BITS Pilani
Pilani Campus

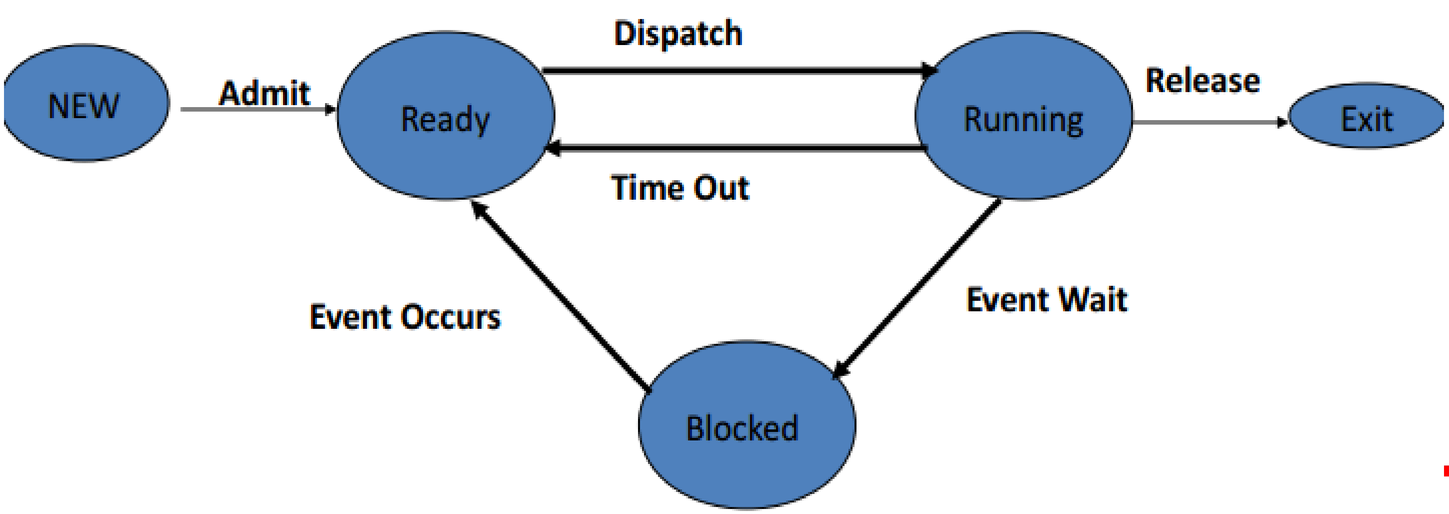
Computer Organization and Software Systems

Session 10

Prof. C R Sarma
WILP.BITS-Pilani

CPU Scheduler

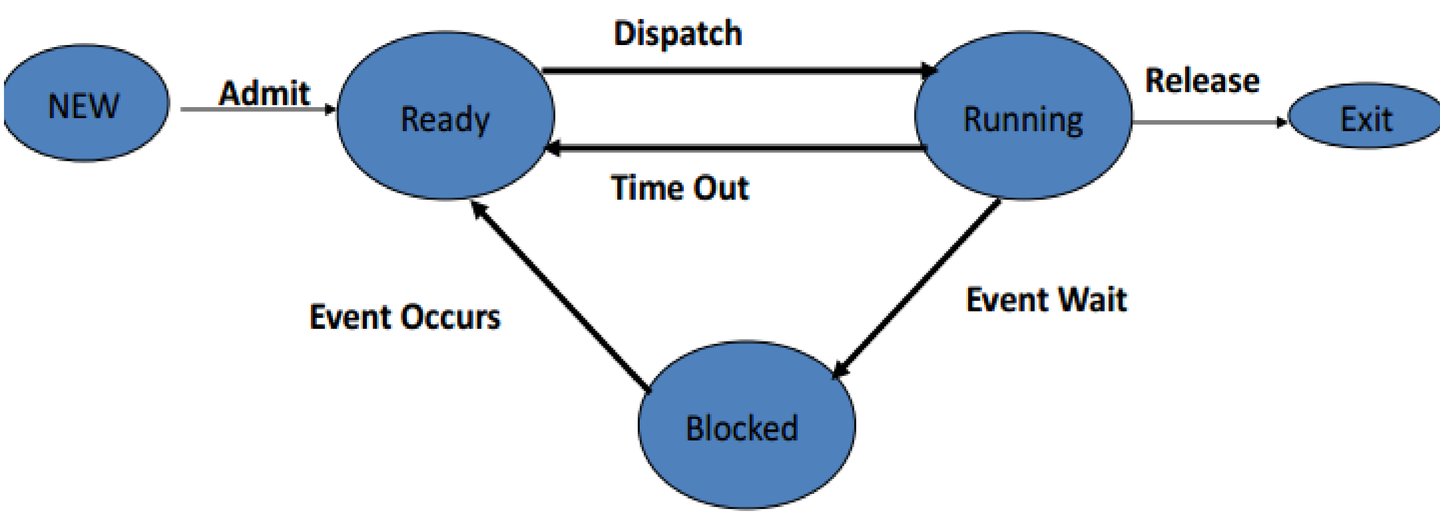
- CPU scheduling decisions may take place when a process:
  - 1. Switches from running to waiting(blocked) state
  - 2. Switches from running to ready state
  - 3. Switches from waiting (blocked) to ready state
  - 4. Switches from running to terminate state



CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  - **1.Switches from running to waiting (blocked) state**
  - 2.Switches from running to ready state
  - 3.Switches from waiting (blocked) to ready
  - **4. Switches from running to terminate sta**

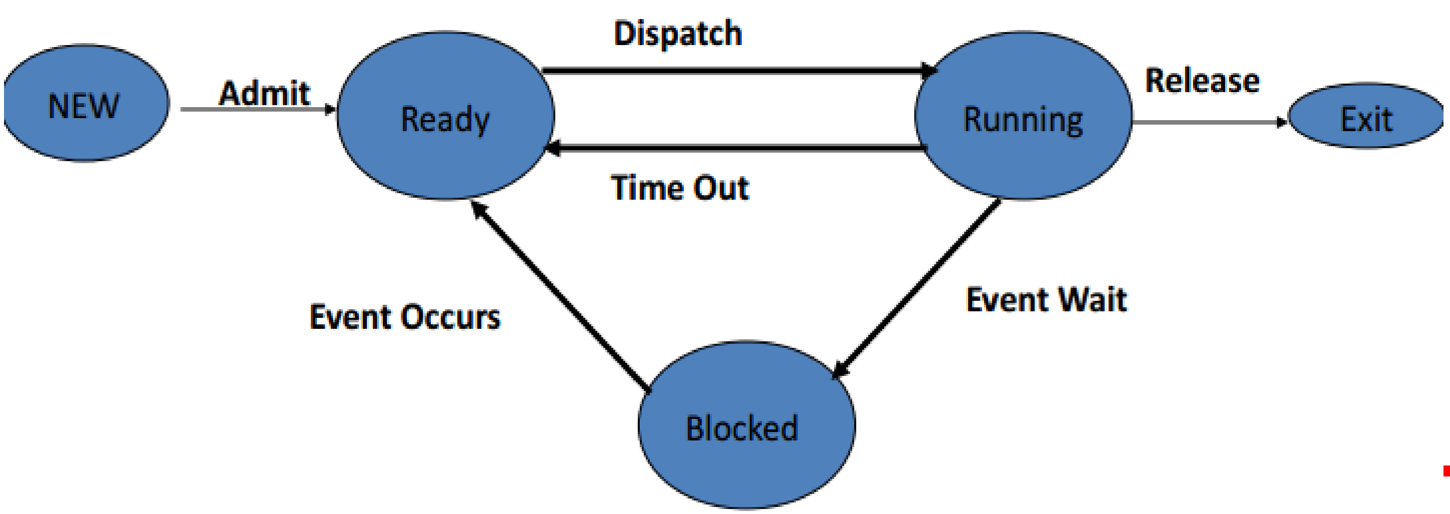
No choice! New process must be selected



CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  - 1.Switches from running to waiting state
  - 2.Switches from running to ready state
  - 3.Switches from waiting to ready
  - 4.Terminates

There is a choice!



Preemptive and Non-preemptive scheduling

- Non-preemptive scheduling
  - A new process is selected to run either
    - when a process terminates or
    - when an explicit system request causes a wait state
      - (e.g., I/O or wait for child)
- Preemptive scheduling
  - New process selected to run when
    - An interrupt occurs
    - When new processes become ready

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler;
- Function of Dispatcher :
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program

Dispatch latency – time it takes for the dispatcher to stop one process and start another running

Good Scheduling Algorithm

- Be fair : don't allow process to starve
- Be efficient : maximize CPU utilization
- Maximize throughput
  - Throughput : number of processes completed in unit time
- Minimize response time
  - Response time : time measured from process creation to the time of first output (response).
- Minimize waiting time
  - Waiting time : the amount of time a process has been waiting in the ready queue

Good Scheduling Algorithm

- Be predictable : a given job should take about the same amount of time to run when run multiple times
- Minimize overhead :Keep scheduling time and context switch time at a minimum
- Maximize resource use : favor processes that will use underutilized resources
- Avoid indefinite postponement : every process should get a chance to run eventually
- Enforce priorities
- Degrade gracefully : as the system becomes more heavily loaded, performance should deteriorate gradually, not abruptly

Performance Metrics

- CPU utilization – keep the CPU as busy as possible. CPU utilization vary from 0 to 100. It varies from 40 (lightly loaded) to 90 (heavily loaded)
- Throughput – Number of processes that complete their execution per time unit.
- Turnaround time – Amount of time to execute a particular process (interval from time of submission to time of completion of process).
- Response time
- Waiting time

Scheduling Algorithms

1. First Come First Served (FCFS)
2. Shortest Job First (SJF)
3. Shortest Remaining Time First
4. Priority Scheduling
5. Round Robin Scheduling
6. Multi level Feedback Queue

BITS Pilani
Pilani Campus

Problems

Scheduling Algorithm - FCFS

- Simplest scheduling algorithm.
- Non-preemptive type of scheduling.
- Process which requests the CPU first is allocated the CPU first.
- The implementation of FCFS is managed with a FIFO queue.

FCFS: Example 1

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	0	3				
P3	0	4				
P4	0	6				

FCFS: Example 2

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	8	3				
P3	3	4				
P4	5	6				

FCFS: Example 3

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	2				
P2	3	1				
P3	5	5				

FCFS Drawbacks

- A process that does not perform any I/O will monopolize the processor (Convoy Effect).
- Favors CPU-bound processes:
  - I/O-bound processes have to wait until CPU-bound process completes.
  - They may have to wait even when their I/O are completed (poor device utilization).

SJF: Shortest Job First

- The job with the shortest next CPU burst time is selected
- Associates with each process the length of its **next CPU burst**.
- Use these lengths to schedule the process with the shortest time.
- If two processes have the same **next CPU burst**, **FCFS** is used to break the tie.
- also known as "**shortest next CPU burst algorithm**"

SJF: Shortest Job First → Types

- Two schemes:
  - nonpreemptive – once CPU given to the process, it cannot be preempted until completes its CPU burst
  - preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal – gives minimum average waiting time for a given set of processes

SJF (non-preemptive): Example 4

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	0	3				
P3	0	4				
P4	0	6				

SJF (non-preemptive): Example 5

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	8	3				
P3	3	4				
P4	5	6				

SJF (Preemptive) / SRTF: Example 6

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	0	7				
P2	8	3				
P3	3	2				
P4	5	6				

SJF…

- Optimal: Shortest average wait time
- It's unfair !!
  - Continuous stream of short jobs will starve long job
- SJF scheduling is used frequently in long-term scheduling.
  - user needs to estimate the process time
- Short term Scheduling: Need to know the execution time of a process
  - May have an estimate, to predict next CPU burst
- Jobs are organized in order of estimated completion time

SJF Drawbacks

- Possibility of starvation for longer processes as long as there is a steady supply of shorter processes.

- CPU bound process gets lower priority but a process doing no I/O could still monopolize the CPU if it is the first one to enter the system.

Note: SJF implicitly incorporates priorities: shortest jobs are given preferences.

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - nonpreemptive
- SJF is a special case of priority scheduling where priority is the predicted next CPU burst time
- Problem ≡ Starvation – low priority processes may never execute
- Solution ≡ Aging – as time progresses increase the priority of the process

Note: Some use larger integer ≡ highest priority)

Priority (non-preemptive): Example 7

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	Pri	FT	TAT	WT	RT
P1	0	3	5				
P2	2	2	3				
P3	3	5	2				
P4	4	4	4				
P5	6	1	1				

Priority (preemptive): Example 8

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	Pri	FT	TAT	WT	RT
P1	0	3	5				
P2	2	2	3				
P3	3	5	2				
P4	4	4	4				
P5	6	1	1				

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then each process gets $1/n$ of the CPU time in chunks of at most q time units at once. No process waits more than $(n-1)q$ time units.
- Performance
  - q large $\Rightarrow$ FCFS
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

Round Robin: Example 10, Quantum = 3

- Draw Gantt chart
- Compute the average wait time, TAT and RT for processes

Process	AT	BT	FT	TAT	WT	RT
P1	5	5				
P2	4	6				
P3	3	7				
P4	1	9				
P5	2	2				
P6	6	3				

Example 11 : RR with Time Quantum = 20

<u>Process</u>	<u>Burst Time</u>
P_1	53
P_2	17
P_3	68
P_4	24

The Gantt chart is:

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

0 20 37 57 77 97 117 121 134 154 162

Typically, higher average turnaround than SJF, but better *response*