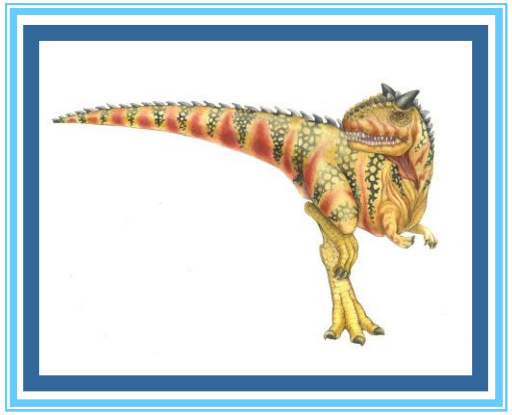
Chapter 5: CPU Scheduling



Chapter 5: CPU Scheduling

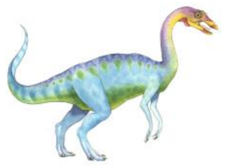
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Objectives

- Describe various CPU scheduling algorithms
- Assess CPU scheduling algorithms based on scheduling criteria

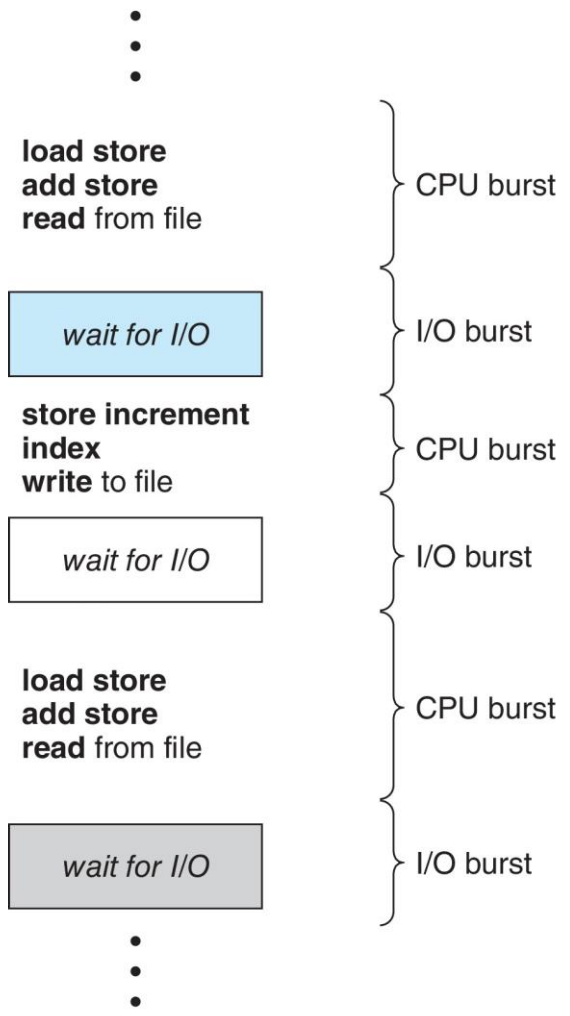
Basic Concepts

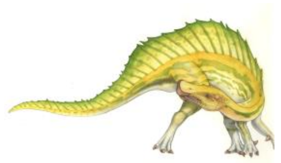
- Maximum CPU utilization obtained with multiprogramming
  - By switching among processes the OS makes the computer more productive
- The idea is simple: the process is executed until it must wait; for example waiting for I/O completion
  - In simple computer, if the CPU becomes idle until the I/O finishes then this is a waste of time
- With multiprogramming the goal is to utilize this time
  - OS keeps several processes in memory
  - When a process has to wait, the OS takes the CPU from that process and gives to another process



Basic Concepts

- Scheduling is a fundamental operating system function.
- Maximum CPU utilization obtained with multiprogramming.
- The algorithm, the scheduler uses is called scheduling algorithm.
- Several processes are kept in memory at one time.
- CPU–I/O Burst Cycle – Process execution consists of a **cycle** of CPU execution and I/O wait
  - Either CPU executing instruction
  - Or process is waiting for I/O
  - The process alternate between these two states
- **CPU burst** followed by **I/O burst**
- CPU burst distribution is of main concern

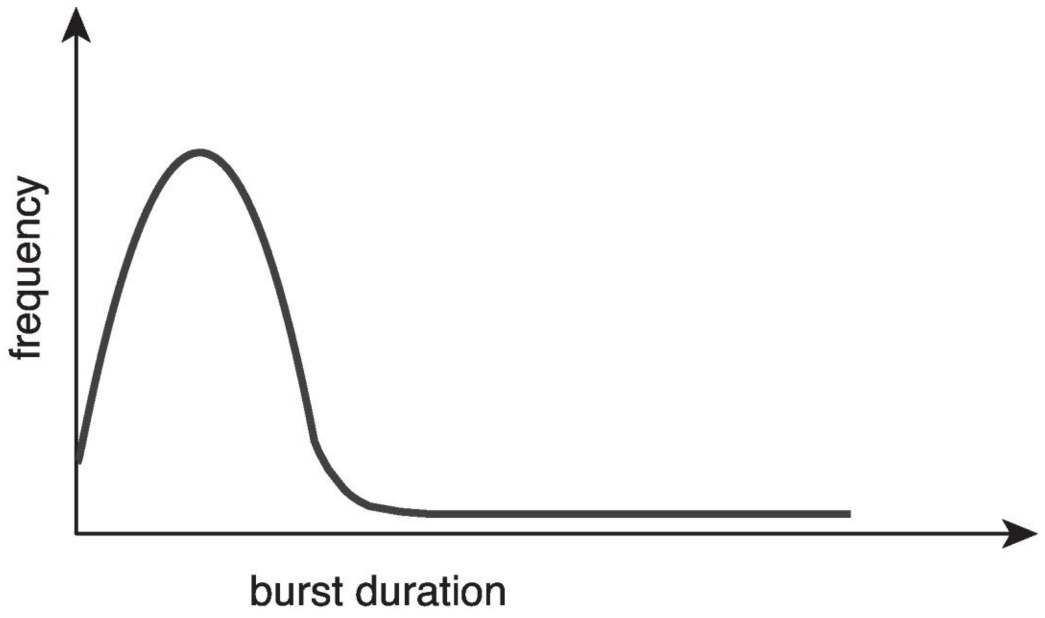




Histogram of CPU-burst Times

Large number of short bursts

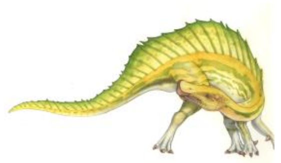
Small number of longer bursts



- I/O-bound program typically has many short CPU bursts
- CPU-bound program has few long CPU bursts

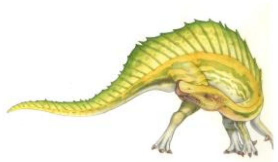
CPU Scheduler (with multiprograming)

- In a single processor system, only one process can run at a time. Others must wait until CPU is free and can be rescheduled.
- Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed (the records in the queues are generally PCBs of the pocesses).
- The short term scheduler (or CPU scheduler- a part of OS) selects a process to get the processor (CPU) among the processes which are already in memory. (i.e. from the ready queue).
- Processor scheduling algorithms try to answer the following crucial question.
  - Which process in the ready queue will get the processor?
- In order to answer this, one should consider the relative importance of several **performance criteria**.



CPU Scheduler

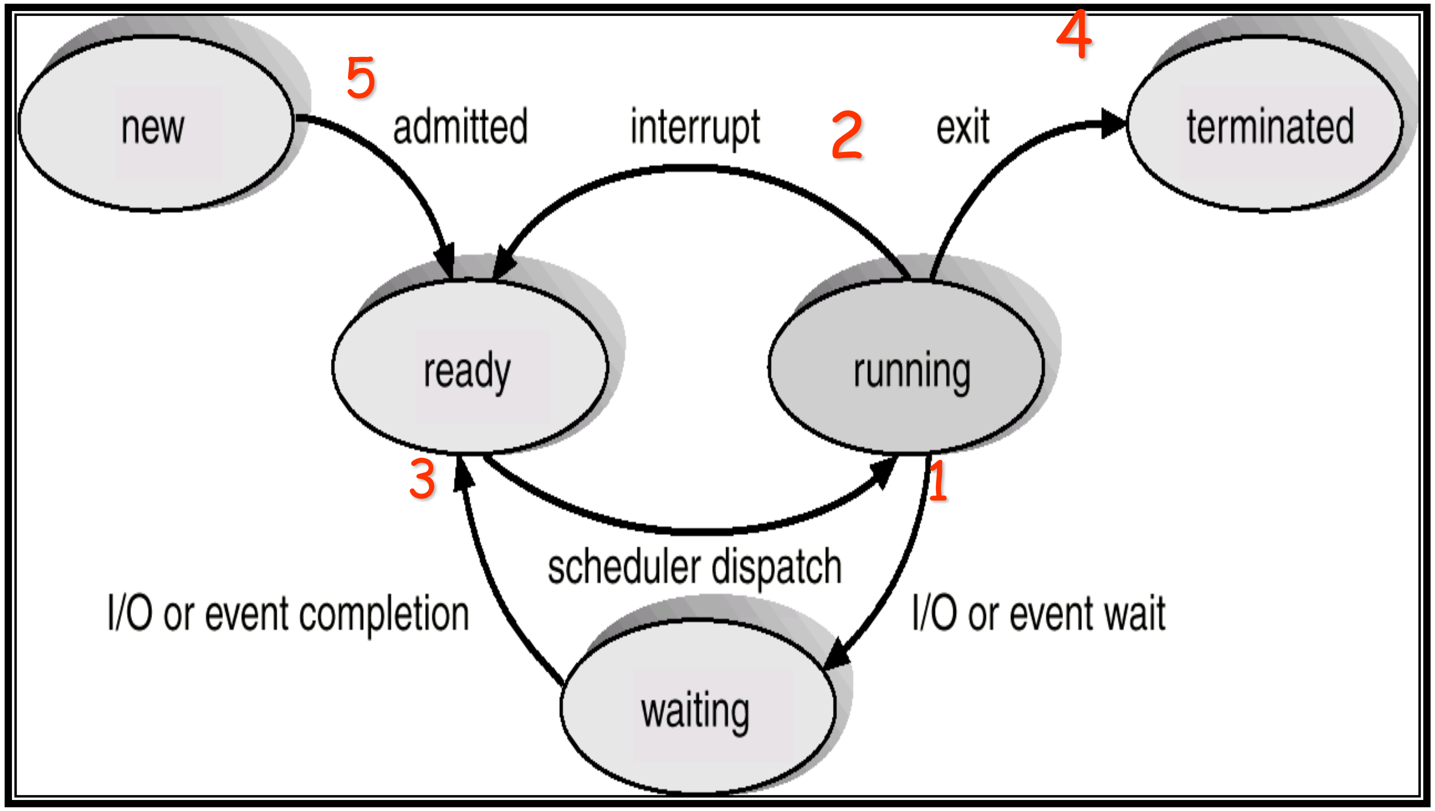
- The **CPU scheduler** selects from among the processes in ready queue, and allocates the a CPU core to one of them
    - Queue may be ordered in various ways; not necessarily FIFO (First In First Out)
    - But all processes in the ready queue are linked waiting for a chance to run on the CPU
    - Records in the queue are simply the PCBs of the processes



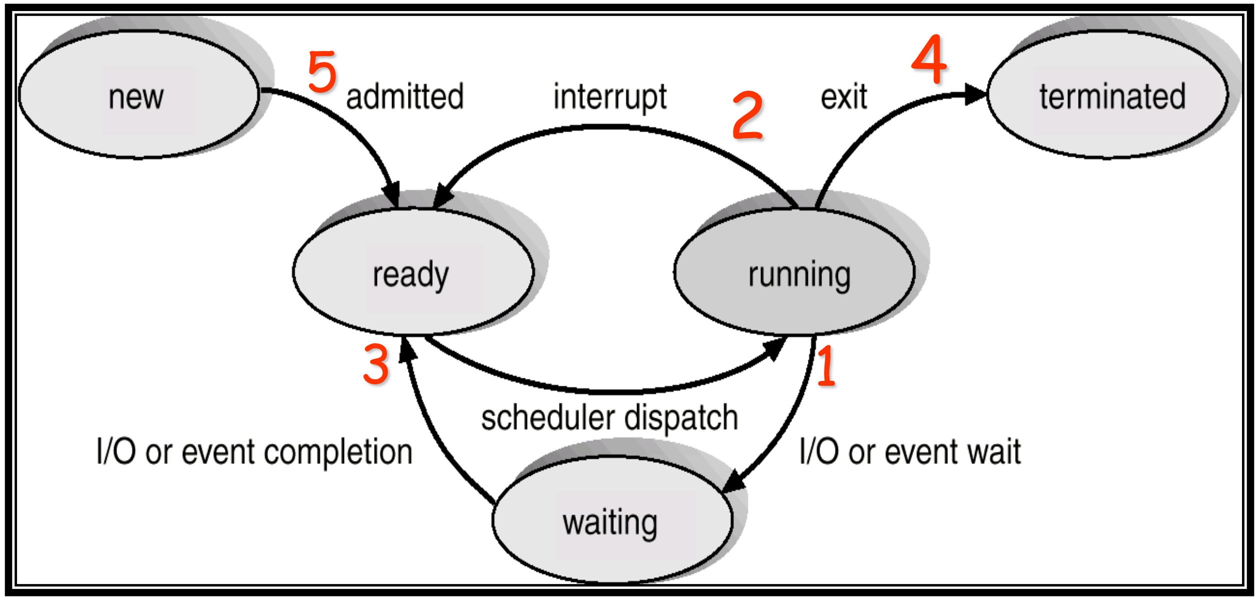
CPU Scheduler

- CPU scheduling decisions may take place when a process:
    1. Switches from running to waiting state
        1. As a result of I/O request or an invocation of wait() for termination of the child process
    2. Switches from running to ready state
        2. When an interrupt occurs
    3. Switches from waiting to ready
        3. For example. I/O completion
    4. Terminates
        4. When a process terminates
    5. Or a new process joins the ready queue

CPU Scheduler (with multiprograming)



Preemptive – Nonpreemptive Scheduling



Interrupt: Scheduler picks another process
Scheduler dispatch: Scheduler picks this process to execute

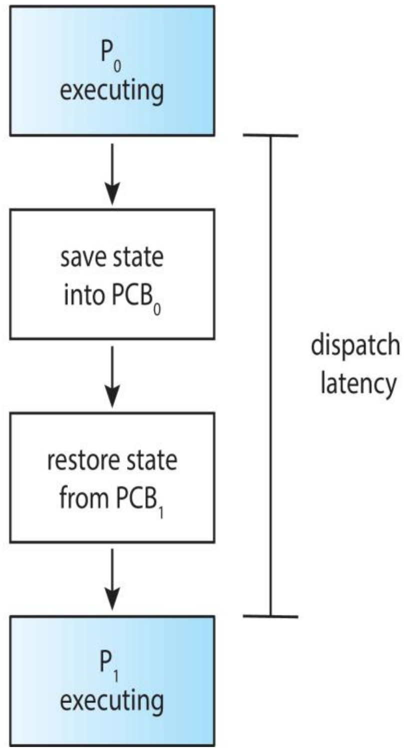
- In (1) and (4), a new process must be selected from the ready queue.
- In (2), (3) and (5), previously running process or a new process may be selected.
- Scheduling algorithms that act only in circumstances (1) and (4) are called **nonpreemptive(or noncooperative).** Once CPU has been allocated to a process, that process keeps the CPU until it releases the CPU (either by termination or by requesting I/O). This scheduling method was used by Microsoft Windows 3.x.
- Otherwise it is **preemptive.** Windows 95 and all subsequent versions of Windows operating systems have used preemptive scheduling.

Preemptive – Nonpreemptive Scheduling

- A scheduling algorithm which acts on all circumstances is called preemptive. (i.e. such an algorithm can select a new process in circumstances 2, 3 and 5).
- The CPU is allocated to the highest-priority process among all ready processes. The scheduler is called each time a process enters the ready state.
- Advantages of non-preemptive scheduling algorithms:
  - They cannot lead the system to a race condition.
  - They are simple.
- Disadvantage of non-preemptive scheduling algorithms:
  - They do not allow real multiprogramming.
- Advantage of preemptive scheduling algorithms:
  - They allow real multiprogramming.
- Disadvantages of preemptive scheduling algorithms:
  - They can lead the system to a race condition.

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- Dispatcher should be as fast as possible, since it is invoked during every context switch
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running



Scheduling Criteria

- **The criteria include the following**
  - **CPU utilization** – keep the CPU as busy as possible
    - top command can be used to obtain CPU utilization on linux and mac
  - **Throughput** – # of processes that complete their execution per time unit
  - **Turnaround time** – amount of time to execute a particular process
    - Interval of time from the time of submission of a process to the time of completion
  - **Waiting time** – amount of time a process has been waiting in the ready queue
  - **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Scheduling Criteria

- CPU utilization - keep the CPU as busy as possible. It is defined as:

  (processor busy time) / (processor busy time +processor idle time)
- Throughput – # of processes that complete their execution per time unit.

- Turnaround time – The interval from the time of submission to the time of completion of a process.

- Waiting time – amount of time a process has been waiting in the ready queue (waiting for I/O device is not counted)

- Response time – amount of time it takes from when a request was submitted until the first response is produced, **not** output.

Scheduling Criteria

- The previous mentioned criteria are used to judge and compare CPU-scheduling algorithms
- It is desirable to maximize CPU utilization and throughput, and to minimize turnaround time, waiting time, and response time
- In most cases, we optimize for the average measure
- However, under some circumstances, you might prefer to optimize the maximum or the minimum values rather than the average
  - For example to guarantee that all users get good service, we might want to minimize the maximum response time

Scheduling Algorithm Optimization Criteria

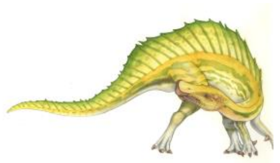
- Maximize CPU utilization
- Maximize throughput
- Minimize turnaround time
- Minimize waiting time
- Minimize response time

Scheduling Algorithms

- CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated to the CPU's core
- Next, we describe some of scheduling algorithms
  - Assuming that only one processing core is available

CPU Scheduling Algorithms

- First- Come, First-Served (FCFS) - Non preemptive
- Shortest-Job-First (SJF) - Non preemptive
- Shortest-Remaining-Time-First (SRTF) - Preemptive
- Priority Scheduling - Non preemptive, Preemptive
- Round Robin (RR) - Preemptive
- Priority – RR (Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling)



First-Come, First-Served (FCFS) Scheduling

- This is the simplest one. In this algorithm the set of ready processes is managed as FIFO (first-in-first-out) Queue. The processes are serviced by the CPU until completion in the order of their entering in the FIFO queue.

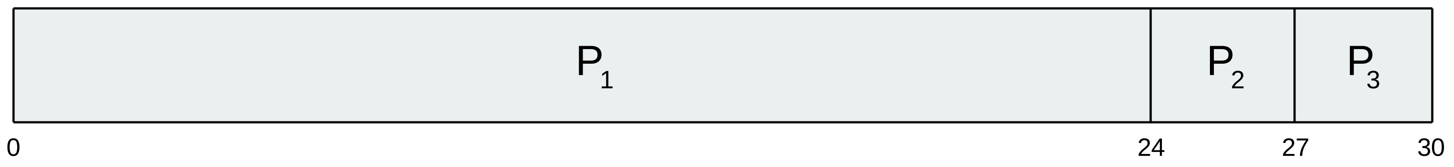
- Once allocated the CPU, each process keeps it until releasing either due to termination or requesting I/O.
- Average waiting time under FCFS policy is often quite long.

- FCFS scheduling algorithm is nonpreemptive.

First- Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart (a bar chart that illustrates a particular schedule, include the start and finish times of each participating process) for the schedule is:



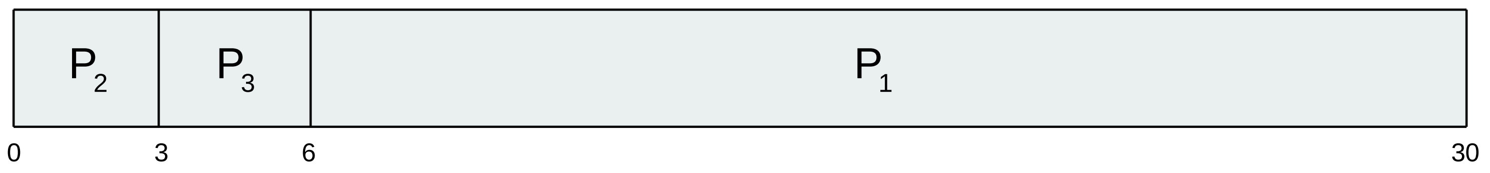
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17
- CPU utilization = (30/(30+0))x100 = 100 %

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- **Convoy effect** - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes
- This observation is used in the next algorithm