler to assign CPU to some other process.
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.
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.
Terminated / Exited	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.

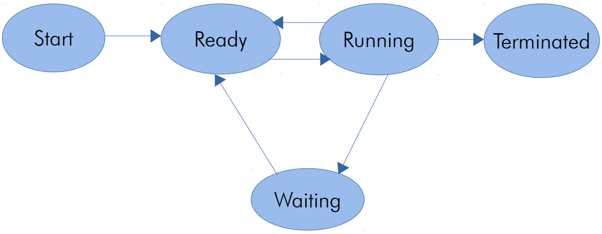


Fig : Block Diagram of Process Cycle Life

Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table:

Name	Information & Description
Process State	The current state of the process i.e., whether it is ready, running, waiting, or whatever.
Process Privileges	This is required to allow/disallow access to system resources.
Process ID	Unique identification for each of the process in the operating system.
Pointer	A pointer to parent process.
Program Counter	Program Counter is a pointer to the address of the next instruction to be executed for this process.
CPU Registers	Various CPU registers where process need to be stored for execution for running state.
CPU Scheduling Information	Process priority and other scheduling information which is required to schedule the process.
Memory management information	This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
Accounting information	This includes the amount of CPU used for process execution, time limits, execution ID etc.

IO status information	This includes a list of I/O devices allocated to the process.

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB:

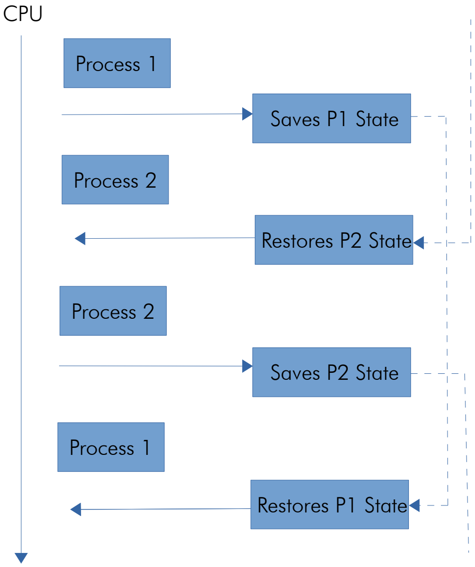
Process ID
State
Pointer
Priority
Program Counter
CPU Registers
I/O Information
Accounting Information
etc....

The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

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

Two-State Process Model

Two-state process model refers to running and non-running states which are described below.

State	Description
Running	When a new process is created, it enters into the system as in the running state.
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.