Unit-II
Lecture: 7
(Process Management)
(Part-IV)

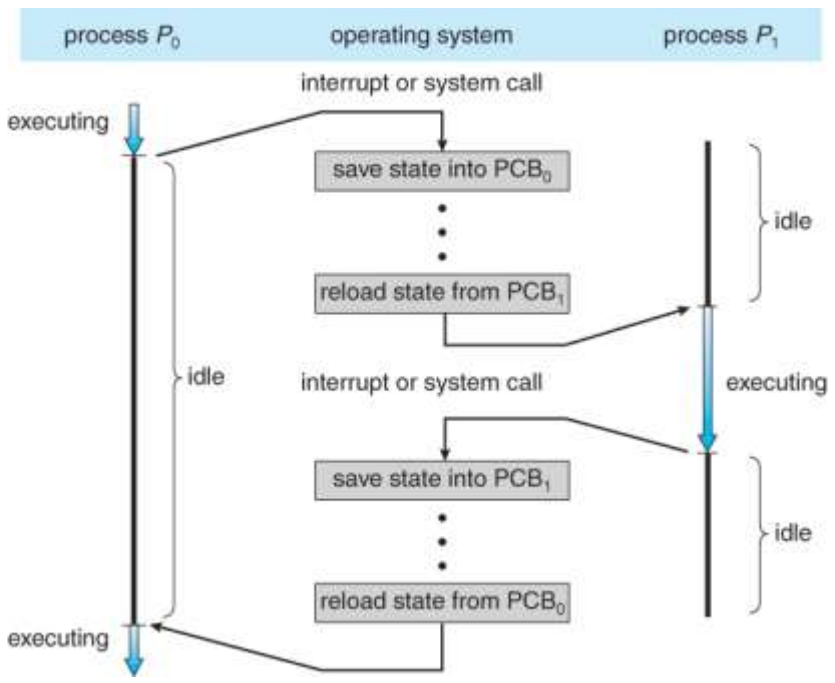


Fig: CPU switching from process to process

The changeover form one process to the next is called a context switch.
During a context switch, the processor obviously cannot perform any useful computation and because of the frequency with which context switches occur, operating systems must minimize the context-switching time in order to reduce system overhead.
Suppose OS is managing two processes p1 and p2, currently, p1 is running and p2 is idle, their PCBs are stored somewhere in the memory.

- The process p1 is currently running means that CPU registers hold the values that correspond to the state of p1.
- After some time suppose OS decides to interrupt p1, so OS will update all the state information fields of the process p1 including the CPU registers to the PCB of p1.
- After this the OS will restore the PCB of p2 from the memory, i.e. OS will update all the CPU register from the PCB of p2 and will start executing process p2.

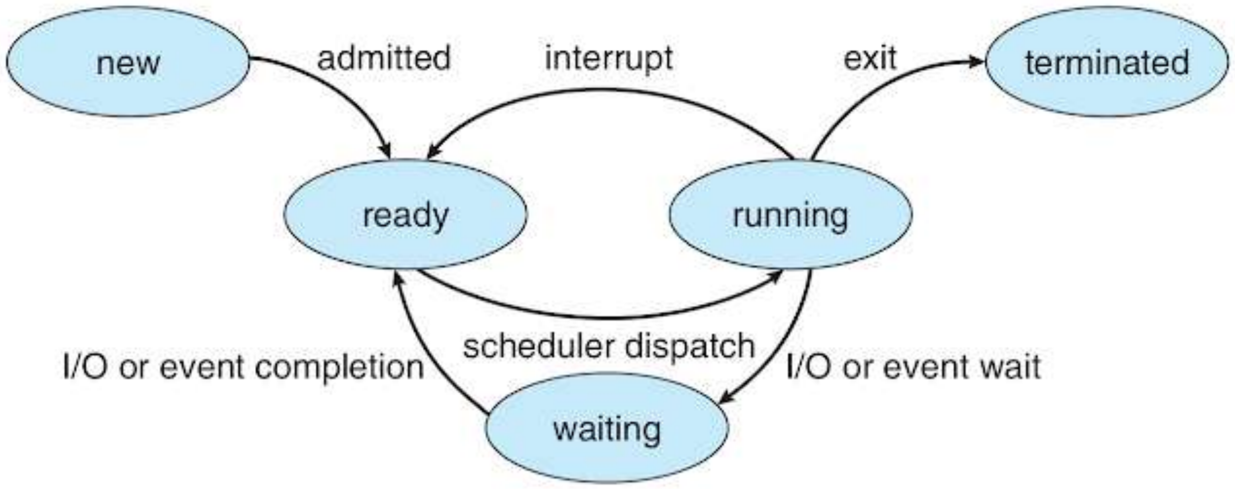
- After some time if process p2 is interrupted the PCB of p1 will be restored and CPU registers will be updated from the value of PCB of p1 and p1 will start executing from the exact same point where it was interrupted earlier by the operating system.

Each time this swapping is performed we call it **context switch**.

Context Switch:
It is a mechanism used by operating system to switch execution from the context of one process to the context of another process.
Context switching is an expensive operation because of the number of instructions involved in loading and restoring values of fields of PCB from the memory.

- Also when a process p1 is executing a lot of its data is stored in the *CPU cache* as accessing the cache is much-much faster as compared to accessing from the memory. When the data we want is present in the cache we say that ***cache is hot***.
- When CPU will switch the context from process p1 to process p2, the process p2 will replace process p1's cache with its own cache, so next time when the context will switch from the process p2 back to the process p1, p1 will not find its data in the cache and has to access it from the memory, so it will incur cache misses, so we call it ***cold cache.***

Process Lifecycle/ Process States:
As a process executes, it changes state. The state of the process refers to what the process currently does. A process can be in one of the following states during its lifetime:



- **New:** This is the initial state in which the process is about to be created but not yet created, it is the program which is present in secondary memory that will be picked up by OS to create the process.

- **Ready:** New → Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution.
    - Processes that are ready for execution by the CPU are maintained in a queue for ready processes. A process may come into this state from New state, from the running state or from the waiting state.

- **Running:** The CPU is working on this process's instructions. From this state a number of things can happen:
    - a running process can be interrupted and the context switch is performed and the running process will move back to the ready state.
    - another possibility is that the running process needs to perform some long operations like reading data from the disk, or waiting for some events, or taking input from the keyboard, at that time the process enters the waiting state, at that time the process enters the waiting state, when the event occurs or the I/O operation completes, the process will become ready again.
    - Finally, when the process completes all the operations in the program or when it encounters some error, it will return corresponding exit code and the process will be terminated.

- **Waiting:** The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur. e.g. the process may be waiting for keyboard input, disk access request, inter-process messages, or a child process to finish.

- **Terminated:** The process has completed.