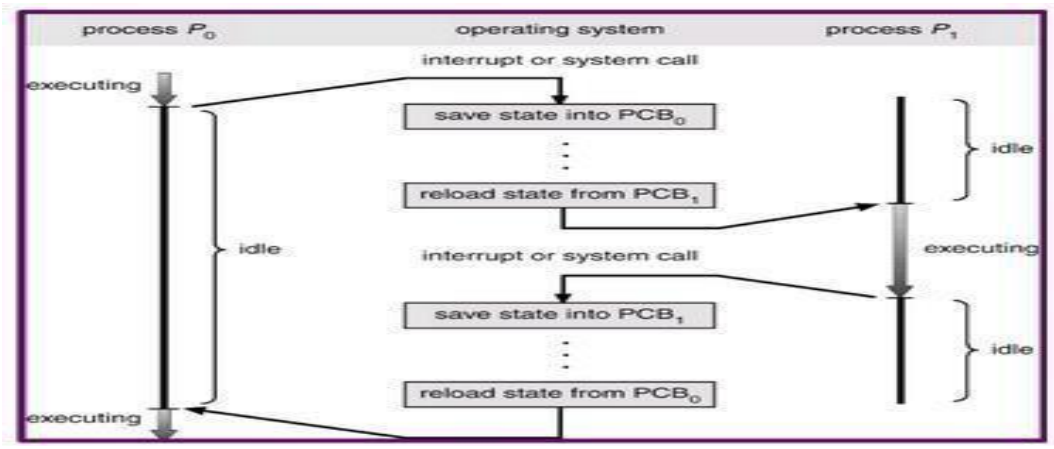
Context Switching

In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state is executed by the CPU. When a running process moves to waiting state or ready state, the state or context of the outgoing process needs to be saved into it's own PCB so that it can be reloaded when required and execution can be resumed from the same point as earlier. So when an interrupt occurs, the system needs to save the current context of the process currently running on the CPU so that it can restore that context when it's processing is done , essentially suspending the process and resuming it. The context is represented in the PCB of the process.
The attributes of a process stored in PCB are also known as the context of the process**.** When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch.
Context switching can happen due to the following reasons:

- When a process of high priority comes in the ready state. In this case, the execution of the running process should be stopped and the higher priority process should be given the CPU for execution.
- When an interruption occurs then the process in the running state should be stopped and the CPU should handle the interrupt before doing something else.
- When a transition between the user mode and kernel mode is required then context switching takes place.


process P_0
operating system
process P_1
interrupt or system call
executing
save state into PCB_0
idle
reload state from PCB_1
idle
interrupt or system call
executing
save state into PCB_1
idle
reload state from PCB_0
executing