Unit III
Process Scheduling

Contents

- Uniprocessor Scheduling: Types of Scheduling: Preemptive, Non-preemptive, Long-term, Medium-term, Short-term scheduling
- Scheduling Algorithms: FCFS, SJF, RR, Priority

Uniprocessor (CPU) Scheduling

- **The problem:** Scheduling the usage of a single processor among all the existing processes in the system
- **The goal is to achieve:**
  - High processor utilization
  - High throughput
    - number of processes completed per unit time
  - Low response time
    - time elapse from the submission of a request to the beginning of the response

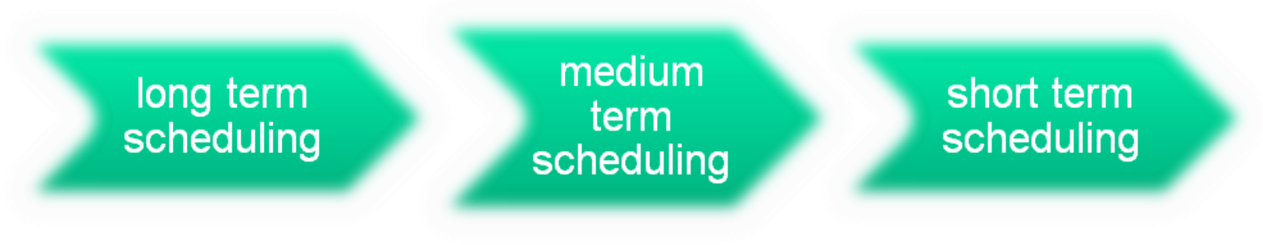
Scheduling Objectives

The scheduling function should

- Share time fairly among processes
- Prevent starvation of a process
- Use the processor efficiently
- Have low overhead
- Prioritise processes when necessary (e.g. real time deadlines)

Processor Scheduling

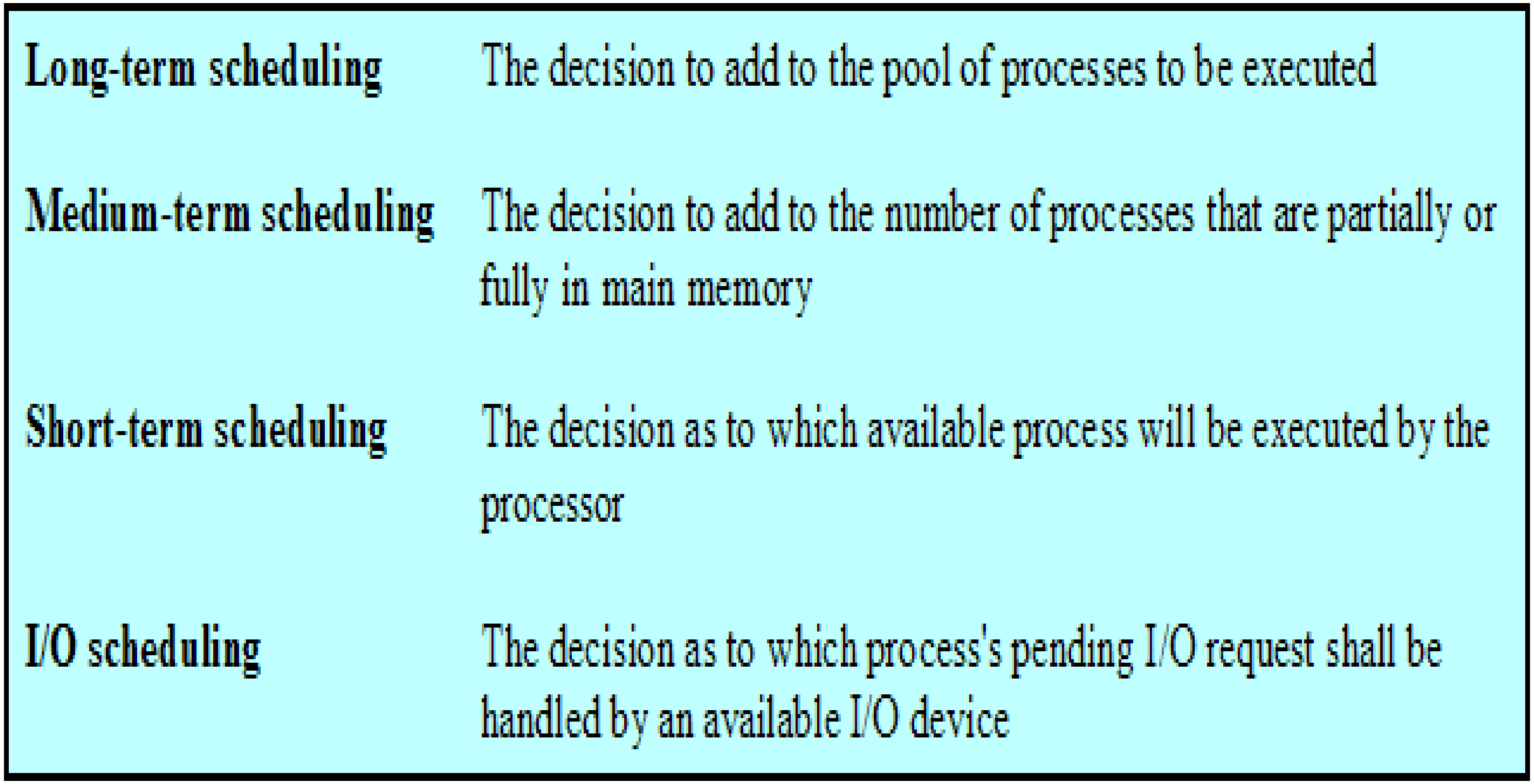
- Aim is to assign processes to be executed by the processor in a way that meets system objectives, such as response time, throughput, and processor efficiency
- Broken down into three separate functions:



Types of scheduling

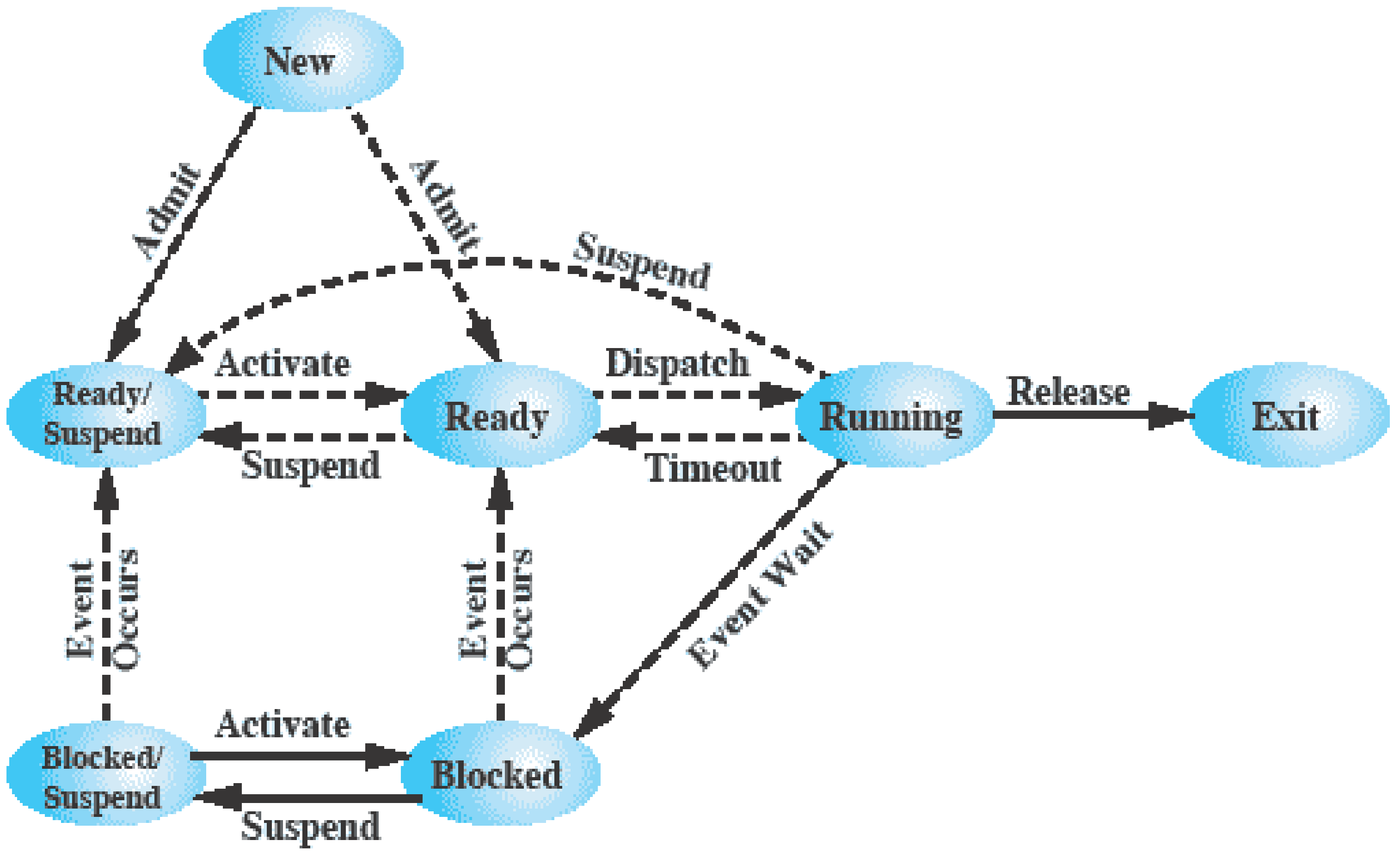
- In terms of in which phase is it applicable
- Long term scheduling
  - When a new process is created
  - Whether to add this to the set of processes who are currently active
- Medium term scheduling
  - Part of swapping function
  - Whether to add a process to those that are at least partially in main memory and therefore available for execution
- Short term scheduling
  - Actual decision as to which ready process to execute next

Types of Scheduling



Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device

Scheduling and Process State Transitions



(b) With Two Suspend States

Scheduling and Process State Transitions

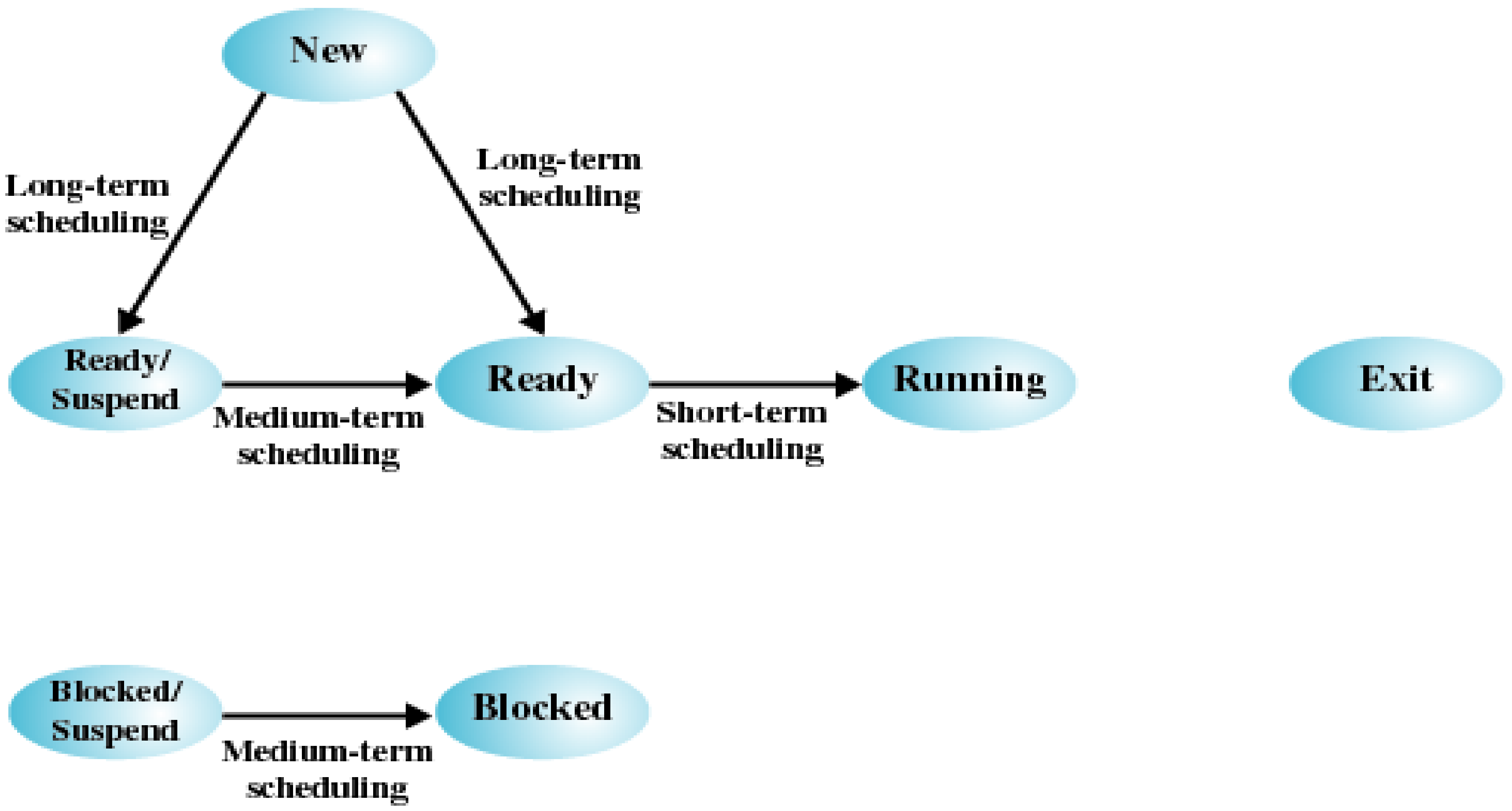
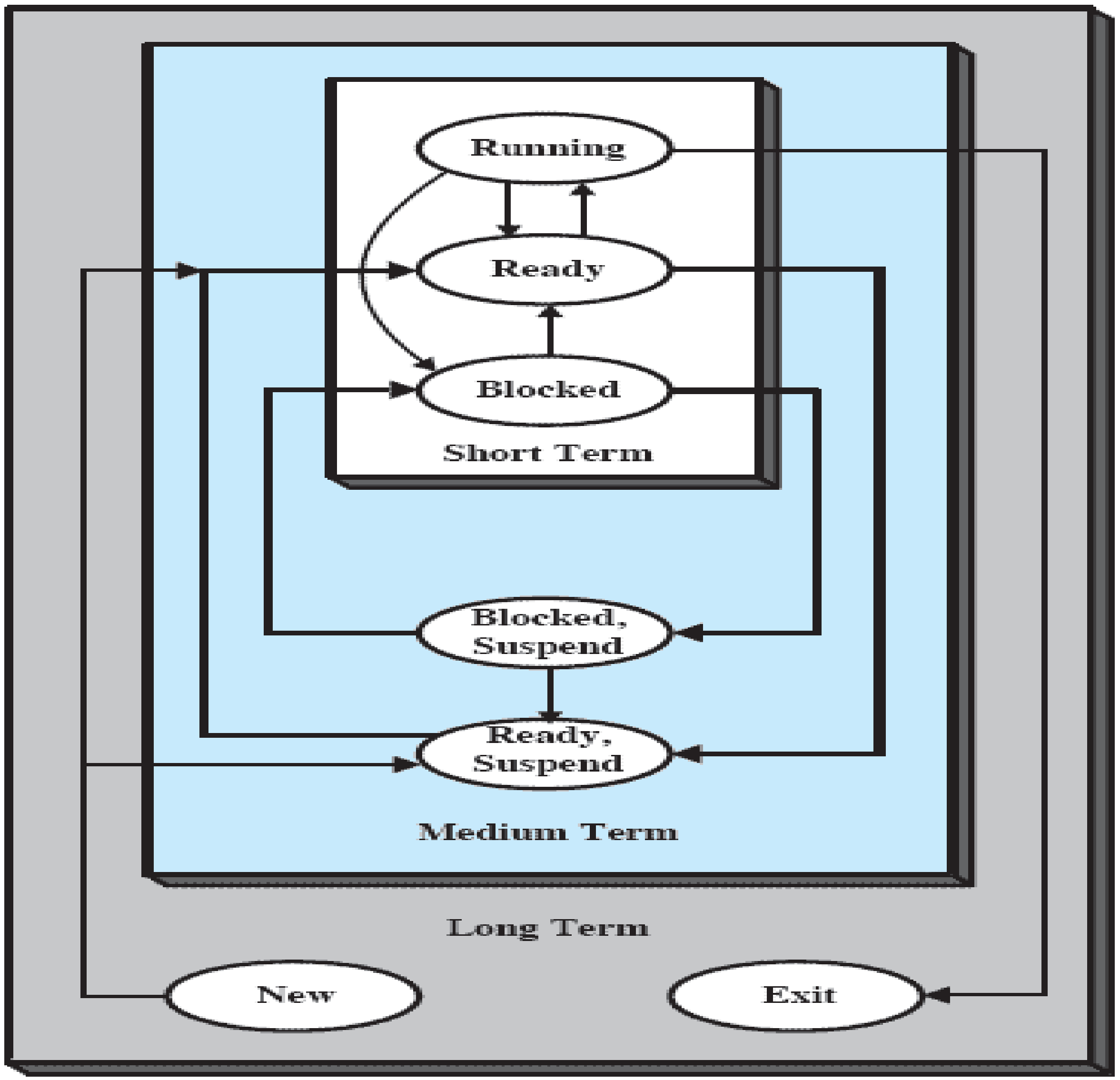


Figure 9.1 Scheduling and Process State Transitions

Nesting of Scheduling Functions



Another way to look at scheduling

- Scheduling determines which process will wait and which will progress
- Different queues are involved
- Scheduling is a matter of managing queues to minimize queuing delay and to optimize performance in a queuing environment

Queuing Diagram for Scheduling

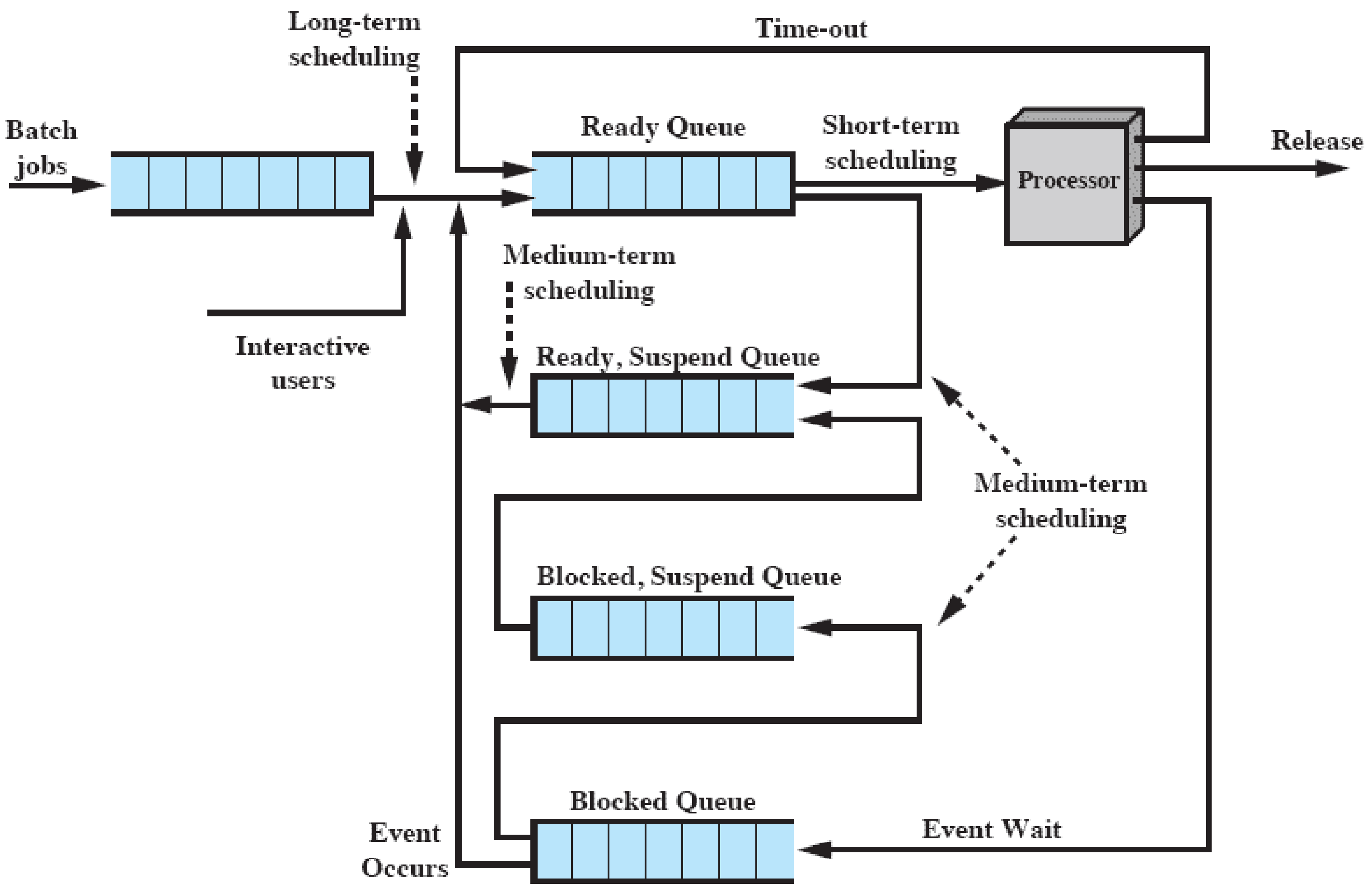


Figure 9.3 Queuing Diagram for Scheduling

Long-Term Scheduling: (Job Scheduler)

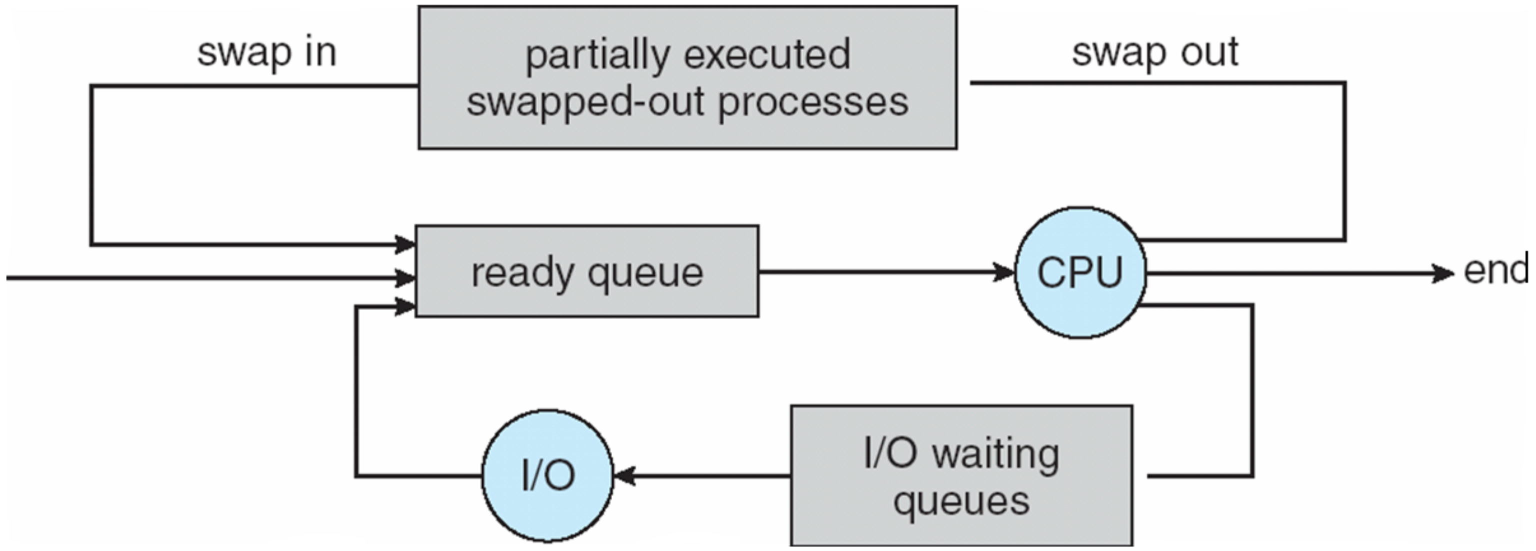
- Selects which processes should be brought into the ready queue
  - May be first-come-first-served
  - Or according to criteria such as priority, I/O requirements or expected execution time
- Controls the degree of multiprogramming
- If more processes are admitted
  - less likely that all processes will be blocked
  - better CPU usage
  - each process has less fraction of the CPU
- The long term scheduler will attempt to keep a mix of processor-bound and I/O-bound processes

Two kinds of processes

- Processes can be described as either:
  - **I/O-bound process –** spends more time doing I/O than computations, many short CPU bursts
    - Processes that are mostly waiting for the completion of input or output (I/O) are I/O Bound.
    - Interactive processes, such as office applications are mostly I/O bound the entire life of the process. Some processes may be I/O bound for only a few short periods of time.
    - The expected short run time of I/O bound processes means that they will not stay the running the process for very long.
    - They should be given high priority by the scheduler.
  - CPU-bound process – spends more time doing computations; few very long CPU bursts
    - CPU Bound processes are ones that are implementing algorithms with a large number of calculations.
    - They can be expected to hold the CPU for as long as the scheduler will allow.
    - Programs such as simulations may be CPU bound for most of the life of the process.
    - Users do not typically expect an immediate response from the computer when running CPU bound programs.
    - They should be given a lower priority by the scheduler.

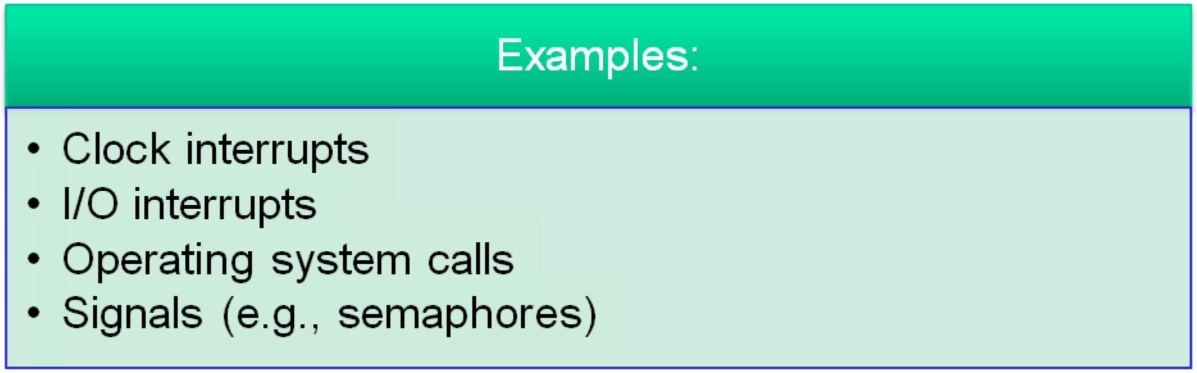
Medium-Term Scheduling (Swapper)

- Part of the swapping function
- Swapping decisions based on the need to manage multiprogramming
- Done by memory management software



Short-Term Scheduling: (CPU Scheduler)

- Selects which process should be executed next and allocates CPU
- The short term scheduler is known as the dispatcher
- Executes most frequently
- Is invoked on a event that may lead to choose another process for execution:



Examples:

- Clock interrupts
- I/O interrupts
- Operating system calls
- Signals (e.g., semaphores)

Short Term Scheduling Criteria

- Main objective is to allocate processor time to optimize certain aspects of system behaviour
- A set of criteria is needed to evaluate the scheduling policy

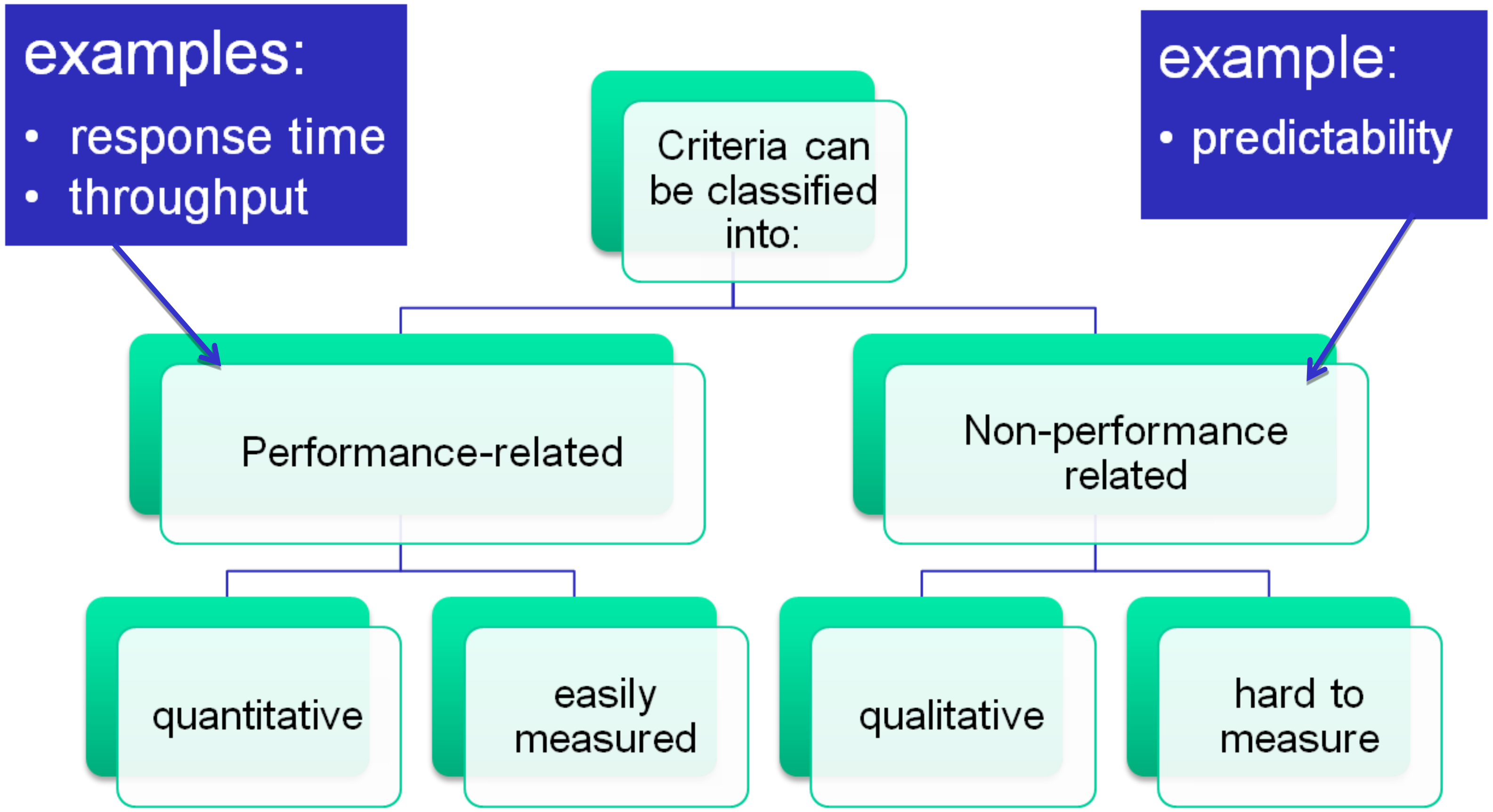
User-oriented criteria

- relate to the behavior of the system as perceived by the individual user or process (such as response time in an interactive system)
- important on virtually all systems

System-oriented criteria

- focus in on effective and efficient utilization of the processor (rate at which processes are completed)
- generally of minor importance on single-user systems

Short-Term Scheduling Criteria: Performance



Short-Tem Scheduling Criteria

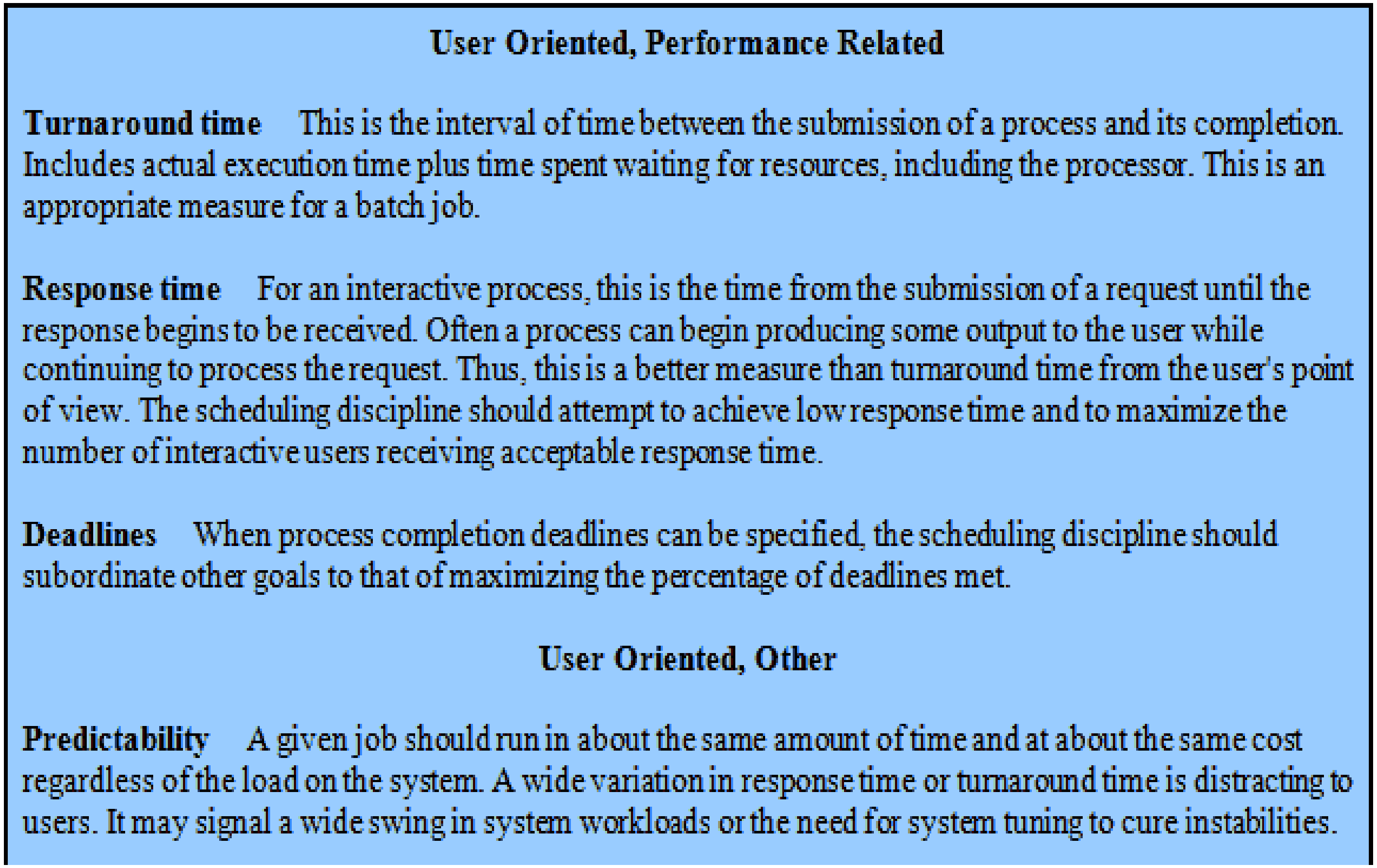
User-oriented: relate to the behaviour of the system as perceived by the individual user or process

- **Measure of user satisfaction**
- **Response time in case of interactive systems**
- **Turnaround Time**
- **Important on almost all systems**

System-oriented: focused on effective and efficient utilization of the processor.

- Measure of system performance
- Processor utilization: keep the CPU as busy as possible
- Fairness
- Throughput: number of process completed per unit time
- Waiting time – amount of time a process has been waiting in the ready queue
- Not that important on single user systems

Interdependent Scheduling Criteria



User Oriented, Performance Related

Turnaround time This is the interval of time between the submission of a process and its completion. Includes actual execution time plus time spent waiting for resources, including the processor. This is an appropriate measure for a batch job.

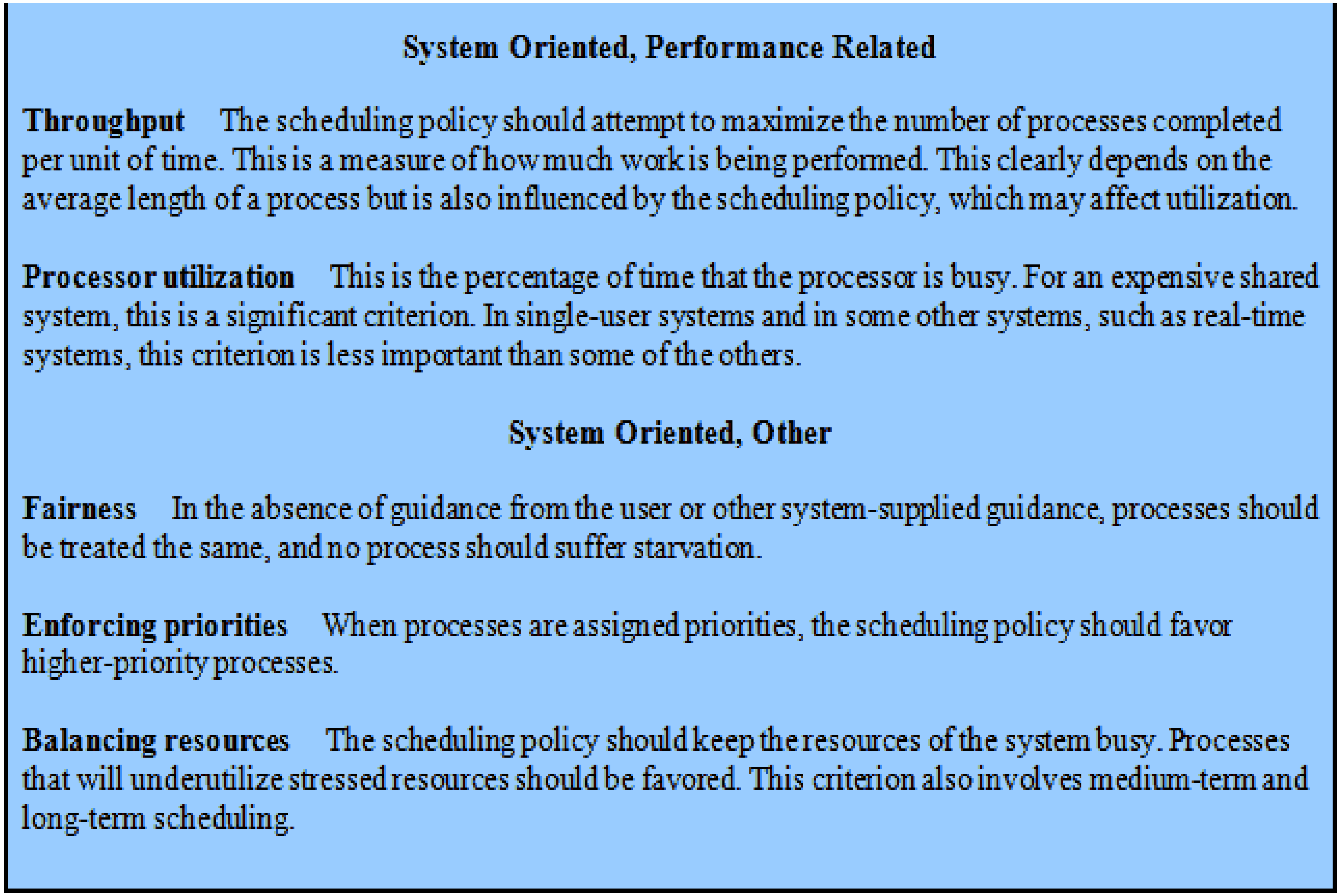
Response time For an interactive process, this is the time from the submission of a request until the response begins to be received. Often a process can begin producing some output to the user while continuing to process the request. Thus, this is a better measure than turnaround time from the user's point of view. The scheduling discipline should attempt to achieve low response time and to maximize the number of interactive users receiving acceptable response time.

Deadlines When process completion deadlines can be specified, the scheduling discipline should subordinate other goals to that of maximizing the percentage of deadlines met.

User Oriented, Other

Predictability A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system tuning to cure instabilities.

Interdependent Scheduling Criteria



System Oriented, Performance Related

Throughput The scheduling policy should attempt to maximize the number of processes completed per unit of time. This is a measure of how much work is being performed. This clearly depends on the average length of a process but is also influenced by the scheduling policy, which may affect utilization.

Processor utilization This is the percentage of time that the processor is busy. For an expensive shared system, this is a significant criterion. In single-user systems and in some other systems, such as real-time systems, this criterion is less important than some of the others.

System Oriented, Other

Fairness In the absence of guidance from the user or other system-supplied guidance, processes should be treated the same, and no process should suffer starvation.

Enforcing priorities When processes are assigned priorities, the scheduling policy should favor higher-priority processes.

Balancing resources The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Selection Function

- Determines which Ready process is dispatched next
- May be based on priority, resource requirements, or the execution characteristics of the process
- If based on execution characteristics, some factors to consider are
  - ***w*** = time spent in system so far, waiting
  - ***e*** = time spent in execution so far
  - ***s*** = total service time required by the process, including *e*; (estimated by system or user)

For example, the selection function max[*w] indicates an FCFS discipline.*

Decision Mode

- When/under what circumstances is the selection function is exercised?

- Two categories:
  - Nonpreemptive
  - Preemptive

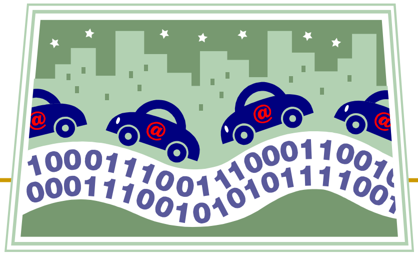
Nonpreemptive vs Preemptive

Nonpreemptive

- once a process is in the running state, it will continue until it terminates <u>or blocks itself for I/O</u>

Preemptive

- currently running process may be interrupted and moved to ready state by the OS
- preemption may occur when a new process arrives, on an interrupt, or periodically



Priorities

- Scheduler will always choose a process of higher priority over one of lower priority
- Have multiple ready queues to represent each level of priority
- Lower-priority may suffer starvation
  - Allow a process to change its priority based on its age or execution history

Processes have priorities

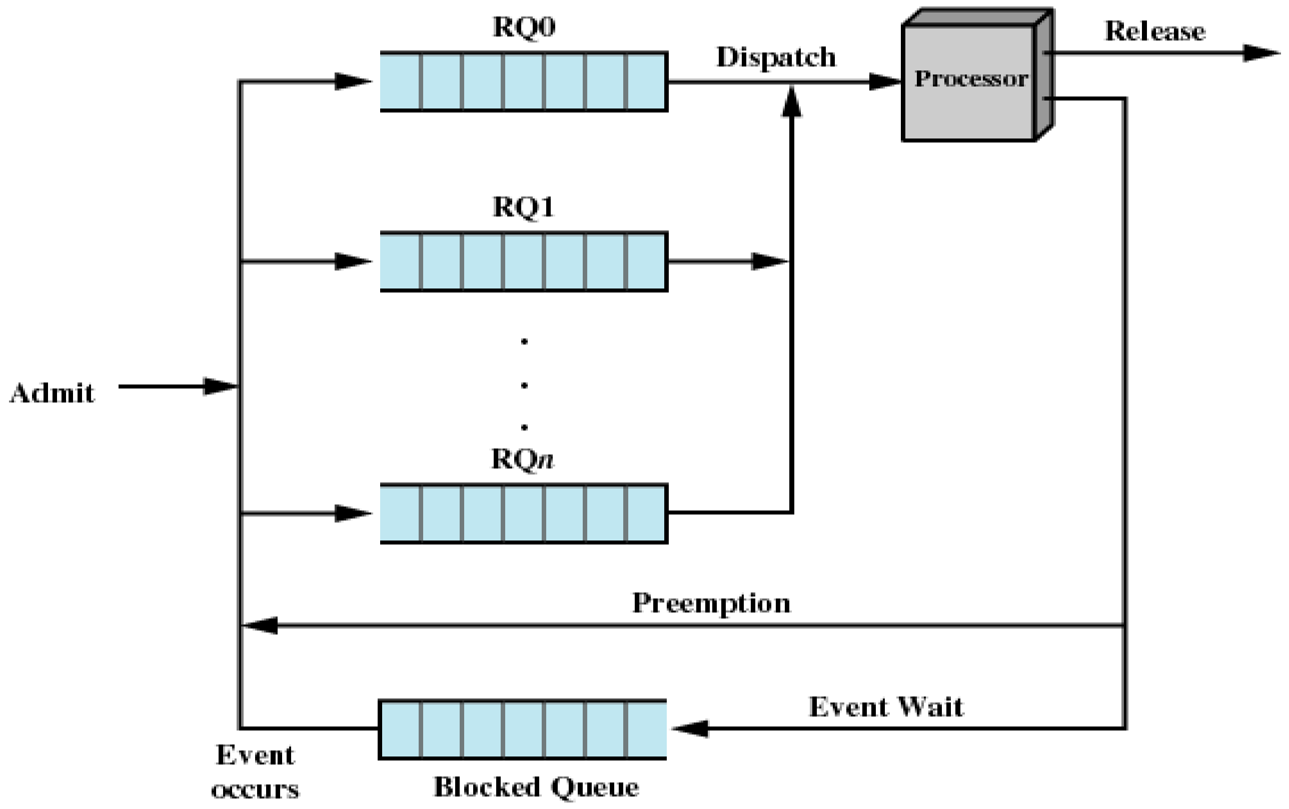


Figure 9.4 Priority Queuing

Contents

- Uniprocessor Scheduling: Types of Scheduling: Preemptive, Non-preemptive, Long-term, Medium-term,   Short-term scheduling
- Scheduling Algorithms: FCFS, SJF, RR, Priority

Process	Arrival Time	Burst Time	Priority	FINISH TIME	TURNAROUND TIME	WAITING TIME
P1	1	2	1			
P2	0	3	3			
P3	2	1	2			

Running example to discuss various scheduling policies

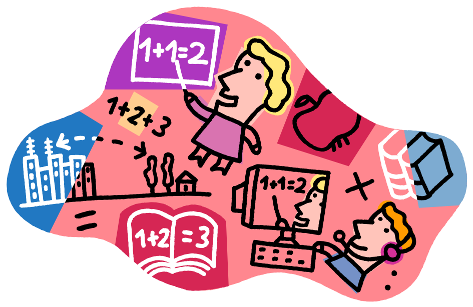
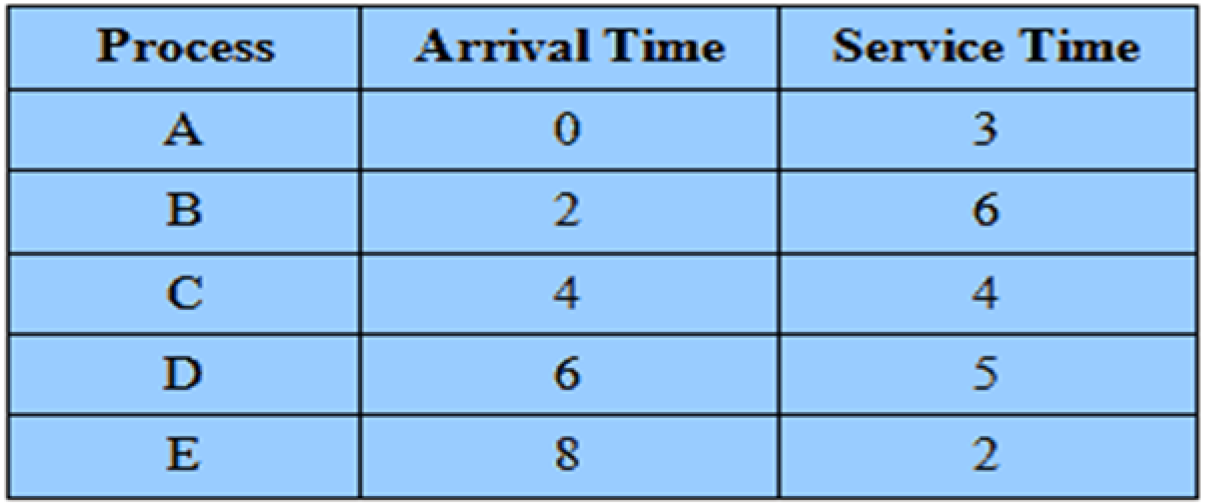
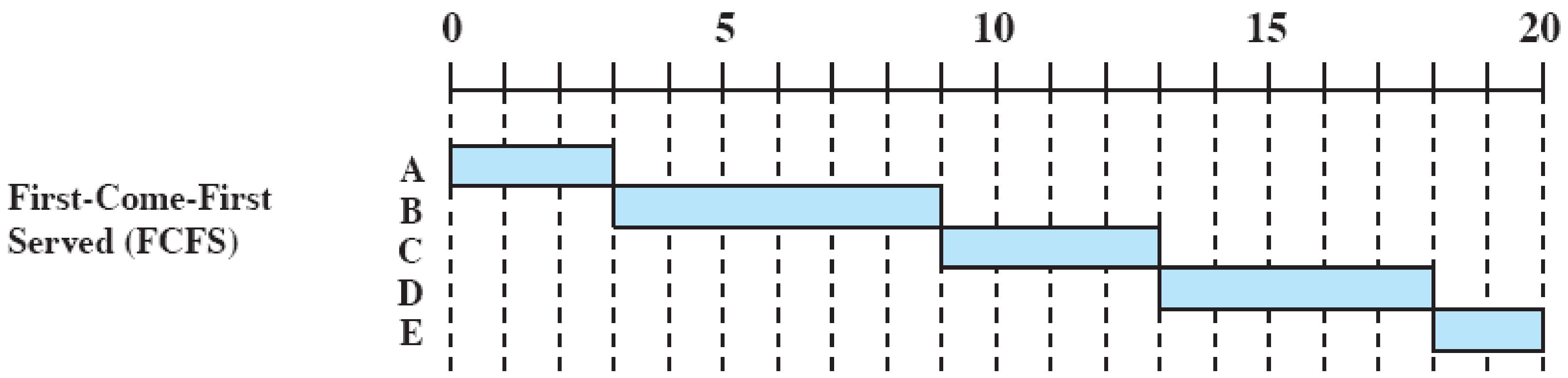


Table 9.4 Process Scheduling Example



Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

First Come First Served (FCFS)



- The simplest scheduling policy is first-come-first-served (FCFS),first-in-first-out (FIFO) or a strict queuing scheme.
- As each process becomes ready, it joins the ready queue.
- When the currently running process ceases to execute, the process that has been in the ready queue the longest is selected for running.

- Selection function: the process that has been waiting the longest in the ready queue (hence, FCFS)
- Decision mode: nonpreemptive, a process runs until it blocks itself

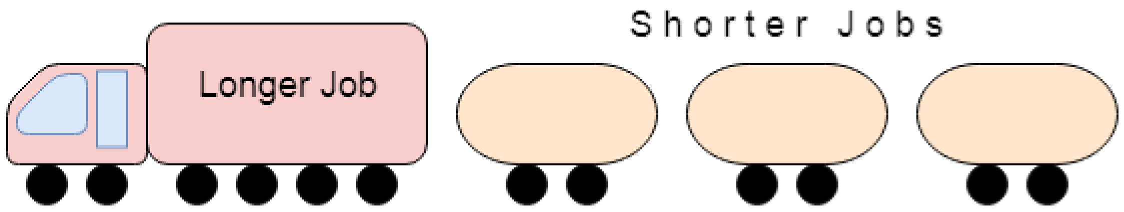
FCFS Drawbacks

- A short process may have to wait a very long time before it can execute.(convoy effect)
- A process that does not perform any I/O will monopolize the processor.
- Favours 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)
  - we could have kept the I/O devices busy by giving more priority to I/O Bound processes.

First Come First Served (FCFS)

- It may suffer from the **convoy effect** if the burst time of the first job is the highest among all.
- As in the real life, if a convoy is passing through the road then the other persons may get blocked until it passes completely. This can be simulated in the Operating System also.
- If the CPU gets the processes of the higher burst time at the front end of the ready queue then the processes of lower burst time may get blocked which means they may never get the CPU if the job in the execution has a very high burst time. This is called **convoy effect** or **starvation**.
- The average waiting time of the system will be very high

The Convoy Effect, Visualized Starvation

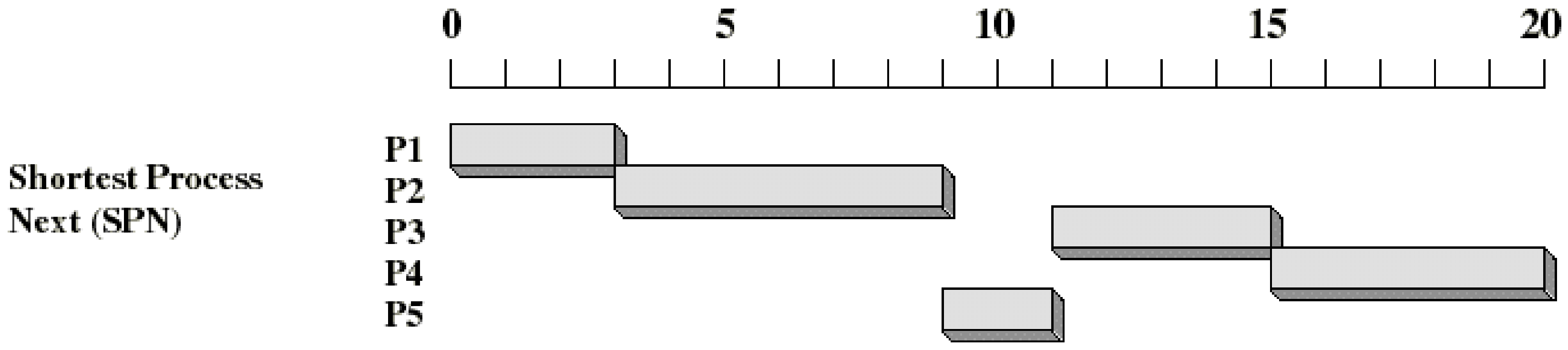


FCFS Drawbacks

- FCFS is not an attractive alternative on its own for a Uniprocessor system.
- However, it is often combined with a priority scheme to provide an effective scheduler.
- Thus, the scheduler may maintain a number of queues, one for each priority level, and dispatch within each queue on a first-come-first-served basis.

- Background processes

Shortest Process Next (SPN) (SJF-Nonpreemptive)



- Selection function: the process with the shortest expected CPU burst time
- Decision mode: **nonpreemptive**
- I/O bound processes will be picked first
- We need to estimate the required processing time (CPU burst time) for each process
- Average waiting time will be minimum.

Shortest Process Next: Critique

- Average waiting time will be minimum.
- Although it is optimal ,can not be implemented at the level of short term CPU scheduling
- As there is no way to know the length of the next cpu burst time.

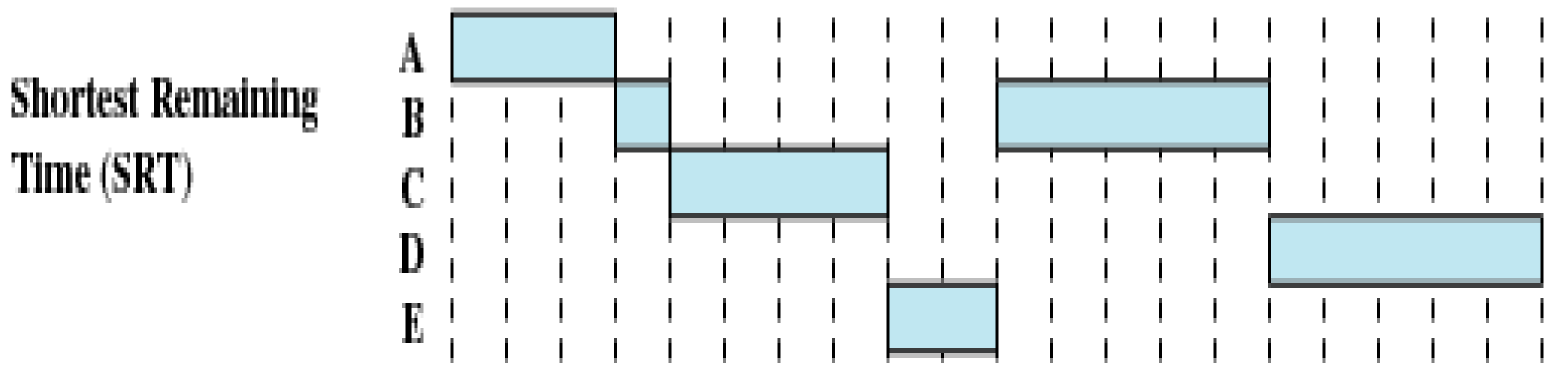
Shortest Process Next: Critique

- Possibility of starvation for longer processes as long as there is a steady supply of shorter processes
- Lack of pre-emption is not suited in a time sharing environment
  - CPU bound process gets lower priority (as it should) but a process doing no I/O could still monopolize the CPU if it is the first one to enter the system
- SPN implicitly incorporates priorities: shortest jobs are given preferences
- The next (pre-emptive) algorithm penalizes directly longer jobs

Shortest Process Next: Critique

- Average waiting time will be minimum
- FCFS is used to break the tie if two processes have same burst time
- SJF is a special case of general priority scheduling
- Priority is the inverse of the next cpu burst
- Larger the cpu burst, lower the priority & vice versa.

Shortest Remaining Time(SJF-Pre-emptive)



- Pre-emptive version of shortest process next policy
- Must estimate processing time

SRT / SJF P

- Estimate of remaining time required here as well
- Risk of starvation of longer processes
- No bias in favor of long processes as with FCFS
- No additional interrupts are generated unlike RR thereby reducing overhead
- But elapsed service times must be recorded, increasing the overhead
- Short job is given preference over running longer job, thus better turn around time

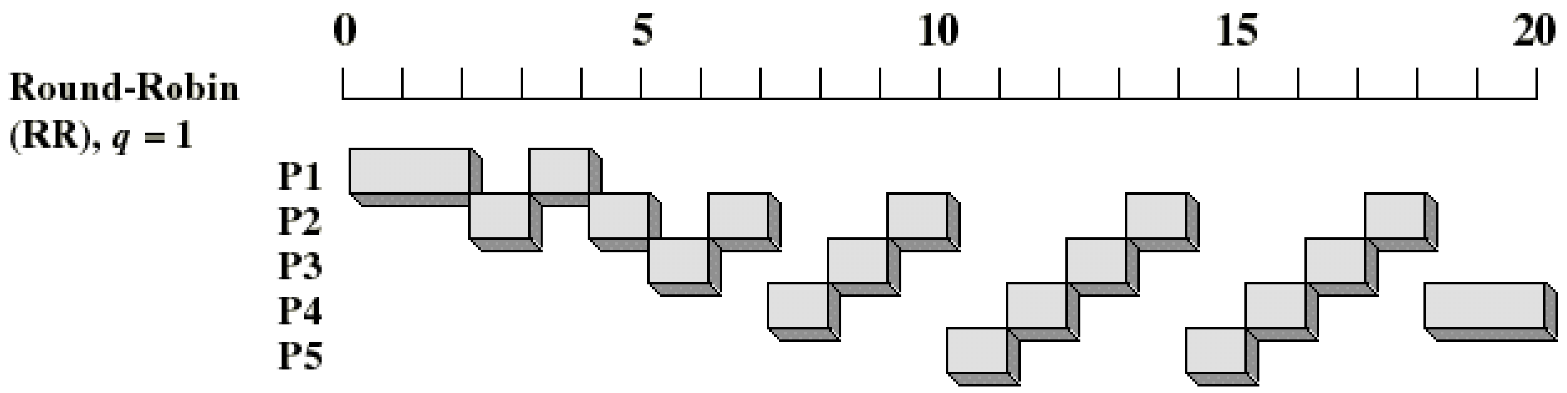
Priority Scheduling

- A priority number (integer) is associated with each process.
- The CPU is allocated to the process with the highest priority
- Some systems have a high number represent high priority.
- Other systems have a low number represent high priority.
- Text uses a low number to represent high priority.
- Priority scheduling may be preemptive or nonpreemptive.

Assigning Priorities

- SJF is a priority scheduling where priority is the predicted next CPU burst time.
- Useful for system processes
- Other bases for assigning priority:
  - Memory requirements
  - Number of open files
  - Avg I/O burst / Avg CPU burst
  - External requirements (amount of money paid, political factors, etc).
- Range of priority 0 to 7 or 0 to 4095
- Priorities can be defined either internally or externally
- Internal priority: time limit, no. of open files
- External priority: set by criteria outside the os e.g. imp of processes
- Problem: Starvation -- low priority processes may never execute.
- Solution: Aging -- as time progresses increase the priority of the process.(senior citizen)

Round-Robin



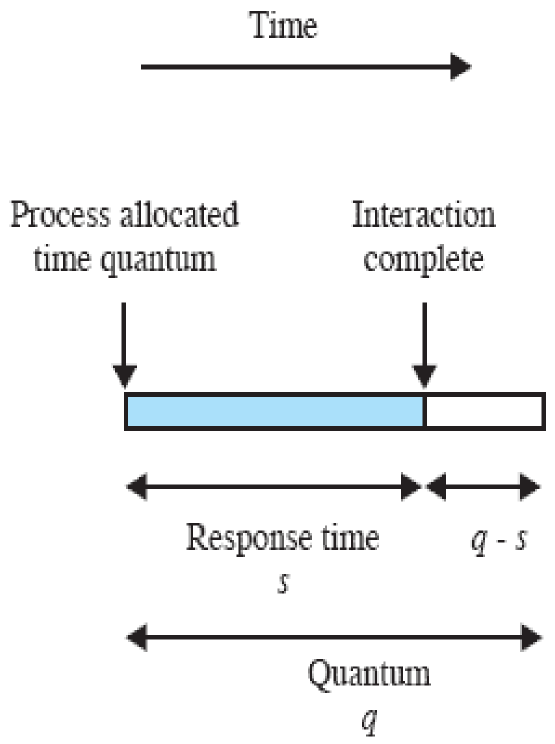
- Selection function: same as FCFS
- Decision mode: preemptive
  - a process is allowed to run until the time slice period (quantum, typically from 10 to 100 ms) has expired
  - then a clock interrupt occurs and the running process is put on the ready queue

Round-Robin

- Clock interrupt is generated at periodic intervals
- When an interrupt occurs, the currently running process is placed in the ready queue
- Next ready job is selected.
- The principal design issue is the length of the time quantum, or slice, to be used.
- If the quantum is very short, then short processes will move through the system relatively quickly.
    - BUT there is processing over-head involved in handling the clock interrupt and performing the scheduling and dispatching function.
    - Thus, very short time quantum should be avoided.

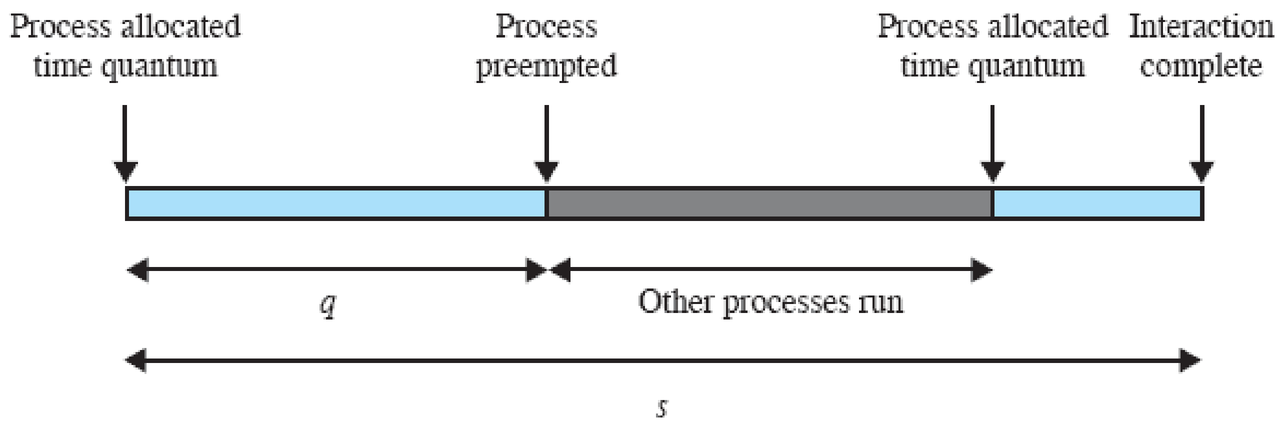
Effect of Size of Preemption Time Quantum

One useful guide is that the time quantum should be slightly greater than the time required for
a typical interaction or process function. If it is less, then most processes will require at least two time quanta.



(a) Time quantum greater than typical interaction

Effect of Size of Preemption Time Quantum


Process allocated time quantum
Process preempted
Process allocated time quantum
Interaction complete
q
Other processes run
s

(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum

Time Quantum for Round Robin

- Must be substantially larger than the time required to handle the clock interrupt and dispatching
- Should be larger than the typical interaction (but not much more to avoid penalizing I/O bound processes)

Round Robin: Critique

- Still favours CPU-bound processes
  - A I/O bound process uses the CPU for a time less than the time quantum, then is blocked waiting for I/O
  - A CPU-bound process runs for all its time slice and is put back into the ready queue (thus, getting in front of blocked processes)

Round Robin

- Advantages
  - Perform best for in terms of response time
  - Useful in Time sharing system,client-server and interactive system
  - Kind of SJF implementation
- Disadvantages
  - Longer processes may starve
  - Performance heavily depends on time quantum

Table 9.3 Characteristics of Various Scheduling Policies

	FCFS	Round Robin	SPN	SRT
Selection Function	max[w]	constant	min[s]	min[$s - e$]
Decision Mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)
Throughput	Not emphasized	May be low if quantum is too small	High	High
Response Time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time
Overhead	Minimum	Minimum	Can be high	Can be high
Effect on Processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes
Starvation	No	No	Possible	Possible

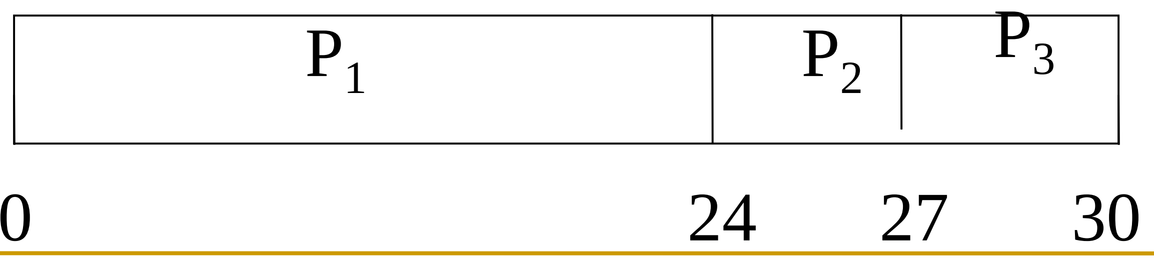
First-Come, First-Served (FCFS) Scheduling

Process that requests the CPU first is allocated the CPU first.
Easily managed with a FIFO queue.
Often the average waiting time is long.

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

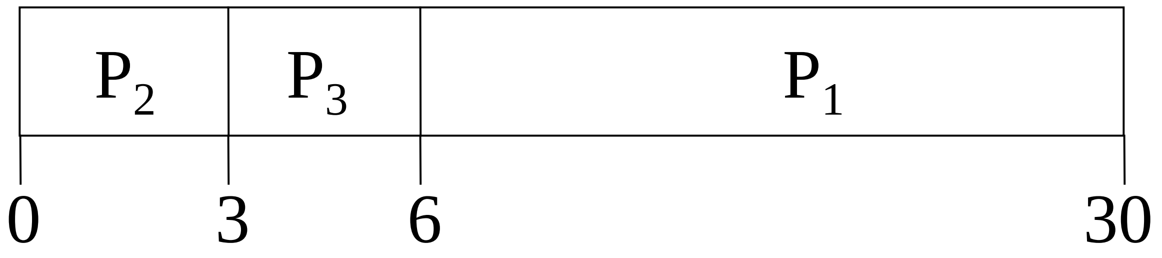
The Gantt Chart for the schedule is:



Suppose that the processes arrive in the order

P_2 , P_3 , P_1 .

The Gantt chart for the schedule is:



Practice

Process	Arrival Time	Burst Time	Priority
A	0	1	3
B	1	9	3
C	3	1	2
D	3	9	1

Draw Gantt Chart to find finish times and also calculate TT and WT in case of each of the following strategies:
-FCFS
-SJF Non Preemptive
-SJF Preemptive
-Round Robin
-Priority Non preemptive

End of Chapter