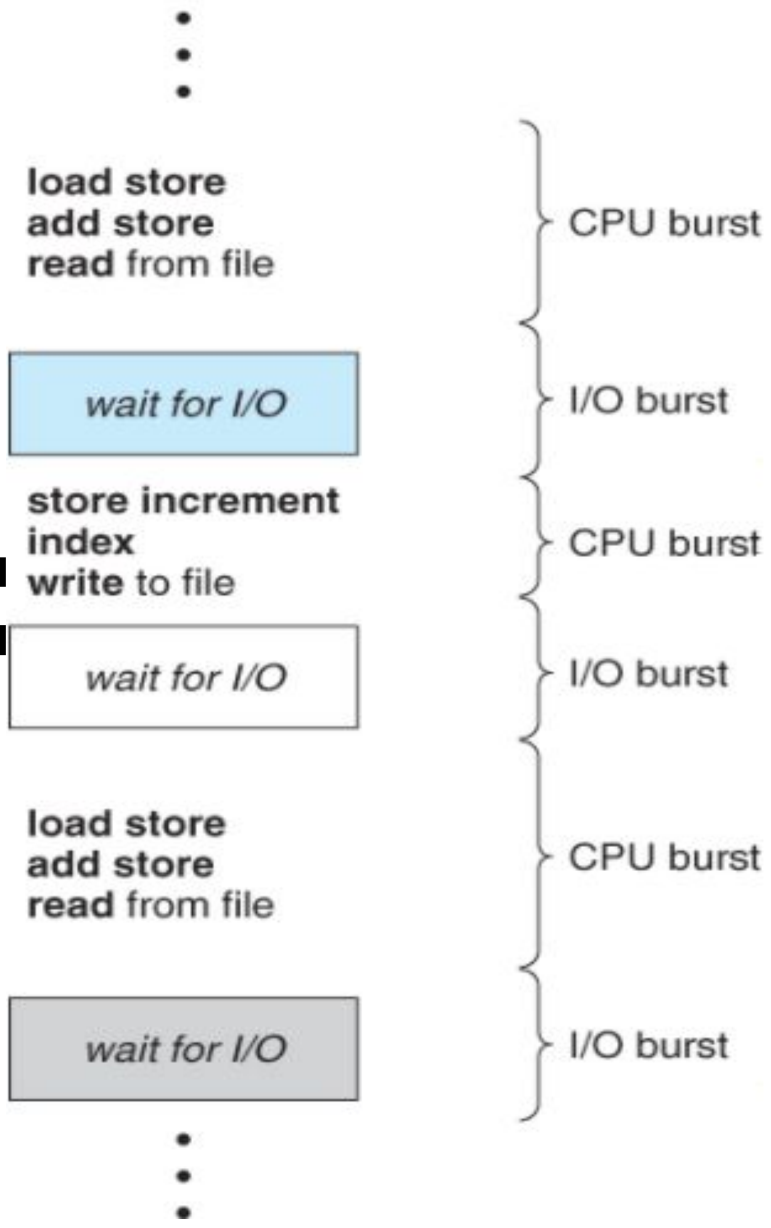
OPERATING SYSTEMS

Razi Uddin

Lecture # 11

LIFE OF A PROCE



CPU SCHEDULER

- Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed.
- The short-term scheduler (i.e., the CPU scheduler) selects a process to give it the CPU.
- It selects from among the processes in memory that are ready to execute and invokes the dispatcher to have the CPU allocated to the selected process.
- A ready queue can be implemented as a FIFO queue, a tree, or simply an unordered linked list.
- The records (nodes) in the ready queue are generally the process control blocks (PCBs) of processes.

DISPATCHER

- The dispatcher is a kernel module that takes control of the CPU from the current process and gives it to the process selected by the short-term scheduler.
- This function involves:

✔ Switching the context (i.e., saving the context of the current process and restoring the context of the newly selected process)

✔ Switching to user mode

✔ Jumping to the proper location in the user program to restart that program

- The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency

PREEMPTIVE AND NON-PREEMPTIVE SCHEDULING

- CPU scheduling can take place under the following circumstances:

1. When a process switches from the running state to the waiting state (for example, an I/O request) (**Non-Preemptive**)
2. When a process switches from the running state to the ready state (for example when an interrupt occurs). (**Preemptive**)
3. When a process switches from the waiting state to the ready state (for example, completion of I/O). (**Preemptive**)
4. When a process terminates. (**Non-Preemptive**)

Preemptive scheduling incurs a cost

SCHEDULING CRITERIA

- **CPU utilization**—We want to keep CPU as busy as possible. In a real system it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily used system)
- **Throughput**—If CPU is busy executing processes then work is being done. One measure of work is the number of processes completed per time, called, throughput. We want to maximize the throughput.
- **Turnaround time**—The interval from the time of submission to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O. We want to minimize the turnaround time.

SCHEDULING CRITERIA

- **Waiting time—**Waiting time is the time spent waiting in the ready queue. We want to minimize the waiting time to increase CPU efficiency.

- **Response time—**It is the time from the submission of a request until the first response is produced. Thus response time is the amount of time it takes to start responding but not the time it takes to output that response. Response time should be minimized.

SCHEDULING ALGORITHMS

- First-Come-First-Served (FCFS) Scheduling
- Shorted Job First (SJF) Scheduling
- Shortest Remaining Time First (SRTF) Scheduling
- Priority Scheduling
- Round-Robin Scheduling
- Multilevel Queues Scheduling
- Multilevel Feedback Queues Scheduling
- UNIX System V Scheduling

FIRST-COME, FIRST-SERVED (FCFS) SCHEDULING

- The process that requests the CPU first (i.e., enters the ready queue first) is allocated the CPU first.
- The implementation of an FCFS policy is managed with a FIFO queue.
- When a process enters the ready queue, its PCB is linked onto the tail of the queue.
- When CPU is free, it is allocated to the process at the head of the queue.
- The running process is removed from the queue.
- The average waiting time under FCFS policy is not minimal and may vary substantially if the process CPU-burst times vary greatly.
- FCFS is a non-preemptive scheduling algorithm.

FIRST-COME, FIRST-SERVED (FCFS) SCHEDULING

- When FCFS scheduling algorithm is used, the convoy effect occurs when short processes wait behind a long process to use the CPU and enter the ready queue in a convoy after completing their I/O.

- This results in lower CPU and device utilization than might be possible if shorter processes were allowed to go first.

SHORTEST-JOB-FIRST SCHEDULING

- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.
- The real difficulty with the SJF algorithm is in knowing the length of the next CPU request.
- For long term scheduling in a batch system, we can use as the length the process time limit that a user specifies when he submits the job.
- For short-term CPU scheduling, there is no way to length of the next CPU burst.
- One approach is to try to approximate SJF scheduling, by assuming that the next CPU burst will be similar in length to the previous ones.

SHORTEST-JOB-FIRST SCHEDULING

- The SJF algorithm may either be preemptive or non-preemptive.
- The choice arises when a new process arrives at the ready queue while a previous process is executing.
- The new process may have a shorter next CPU burst than what is left of the currently executing process.
- A preemptive SJF algorithm preempts the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst.
- Preemptive SJF scheduling is sometimes called shortest remaining-time-first scheduling.