Chapter 5: **CPU Scheduling**

CPU Scheduling

Objectives

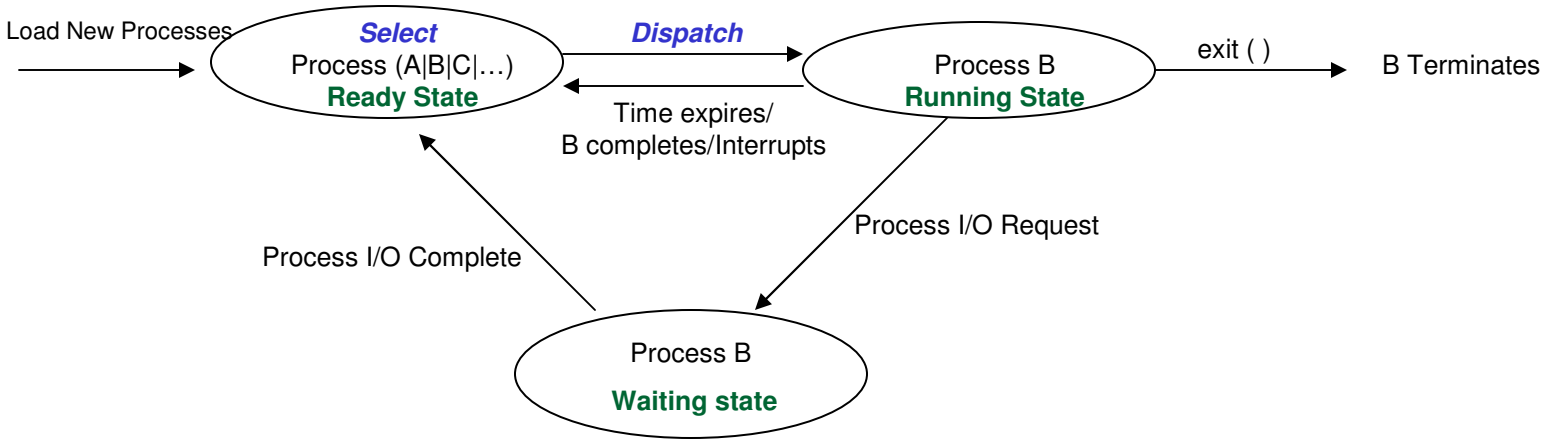
- Introduce CPU Scheduling as the basis of Multi-programmed OS
- Describe various CPU-Scheduling Algorithms
- Discuss Evaluation Criteria for Selecting CPU-scheduling Algorithm

CPU Scheduling
Basic Concept

- Single Processor System
  - Executes (runs) one process at a time
    - Other processes **wait** for the CPU to be free
  - What if the Execution process waits for completion of I/O request?
    - In a simple Computer system, CPU will be idle Unproductive
  - Multi-programming

  Keep several (non-running) processes in memory at the same time
    - When the Execution Process waits [for Completion of I/O Request]
    - OS ***schedules*** another process (thread?) on the CPU

CPU Scheduling
Overview



CPU Scheduling Functions

- CPU Scheduler
  - *Selects* a process from Ready Queue in memory for allocation to the CPU
- Dispatcher
  - Gives Control of the CPU to the Process Selected by CPU Scheduler
    - Context switch
    - Switching to user-level mode
    - Jump to the Instruction address to restart execution of program

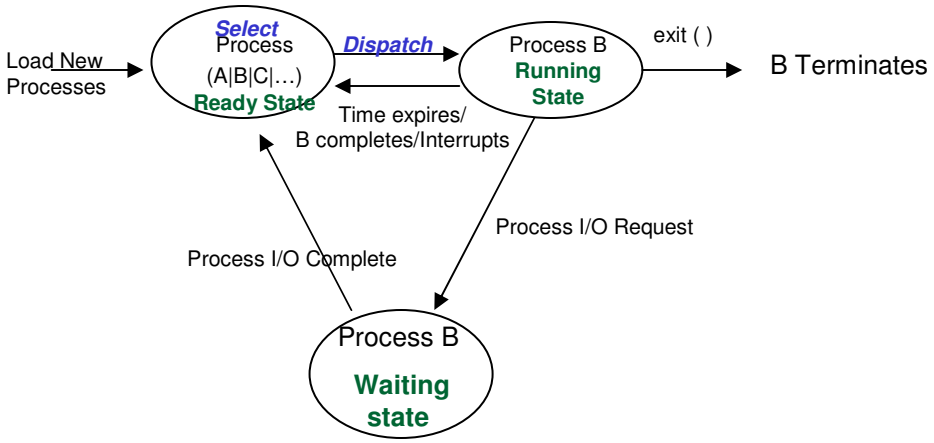
How does the Short-term Scheduler select a process from the Ready queue?

- We will talk about Scheduling Algorithms shortly

CPU Scheduling Functions

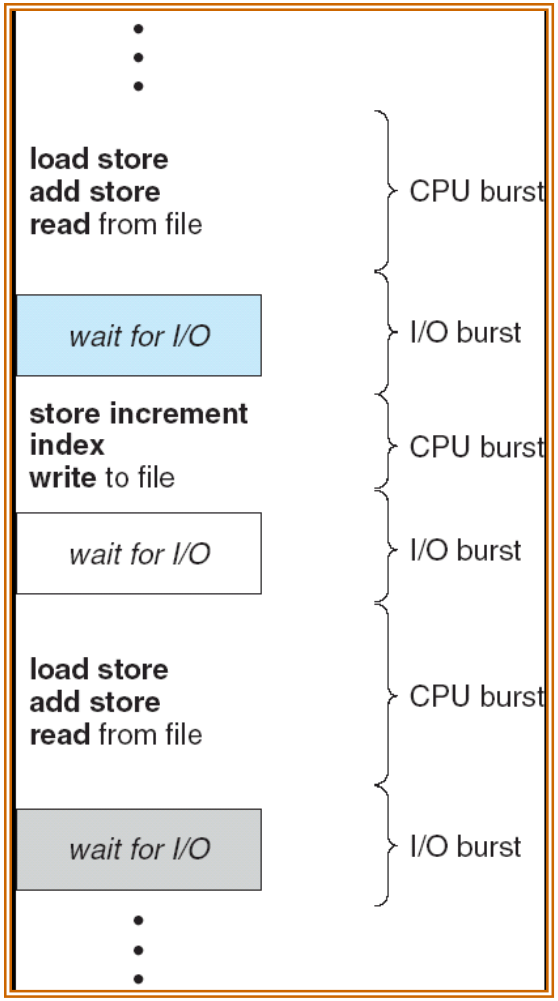
Preemptive Vs. Non-Preemptive

- Non-Preemptive Scheduling
  - Scheduler executes when Process Switches from Running state to Waiting State or
  - Process Terminates
- Preemptive Scheduling
  - Scheduler executes when Process switches from Running state to Ready state
  - Scheduler executes when Process switches form Waiting state to Ready state
  - Incurs (*synchronization*) cost associated with access to shared data



CPU Scheduling
Basic Assumption

- CPU-I/O Burst Cycle
  - Process Execution consists of a cycle dominated by CPU execution and I/O Wait
    - Execution starts with CPU burst, then
    - I/O burst, then
    - Another CPU burst and another I/O burst…
  - Processes alternate between these two states



What is the final CPU burst request?

When is Process CPU bound?

When is Process I/O bound?

CPU Scheduling

Criteria

How does OS select a process, in ready state, for allocation to CPU?

- High CPU utilization
  - Keep CPU as busy as possible
    - Lightly loaded ~40% usage
    - Heavily loaded ~ 90% usage
- Increased Throughput
  - Schedule many short duration processes per unit of time
- Minimize Turnaround Time
  - Keep elapsed time (waiting to get into memory, Ready state, Running State, Waiting State) short
- Minimize Waiting Time
  - Keep total time waiting in Ready queue to a minimum
- Minimize Response Time
  - For interactive systems, keep the elapsed time for submission of a request until the FIRST response to a minimum

CPU Scheduling Algorithms

First-Come, First-Served Scheduling (FCFS)

- Rule
  - Select process from Ready Queue in the order of its arrival (FIFO)
    - Run process until it terminates or blocks (non-preemptive)
  - Process that selects CPU first is allocated CPU first

- Implement using FIFO Queue
  - New process PCB is placed at tail end of queue
  - When CPU is available
    - Dispatcher assigns the Leading PCB in the queue
    - Running process is removed from queue

First-Come, First-Served Scheduling

Example

Assume Process arrive at t=0 ⟶

Process Arrival in Ready queue	P1	P2	P3
Burst Time (milliseconds)	24	3	3

FCFS schedules:

Waiting Time (msec): 0 24 27 30

What is the average waiting time?

(0 + 24 +27)/3
= 17 msec

First-Come, First-Served Scheduling

Example

Assume Process arrive at t=0 ⟶

Process Arrival in Ready queue	P2	P3	P1
Burst Time (milliseconds)	3	3	24

FCFS schedules:

Waiting Time (msec): 0 3 6 30

What is the average waiting time?

(0 + 3 +6)/3
= 3 msec

CPU Scheduling Algorithms

Shortest-Job-First Scheduling (SJF)

- Rule
  - Select process from Ready Queue that has shortest next CPU burst
    - Run process until termination or process blocks (non-preemptive)
  - Use FCFS if a tie exists for shortest next burst

- SJF Challenges
  - Difficult to determine next CPU burst
  - Approximate using heuristic data
    - Past performance and predictions of processes
    - For Batch jobs – use user specified time limit

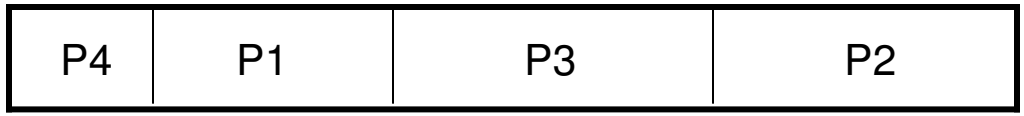
Shortest-Job-First Scheduling

Example

Assume Process arrive at t=0 ⟶

Process Arrival in Ready queue	P1	P2	P3	P4
Burst Time (milliseconds)	6	8	7	3

SJF schedules:



Waiting Time (msec): 0 3 9 16 24

What is the average waiting time?

(0 + 3 +9 + 16)/4

= 7 msec

CPU Scheduling Algorithms
Priority Scheduling

- Rule
  - Each process in Ready Queue is associated with a priority level
  - Select Process with highest priority level
    - Priority levels indicated by range of numbers
      - Order of magnitude is system dependent
  - Can be run either as preemptive or non-preemptive
    - Preemptive Mode
      - If a process arrives with higher order priority than running process
        - CPU is preempted
    - Non-Preemptive Scheduling
      - If a process arrives with higher order priority than running process
        - Put new process ahead of queue
  - Use FCFS if a processes have equal priority level

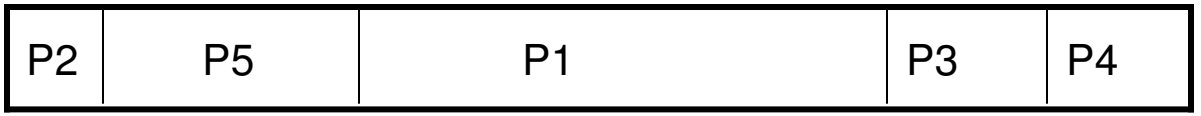
Priority Scheduling

Example

Assume Process arrive at t=0 ⟶

Process Arrival in Ready queue	P1	P2	P3	P4	P5
Burst Time (milliseconds)	10	1	2	1	5
Priority	3	1	4	5	2

Priority schedules:



Waiting Time (msec): 0 1 6 16 18 19

What is the average waiting time?

(0 + 1 +6 + 16 + 18)/5

= 8.2 msec

Priority Scheduling

Priority Definitions

How do you determine priority-levels?

Use Measurable Quantities

- **External Policy**
  - Based on criteria external to the OS
    - Politics, Importance of process (Students/Faculty/Staff)
    - Availability of Funds
- **Internal Policy**
  - Time limits, memory Requirements
  - Number of open files per process
  - Ratio of I/O burst to CPU burst

Priority Scheduling
Challenges

Indefinite Blocking for low-priority processes

- Assume a stream of high priority processes in the Ready State Queue:
  - Processes with low priority may be blocked for a long time (or indefinitely) from access to the CPU (Starvation) if the system continues to be heavily loaded

Possible solution to Starvation

- Aging Technique
  - Gradually Increase the priority-levels of low priority processes at fixed time intervals
    - Suppose Priority Range: Low (120) and High (0)
    - Increase Processes with priority levels in range 120 to 60 by 1 (subtract 1) every 10 minutes say

CPU Scheduling Algorithms
Round-Robin Scheduling

- Rule
  - Define a small time unit "*Time Slice" (or Time Quantum) in range 10 to 100msec*
  - Assume Ready Queue is configured as a Circular Queue
  - Scheduler associates each Process, in Ready Queue, with fixed Time Slice
  - Select process to run on CPU for at most one time slice:
    - If process does not complete, move to tail end of Ready queue
    - If process terminates or blocks
      - Select another process from the front of the queue and run for at most one time slice
    - Repeat selection and execution until all process are executed
  - Implement Queue using FIFO

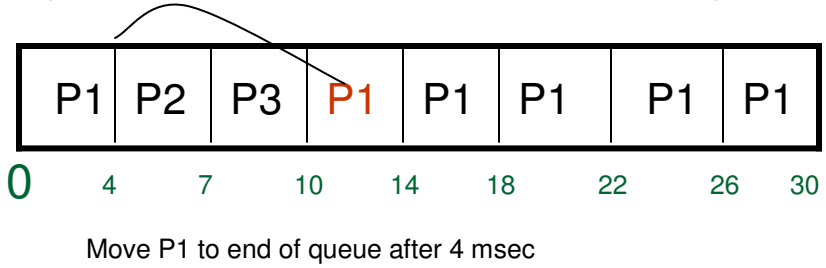
Round-Robin Scheduling

Example

Assume Process arrive at t=0 ⟶

Process Arrival in Ready queue	P1	P2	P3
Burst Time (milliseconds)	24	3	2

FCFS schedules (Assume a Time Slice of 4msec):



Move P1 to end of queue after 4 msec

Waiting Time (msec):

Average Waiting Time = (4 + 7 + 10 – 4))/3
= 5.66 msec

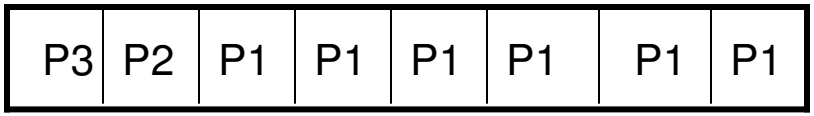
Round-Robin Scheduling

Example 2

Assume Process arrive at t=0 ⟶

Process Arrival in Ready queue	P3	P2	P1
Burst Time (milliseconds)	3	3	24

FCFS schedules (Assume a Time Slice of 4msec):



Waiting Time (msec): 0 3 6 10 14 18 22 26 30

Average Waiting Time = (3 +6))/3
= 3 msec

CPU Scheduling Algorithms

Multilevel Queue Scheduling

Assume Process can be easily classified into groups

Use Algorithmic Decomposition Principle

- Rule
  - Partition Ready Queue to Several Separate Queues
  - Assign Processes to individual Queues based on:
    - Process attributes: Process priority, Type, memory size
    - External Policies
  - Each Queue has its own Scheduling Algorithm
    - Use FCFS for batch processing (background queue)
      - Requires High throughput
    - use RR for Interactive processing (foreground queue)
      - Requires Minimum Response time
  - Associate scheduling Priority with each queue
    - Foreground Queue has priority 0 (Highest)
    - Background Queue has priority level 20 (low)
    - Use preemptive scheduling to go from priority level 0 to Priority Level 20 queue