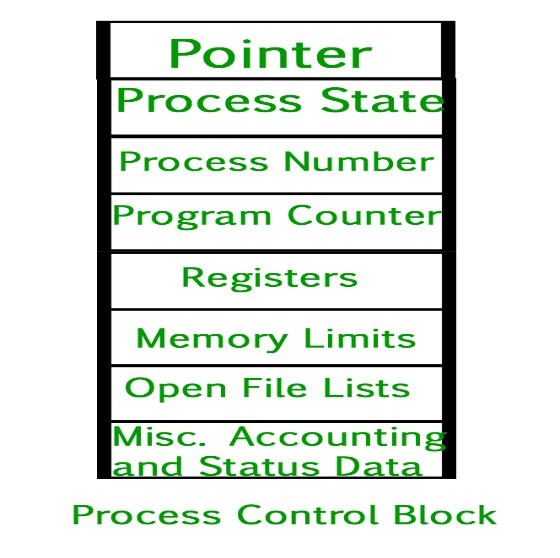
**Process Table and Process Control Block (PCB)**

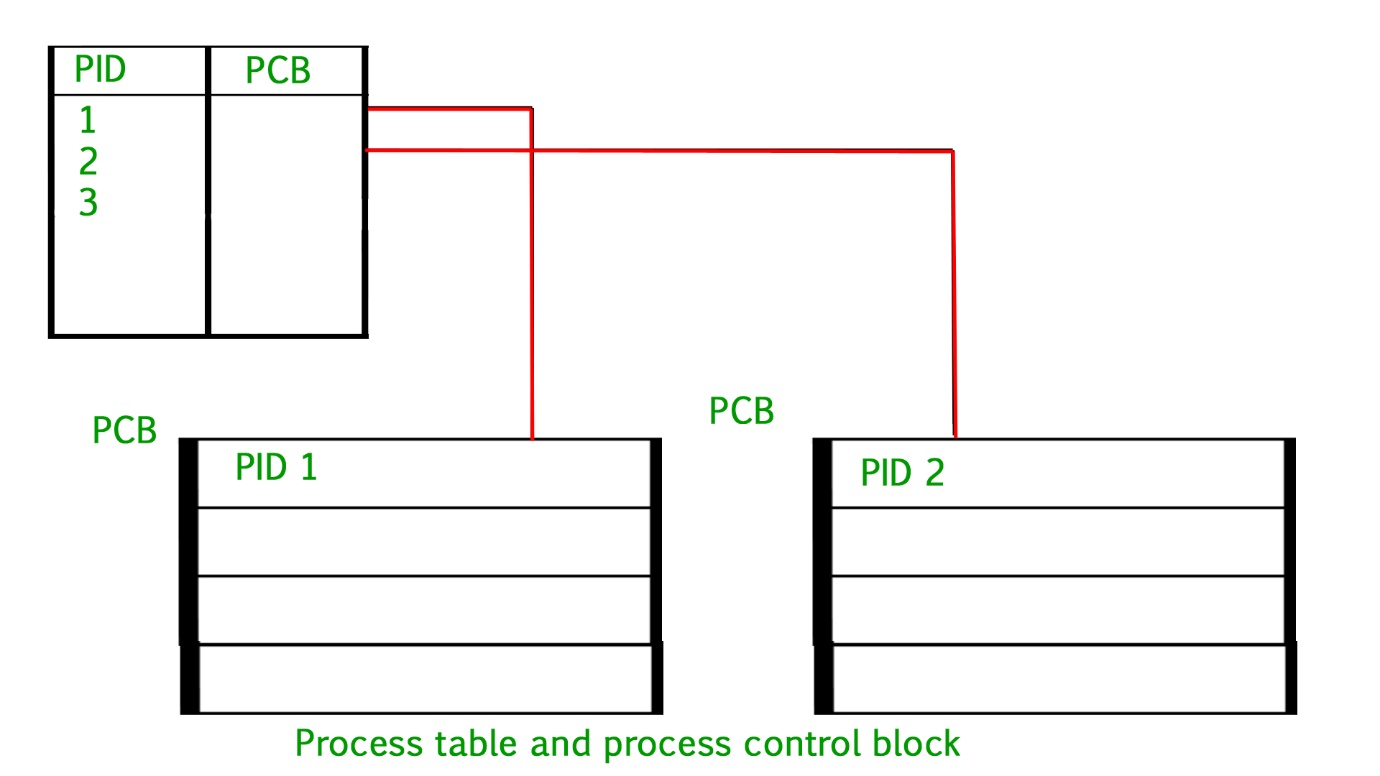
While creating a process the operating system performs several operations. To identify the processes, it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB) is used to track the process’s execution status. Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc. All these information is required and must be saved when the process is switched from one state to another. When the process makes a transition from one state to another, the operating system must update information in the process’s PCB.

A process control block (PCB) contains information about the process, i.e. registers, quantum, priority, etc. The process table is an array of PCB’s, that means logically contains a PCB for all of the current processes in the system.



* **Pointer –** It is a stack pointer which is required to be saved when the process is switched from one state to another to retain the current position of the process.
* **Process state –** It stores the respective state of the process.
* **Process number –** Every process is assigned with a unique id known as process ID or PID which stores the process identifier.
* **Program counter –** It stores the counter which contains the address of the next instruction that is to be executed for the process.
* **Register –** These are the CPU registers which includes: accumulator, base, registers and general purpose registers.
* **Memory limits –** This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.
* **Open files list –** This information includes the list of files opened for a process.

**Miscellaneous accounting and status data –** This field includes information about the amount of CPU used, time constraints, jobs or process number, etc.  
The process control block stores the register content also known as execution content of the processor when it was blocked from running. This execution content architecture enables the operating system to restore a process’s execution context when the process returns to the running state. When the process makes a transition from one state to another, the operating system updates its information in the process’s PCB. The operating system maintains pointers to each process’s PCB in a process table so that it can access the PCB quickly.



**Difference between Swapping and Context Switching**

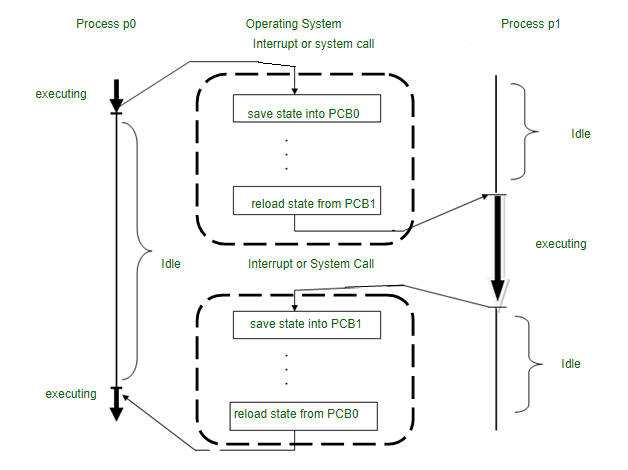
* Difficulty Level : [Medium](https://www.geeksforgeeks.org/medium/)
* Last Updated : 22 Nov, 2021

 Read

 Discuss

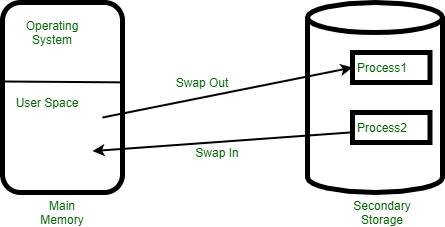
Programs are sets of instructions designed to accomplish specific tasks. Similarly, a process refers to a runtime instance of a computer program. During the execution of a program, several threads may be running in parallel. Single-threaded processes refer to the thread itself as the process.

**1. Context switching :**  
An operating system uses this technique to switch a process between states to execute its functions through CPUs. It is a process of saving the context(state) of the old process(suspend) and loading it into the new process(resume). It occurs whenever the CPU switches between one process and another. Basically, the state of CPU’s registers and program counter at any time represent a context. Here, the saved state of the currently executing process means to copy all live registers to PCB(Process Control Block). Moreover, after that, restore the state of the process to run or execute next, which means copying live registers’ values from PCB to registers.



**2. Swapping :**   
This is the process by which a process is temporarily swapped (moved) from the main memory (RAM) to the secondary memory (Disk). The main memory is fast but has less space than secondary storage, so the inactive processes are moved to secondary memory, and the system swaps the memory from secondary to the main memory later. During swapping, most of the time is spent transferring information, and the amount of memory swapped is directly proportional to the total time. Swapping has been divided into two more concepts: Swap-in and Swap-out.

Swap-in is the process of removing a program from a hard disk and moving it back to the main memory or RAM.  
Swap-out removes a program from RAM or main memory and moves or stores it to the hard disk or secondary storage.



**Difference between Swapping and Context Switching :**

| Context switching | Swapping |
| --- | --- |
| It is a procedure for storing the state of an old process and loading it into a new process. | Essentially, it is a method of replicating the entire process. |
| A context switch occurs when the kernel switches contexts when it transfers control of the CPU from one process to another already ready to run state. | Swapping happens when the entire process is moved to the disk. |
| A context switch determines whether a process is in the pause mode. | When it comes to Swapping, It deals with memory, how much memory is being swapped. |
| The context switch toggles the process from running to ready states, while the dispatcher is responsible for allocating CPU resources to processes present in the ready queue. | It is an OS term that we use for referring to the exchange of data between the disk and the main memory. |
| Active processes do context switching. | Inactive processes do swapping. |
| It offers a higher degree of multi-tasking. | It provides a more significant degree of multiprogramming. |
| It helps to get better utilization of the operating system. | It helps to get better utilization of memory. |

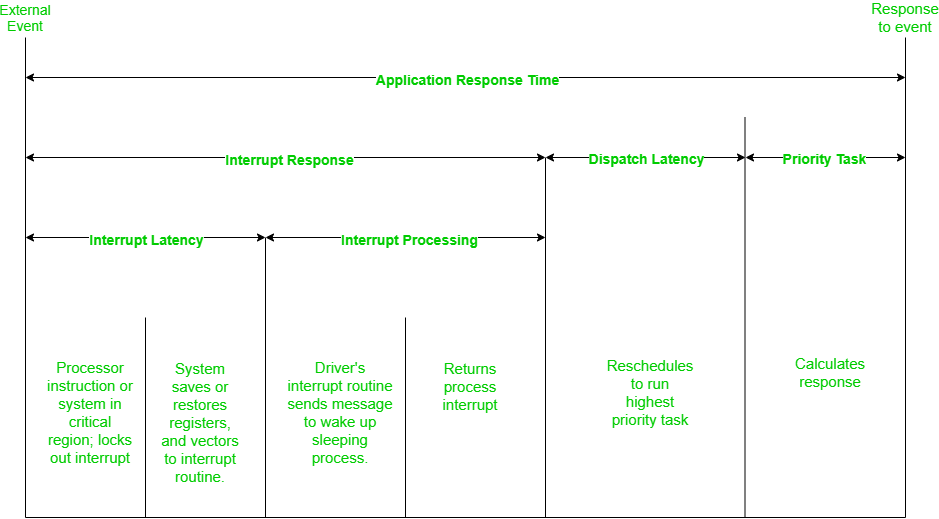
**Difference between “Dispatch Latency” and “Context Switch” in operating systems**

**Dispatcher** is a module that gives control of the CPU to the process selected by the short-term scheduler, this involves:

* switching context
* switching to user mode
* jumping to the proper location in the user program to continue executing that program.

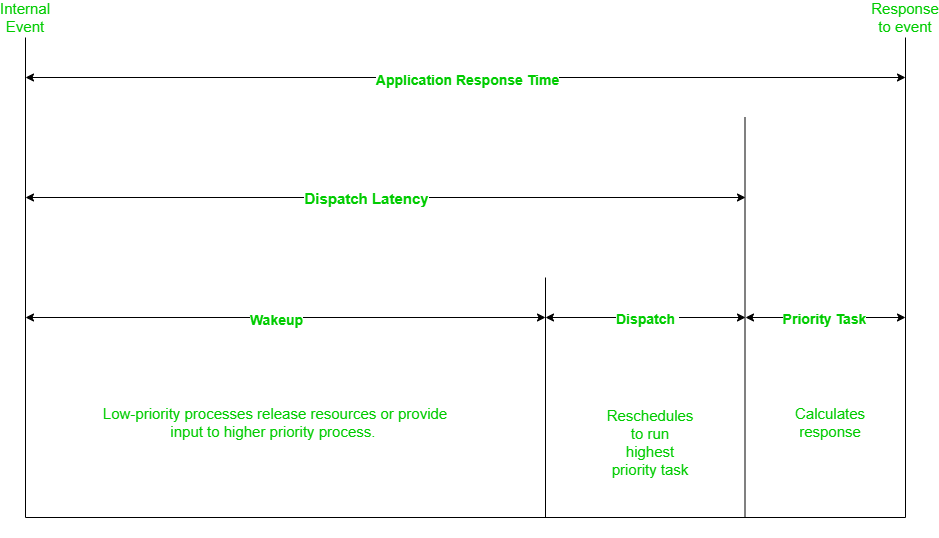
**Context Switching :**  
It is the process of storing the state of an old process and load the saved state for the new process via a context switch. Context of a process is represented in the PCB (process control block).  
During context switch a process’ attributes like registers, pointer, program counter etc which are all stored in the PCB are saved in a per-process stack in kernel memory and a new process takes its place by either updating its PCB if it’s an old process or creating a new PCB if it’s a new process.  
Context switching offers a high degree of multitasking and helps to get better utilization of the operating system.

**Dispatch Latency :**  
Itis the time taken by the dispatcher in context switching of a process from run state and putting another process in the run state.  
Dispatch latency is an overhead, and the system does no useful work while context switching.  
Some hardware provides multiple sets of registers per CPU which allows multiple contexts to be loaded at once.



*Position of Dispatch Latency inside an Application Response Time*

Dispatch latency is a small part of the whole application response time.



*Expanded View of Dispatch Latency inside Application Response Time*

Dispatch latency consists of an old task releasing its resources (wakeup) and then rescheduling the new task (dispatch), all of this also comes under context switching.  
Let’s look at an example to understand context switching and dispatch latency.

**Example –**

**Priority –**   
P1 > P2 > P3 > P4 > P5 > P6

**Ready Queue –**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P5 | P6 |

**Running –**

|  |
| --- |
| P1 |

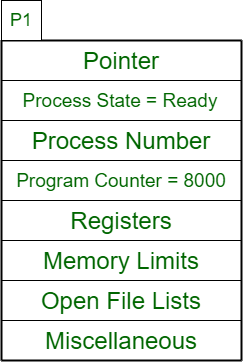
Task P7 with priority greater than P1 comes into ready queue.

**Now Ready Queue will be  –**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P5 | P6 | P7 |

Short term scheduler chooses P7 to swap with P1 in running state.

**Context Switching –**  
**1.**P1’s context (PCB) is saved and stored in a per-process stack in kernel memory.



**2.** P7’s context (PCB) is updated.



The time elapsed in steps 1 and 2 is the **dispatch latency**.

**Running –**

|  |
| --- |
| P7 |

Both “Context Switching” and “Dispatch Latency” are related to the dispatcher, which comes into play after the short term scheduler decides which process to bring from the ready queue to the running state.

| Dispatch Latency | Context Switching |
| --- | --- |
| The amount of time taken by the dispatcher to pause one process  and start another is called dispatch latency. | The process of saving the state of a previously running process or thread  and loading the initial or previously saved state of a new process by the dispatcher. |
| Dispatch latency is a time value. | Context switching is a process. |
| It is a consequence of context switching. | It is performed by the dispatcher, when initiated by an interrupt. |
| Latency is dependent on hardware support. | The more complex the OS and PCB, the longer the context switch. |