Unit-2 Process Management & Deadlock

Lecture – 2

Process Scheduling

CPU Bound process

- A CPU-bound process is a process that spends the majority of its time performing computations rather than waiting for input/output (I/O) operations. These processes utilize the CPU extensively.

I/O bound process

- An I/O-bound process is a process that spends more time waiting for I/O operations (such as reading from disk or network) than performing computations. These processes frequently interact with I/O devices.

Degree of multiprogramming

- The degree of multiprogramming refers to the number of processes that are kept in memory at the same time.

Scheduling queues

- In a multiprogramming operating system, processes go through different stages. At each stage, they are placed into different scheduling queues based on their current state. These queues help the OS manage which process to run next and how to efficiently allocate CPU and other resources.
- Types of scheduling queues: **Job Queue**, **Ready Queue**, **Waiting Queue**, **Device Queue.**
- These queues are usually implemented internally as linked lists to allow fast and flexible process management.
- Job Queue
    1. Contains all the processes in the system (whether in memory or waiting to be admitted).
    2. It includes newly submitted processes waiting to be brought into the main memory.
- Ready Queue
    1. Contains all processes that are in memory and ready to execute but are waiting for CPU time.
    2. The scheduler picks processes from this queue and assigns them to the CPU.
    3. Only one process can use the CPU at a time; the rest stay in the queue.
- Waiting (or Blocked) Queue
    1. Contains processes that are waiting for some I/O operation to complete (e.g., disk read, keyboard input).
    2. These processes are not ready for execution until the I/O is done.
- Device Queues
    1. Each I/O device has its own queue.

2. When a process requests an I/O operation, it's placed in the corresponding device queue until the operation completes.

- A common representation of process scheduling is shown through queueing diagram in the figure-1

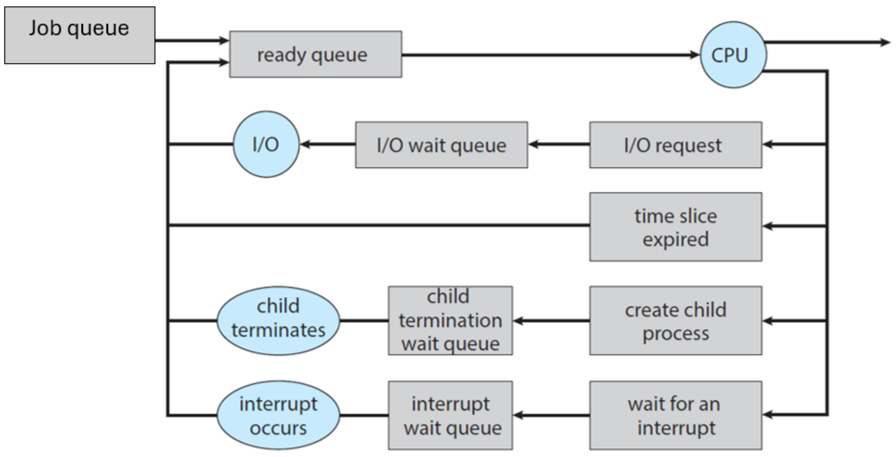

Figure-1 Queueing-diagram representation of process scheduling

- Two types of queues are present: the ready queue and the set of wait queues.
- The circles represent the resources that serve the queues, and the arrows indicate the flow of processes in the system.
- A new process is initially put in the ready queue from the job queue by the long term schedular. It waits there until it is selected for execution, or dispatched.
- Once the process is allocated a CPU core and is executing, one of several events could occur:
  1. The process could issue an I/O request and then be placed in an I/O wait queue.
  2. The process could create a new child process and then be placed in a wait queue while it awaits the child's termination.
  3. The process could be removed forcibly from the CPU, as a result of an interrupt or having its time slice expire, and be put back in the ready queue.
- In the first two cases, the process eventually switches from the waiting state to the ready state and is then put back in the ready queue.
- A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

Schedulers

- A process migrates between the ready queue and various waiting (I/O) queues throughout its lifetime. These migrations are managed by special components of the operating system known as schedulers.
- There are three types of process schedulers:
  - Long Term scheduler or Job scheduler
  - Short term scheduler
  - Medium term scheduler

- The role of these three schedulers are shown in the figure - 2 below.

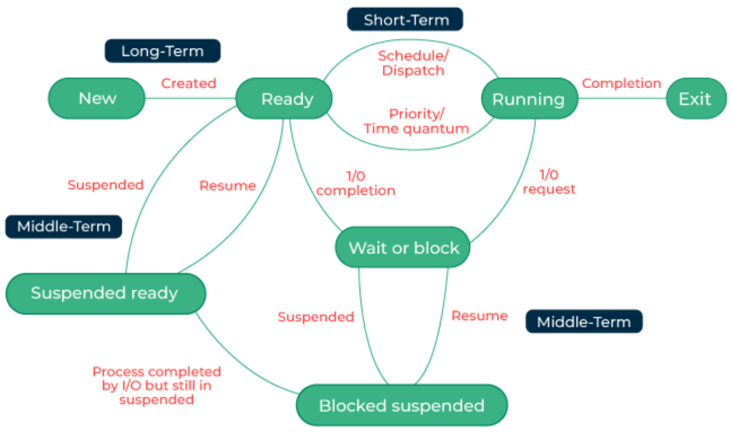

Figure –2 The role of long-term, medium-term, and short-term schedulers

- **Long term scheduler**
    - Long Term Scheduler loads a process from disk to main memory for execution.
    - It mainly moves processes from Job Queue to Ready Queue.
    - It controls the Degree of Multi-programming, i.e., the number of processes present in a ready state or in main memory at any point in time.
    - It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. The job scheduler increases efficiency by maintaining a balance between the two.
    - In some systems, the long-term scheduler might not even exist. For example, in time-sharing systems like Microsoft Windows
    - Slowest among the three (that is why called long term).
- **Short term scheduler**
    - CPU Scheduler is responsible for selecting one process from the ready state for running
    - STS (Short Term Scheduler) must select a new process for the CPU frequently to avoid starvation.
    - The CPU scheduler uses different scheduling algorithms to balance the allocation of CPU time.
    - Its main objective is to make the best use of CPU.
    - It mainly calls dispatcher.
    - Fastest among the three that is why called Short Term.
- **Medium term scheduler**
    - It is responsible for moving a process from memory to disk (or swapping).
    - It reduces the degree of multiprogramming (Number of processes present in main memory).
    - A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to

be swapped out or rolled out. Swapping may be necessary to improve the process mix (of CPU bound and IO bound)

- When needed, it brings process back into memory and pick up right where it left off.
- It is faster than long term and slower than short term.

Comparison among the schedulers

Long Term Scheduler	Short Term Schedular	Medium Term Scheduler
It is a job scheduler	It is a CPU scheduler	It is a process-swapping scheduler.
The slowest scheduler.	Speed is the fastest among all of them.	Speed lies in between both short and long-term schedulers.
It controls the degree of multiprogramming	It gives less control over how much multiprogramming is done.	It reduces the degree of multiprogramming.
It is barely present or nonexistent in the time-sharing system.	It is a minimal time-sharing system.	It is a component of systems for time sharing.
It can re-enter the process into memory, allowing for the continuation of execution.	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

- Switching the CPU from one process to another involves saving the current process's state and restoring the state of the next scheduled process. This operation is called a context switch.
- When a context switch occurs, the kernel saves the context (such as the CPU registers and program counter) of the currently running process into its Process Control Block (PCB), and then loads the saved context of the next process to run.
- Context switch time is considered pure overhead, as the CPU performs no productive computation during this switching—it merely prepares the CPU to resume a different process.
- The efficiency of context switching depends heavily on the hardware support available; systems with more efficient hardware can reduce context-switch latency significantly.
- The following diagram – 3 shows the context switch from process to process.

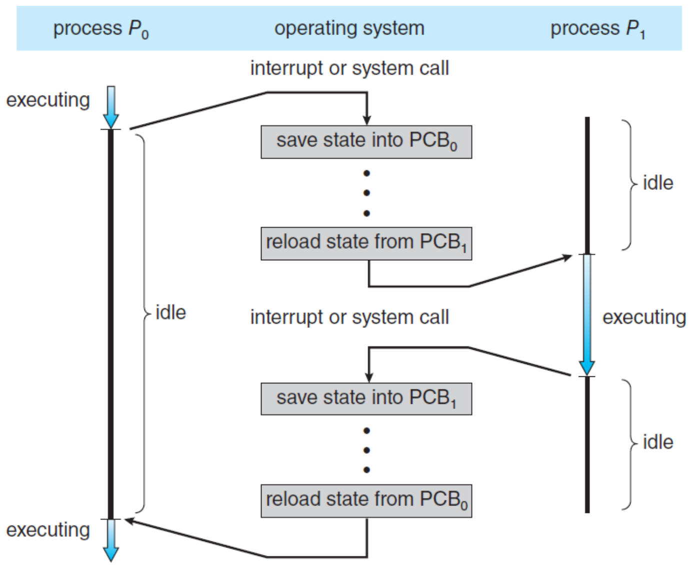

Figure – 3 Context switch from process to process

- As shown in the diagram, process P_0 is in the executing state, while process P_1 is in the idle state (not running).
    - When an interrupt or system call occurs, the operating system (OS) decides to perform a context switch from P_0 to P_1.
    - The current CPU state (registers, program counter, etc.) is saved into the PCB of P_0, allowing it to resume later from the exact same point.
    - The OS then loads the saved state of P_1 from its PCB and resumes execution from where P_1 had left off.
    - As a result, P_0 becomes idle, and P_1 starts executing.
- Later, another interrupt or system call may trigger a context switch back to P_0:
    - While P_1 is executing, an interrupt occurs, and the OS initiates another context switch.
    - The current CPU state is saved into the PCB of P_1.
    - The previously saved state of P_0 is restored into the CPU.
    - The CPU resumes executing P_0, and P_1 becomes idle again.

References

- Operating System Concepts: Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Ninth Edition, Wiley India.
- https://www.geeksforgeeks.org/operating-systems/process-schedulers-in-operating-system/