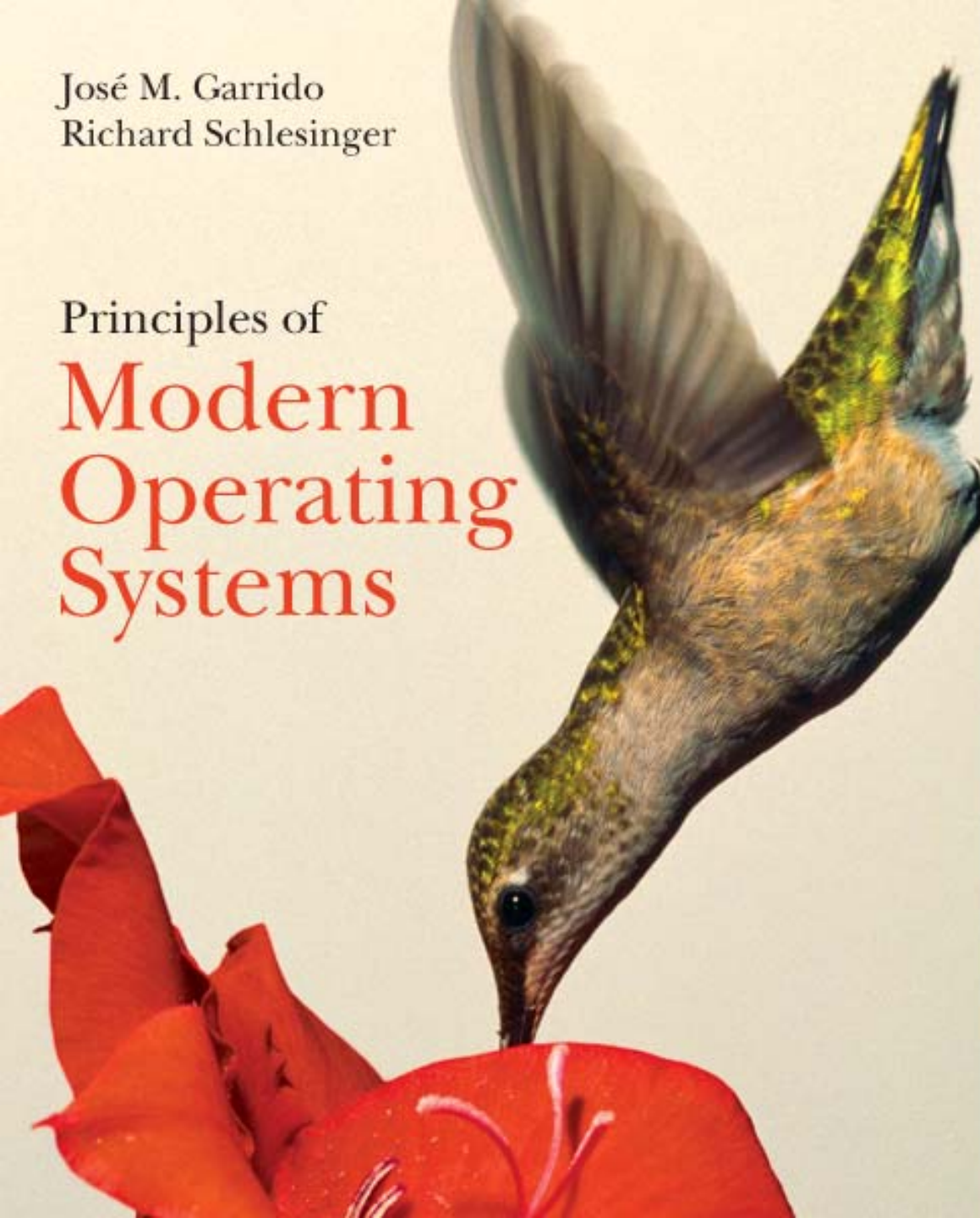


José M. Garrido
Richard Schlesinger

Principles of
**Modern
Operating
Systems**



Chapter 5

CPU Scheduling

Goals of CPU Scheduling

- CPU scheduling is the sharing of the CPU among the processes in the ready queue
- The critical activities are:
 - the ordering of the allocation and de-allocation of the CPU to the various processes and threads, one at a time
 - deciding when to de-allocate and allocate the CPU from a process to another process
- These activities must be carried out in such a way as to meet the performance objectives of the system

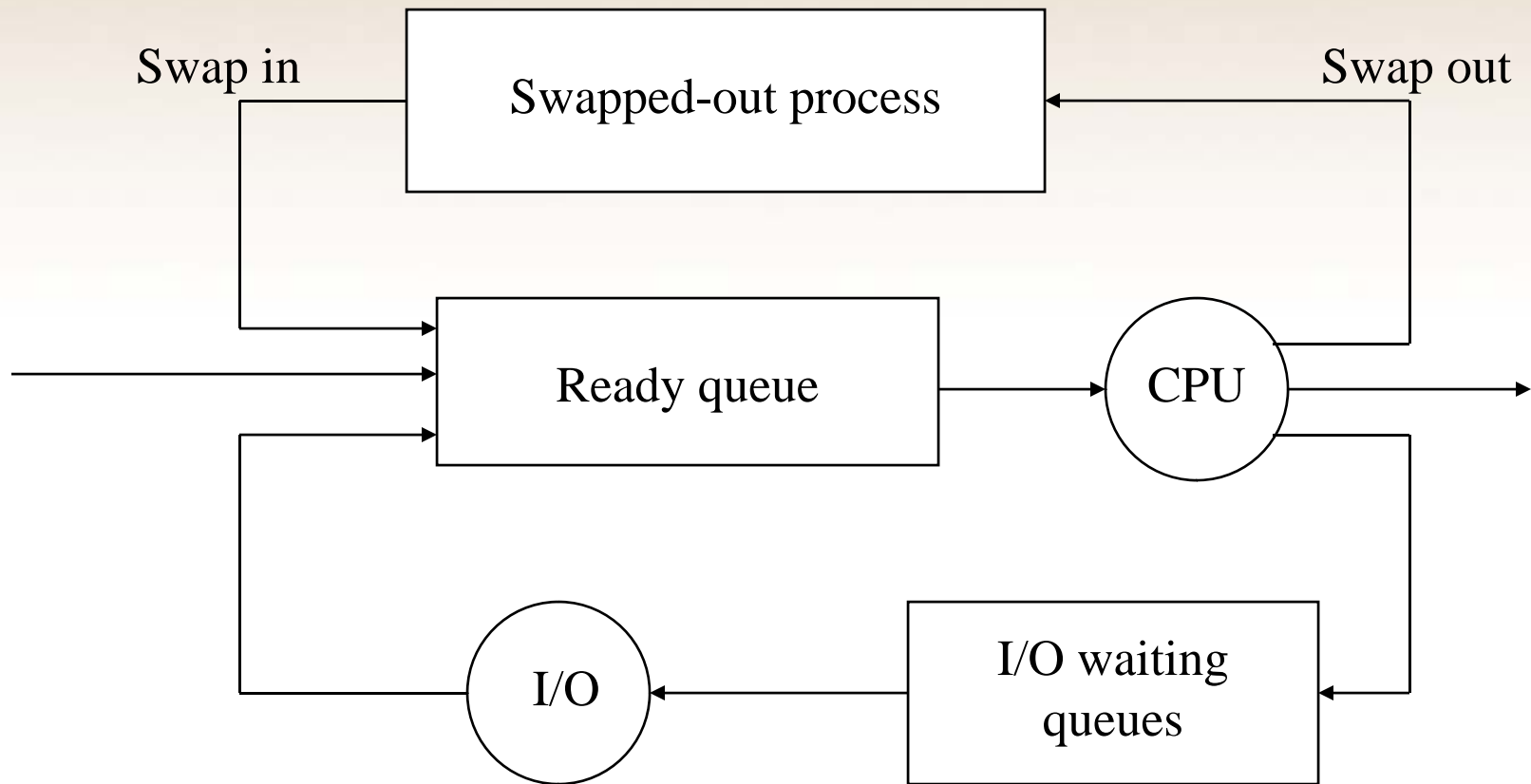
Types of Schedulers

- Long-term scheduler (memory allocation)
 - Determines which processes are loaded into memory
 - Controls the degree of multiprogramming
- Medium-term scheduler
 - Suspends (swaps out) and resumes (swaps in) processes
- Short-term scheduler (CPU scheduling)
 - Selects one of the processes that are ready and allocates the CPU to it.

Medium-Term Scheduler

The basic idea is that some of the processes can be removed from memory (to disk) and thus reduce the degree of multiprogramming. This is called swapping.

Medium-Term Scheduler



CPU Scheduling

- A CPU scheduling policy defines the order in which processes are selected from the ready queue for CPU processing.
- The scheduling mechanism also decides when and how to carry out the context switch to the selected process, i.e., the de-allocation of the CPU from the current process and allocation of the CPU to the selected process.

CPU Scheduling (2)

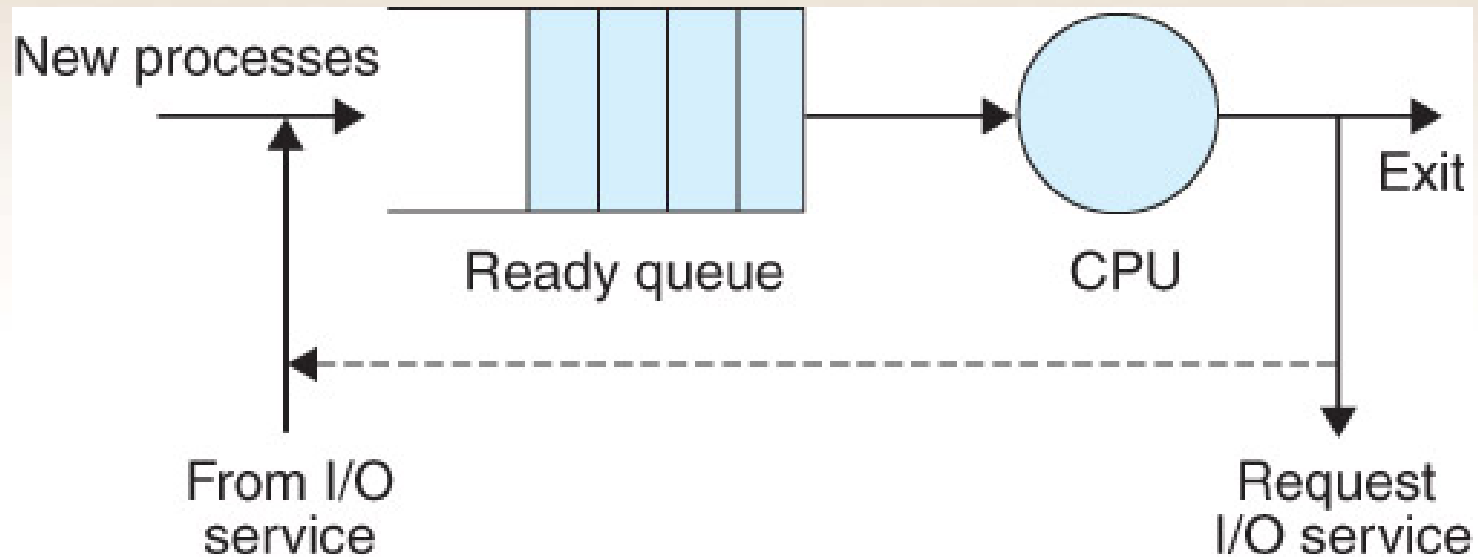
- The scheduler selects the next process to execute from among several processes waiting in the ready queue.
- The dispatcher allocates the CPU to the selected process at the appropriate time.

Process Activities

Every process that request CPU service, carries out the following sequence of actions:

1. Join the ready queue and wait for CPU service.
2. Execute (receive CPU) for the duration of the current CPU burst or for the duration of the time slice (timeout).
3. Join the I/O queue to wait for I/O service or return to the ready queue to wait for more CPU service.
4. Terminate and exit if service is completed, i.e., there are no more CPU or I/O bursts. If more service is required, return to the ready queue to wait for more CPU service.

Simple Model for CPU Scheduling



Scheduler

- Insertion of processes that request CPU service into the ready queue. This queue is usually a data structure that represents a simple first-in-first-out (FIFO) list, a set of simple lists, or as a priority list. This function is provided by the enqueueer, a component of the scheduler.

Scheduler

- The occurrence of a context switch, carried by the context switcher that saves the context of the current process and de-allocates the CPU from that process.
- The selection of the next process from the ready queue and loading its context. This can be carried out by the dispatcher, which then allocates the CPU to the newly selected process.

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler
 - context switching
 - switching to user mode
 - jumping to the proper location in the user program to restart (resume) that program
- Dispatch latency – the time that elapses from the stopping one process to the starting of the newly selected process

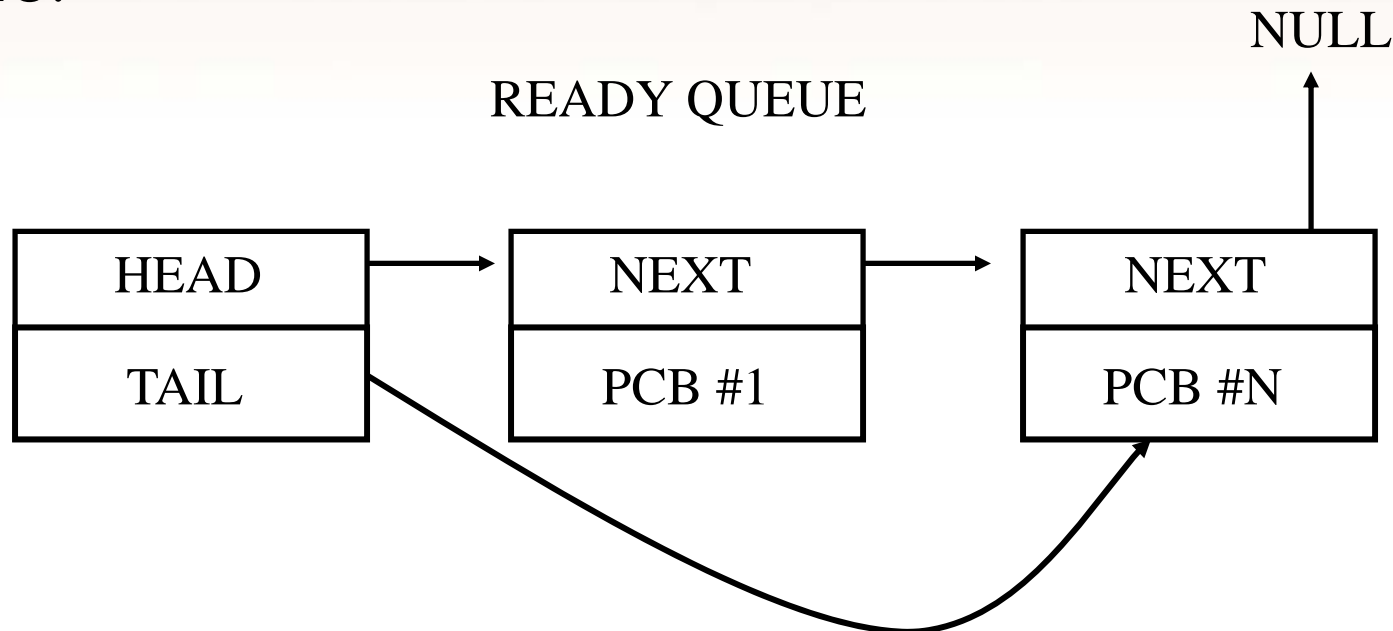
Occurrence of a Context Switch

A context switch can occur at any of the following possible times:

- The executing process has completed its current CPU burst. This is the normal case in simple batch systems.
- The executing process is interrupted by the operating system because its allocated time (time slice) has expired. This is a normal case in time-sharing systems.
- The executing process is interrupted by the operating system because a higher priority process has arrived requesting CPU service.

Process Queues

The processes that are ready and waiting to execute are kept on a list called the ready queue. A similar list exists for each I/O device, called the device queue.



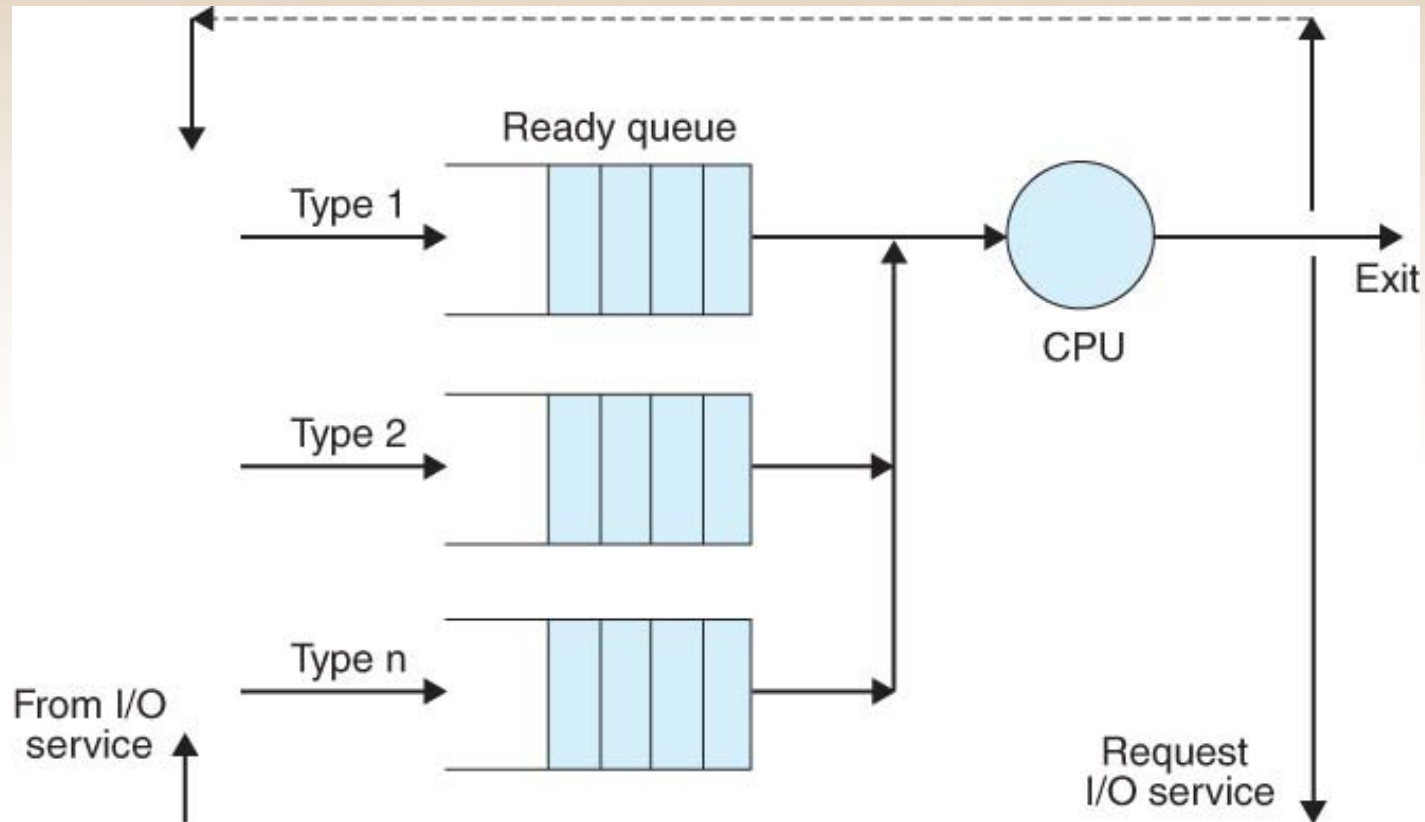
Process Queues

- Job queue (all processes)
- Ready queue
- Device queue
- Arrivals queue (external)

Multiple Classes of Processes

- There are several classes or types of processes
- For every class there can be a set of workload parameters:
 - mean inter-arrival period
 - mean service period
- Every class normally has an associated priority
- The scheduling of multiple classes of processes is not fair, the system gives preference to processes of some types

Scheduling with Multiple Queues



Scheduling of Multi-Class Systems

- Not Fair – Processes are not treated alike, preference is given to higher-priority processes
- A major problem in multi-class systems is **STARVATION**
 - indefinite waiting, one or more low-priority processes may never execute
 - solution is **AGING**

Criteria for Scheduling Algorithms

- CPU utilization
- Throughput
- Turnaround time
- Waiting time
- Response time
- Fairness

CPU-IO Bursts of Processes

An important property of a process is its CPU-IO burst

- An I/O bound process has many short CPU burst
- A CPU bound process has few long CPU bursts
- The OS tries to main maintain a balance of these two types pf processes

CPU Scheduling Policies

Categories of scheduling policies:

- Non-Preemptive -- no interrupts are allowed. A process completes execution of its CPU burst
- Preemptive – a process can be interrupted before the process complete its CPU burst

Priorities and Scheduling

- Priorities can be used with either preemptive or non-preemptive scheduling.
- Depending on the goals of an operating system, one or more of various scheduling policies can be used; each will result in a different system performance.
- The criteria are based on relevant performance measures and the various scheduling policies are evaluated based on the criteria.

CPU Scheduling Policies

- First-come-first-served (FCFS)
- Shortest job first (Shortest process next)
- Longest job first
- Priority scheduling
- Round robin (RR)
- Shortest remaining time (SRT) also known as shortest remaining time first (SRTF)

FCFS Scheduling

First come first served (FCFS) scheduling algorithm, a non-preemptive policy

- The order of service is the same order of arrivals
- Managed with FIFO queue
- Simple to understand and implement
- Scheduling is FAIR
- The performance of this scheme is relatively poor

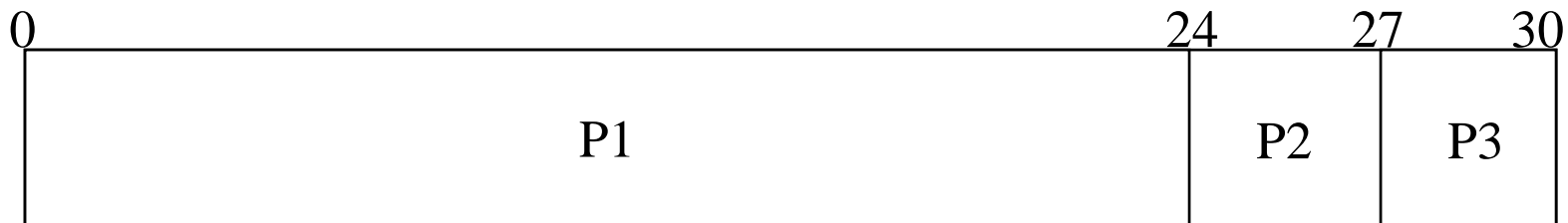
Deterministic Modeling FCFS

Example

If three CPU-bound processes arrive P1, P2, P3, with time bursts 24msec, 3msec, and 3msec. Respectively.

The average turnaround time is:

$$(24 + 27 + 30) / 3 = 27\text{msec}$$



Other Types of Non-Preemptive Scheduling

- Shortest Job First (SJF), or Shortest Process Next (SPN) -- The selection from the ready queue is based on the length of the next CPU burst.
- Priority scheduling -- An explicit priority is assigned to each process. Used in multi-class systems.

SJF (SPN) Scheduling

- The scheduler selects next the process with the shortest CPU burst
- Basically a non-preemptive policy
- SJF is optimal - gives the minimum average waiting time for a given set of processes.

Priority Scheduling (no Preemption)

- A priority is associated with every process type or class
- The next process selected from the ready queue is the one with the highest priority
- A high-priority process will be placed at the head of the ready queue
- The priority is implemented as an integer attribute of a process

Preemptive Scheduling

- A running process can be interrupted when its time slice expires (Round robin)
- A running process can be interrupted when its remaining time is longer than the CPU burst of an arriving process (SRTF)
- Priority preemptive scheduling - A currently running process will be preempted if a higher-priority process arrives (PPS)

Round Robin Scheduling

- A preemptive scheduling designed for Time Sharing Systems
- The Ready Queue is treated as a circular queue
- A small execution time interval is defined as the Time Quantum, or time slice

Interrupts in Round Robin

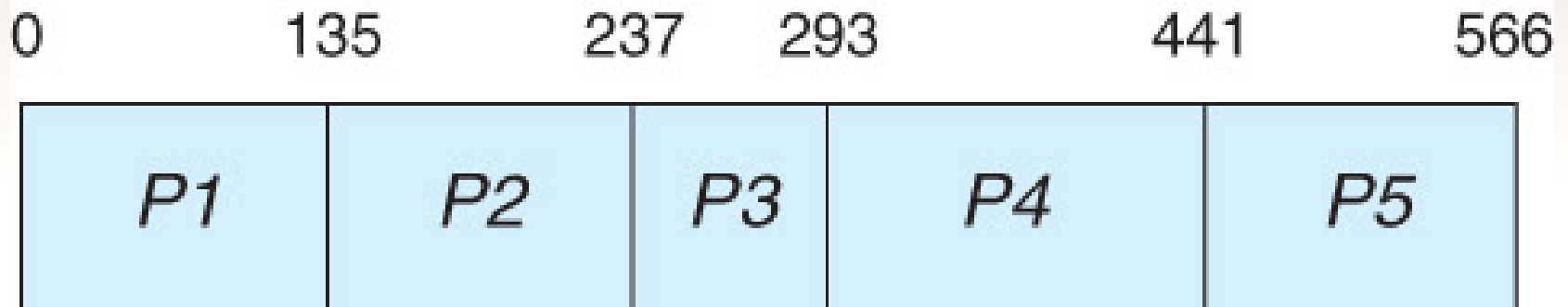
- When the executing interval of a process reaches the time quantum, the timer will cause the OS to interrupt the process
- The OS carries out a context switch to the next selected process from the ready queue.

Length of the Time Quantum

- Time quantum too short will generate many context switching and results in lower CPU efficiency.
- Time quantum too long may result in poor performance time.

Deterministic Modeling - FCFS

Consider the following workload: Five processes arrive at time 0, in the order: P1, P2, P3, P4, P5; with CPU burst times: 135, 102, 56, 148, 125 msec., respectively. The chart for FCFS is:

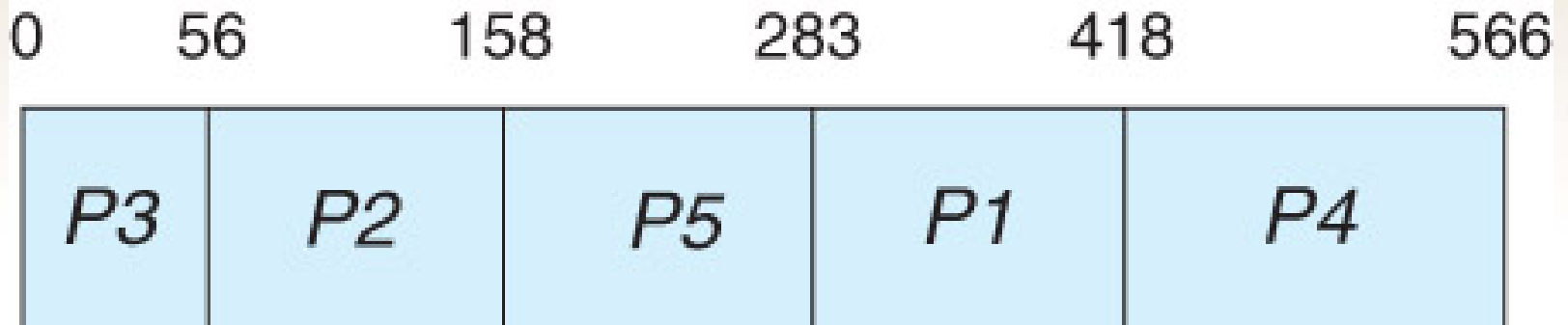


The average waiting time is:

$$(0 + 135 + 237 + 293 + 441) / 5 = 221 \text{ msec}$$

Shortest Process Next (SPN)

The chart for SPN is:



The average waiting time is:

$$(283+56+0+418+158) / 5 = 183 \text{ msec.}$$

Scheduling Algorithm Evaluation

- Criteria for evaluation, and measure performance of the computer system
 - Maximize CPU utilization under certain constraints
 - Minimize response time
 - Maximize throughput under certain constraints
- Analytic evaluation - use algorithm and system workload
 - Deterministic modeling
 - Queuing models
- Simulation - involves programming a model of the computer system

Example 3 Scheduling

Assume the following processes P1,P2, P3, P4 and P5 arrive at 1, 2, 4, 5, 5 respectively.

The CPU burst and the priority assigned to each process are:

P1:	45	3
P2:	5	5
P3:	15	2
P4:	20	1
P5:	25	4

For FCFS, RR, SJF and PR scheduling, determine a) the turnaround time for every process, b) waiting time for every process and the average waiting time, c) throughput for the system.

Use a time quantum of 10 time units, and negligible context time.

Shortest Remaining Time First

- Shortest remaining time (SRTF) is a preemptive version of SPN scheduling.
- With this scheduling policy, a new process that arrives will cause the scheduler to interrupt the currently executing process if the CPU burst of the newly arrived process is less than the remaining service period of the process currently executing.
- There is then a context switch and the new process is started immediately.

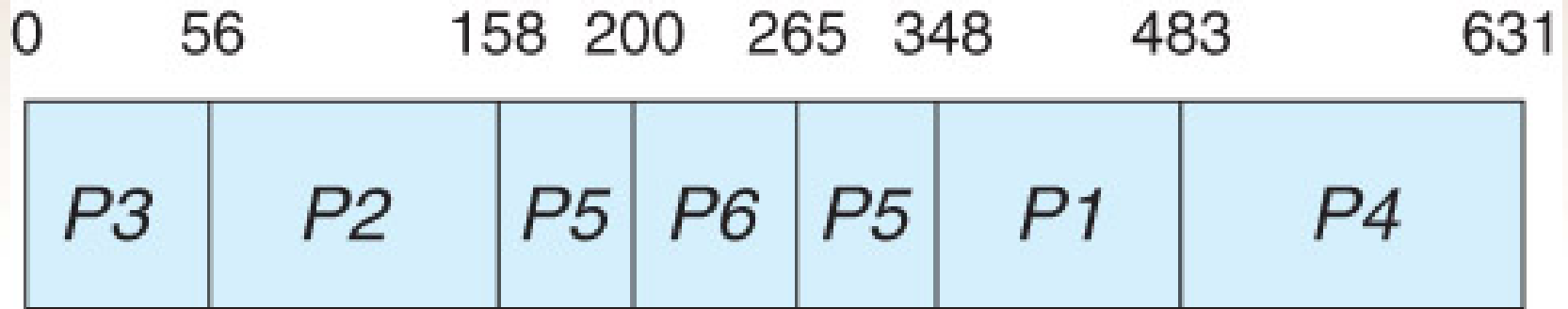
SRTF Scheduling

- When there are no arrivals, the scheduler selects from the ready queue the process with the shortest CPU service period (burst).
- As with SPN, this scheduling policy can be considered multi-class because the scheduler gives preference to the group of processes with the shortest remaining service time and processes with the shortest CPU burst.

SRTF Example

Process	Arrival time	CPU burst
P1	0	135
P2	0	102
P3	0	56
P4	0	148
P5	0	125
P6	200	65

Gantt Chart for SRTF Example



SRTF Example

- The scheduler compares the CPU burst of process P6 with the remaining service time of process P5, which is the currently executing process.
- At time 200, P5 has executed during 42 msec and has remaining service time of 83 microseconds, which is greater than the CPU burst of P6.
- Therefore, P5 is interrupted, a context switch to P6 is carried out, and P6 starts immediately.

Results of Example Using SRT

Process	Start	Completion	Wait	Turnaround	Ntat
P1	348	483	348	483	3.577
P2	56	158	56	158	1.549
P3	0	56	0	56	1.0
P4	483	631	483	631	4.263
P5	158	348	223	348	2.784
P6	200	265	0	65	1.0

The average wait period is: 185.0 microsec.

The average turnaround time is: 290.16 microsec.

Dynamic Priority Scheduling

- Processes that request I/O service will typically start with CPU service; for a short time; request another I/O operation; and release the CPU.
- If these processes are given higher priority, they can keep the I/O devices busy without using a significant amount of CPU time.
- This will tend to maximize I/O utilization while using a relatively small amount of CPU time.
- The remaining CPU capacity will be available for processes that are requesting CPU bursts.

Dynamic Priority Scheduling (2)

- The CPU scheduler dynamically adjusts the priority of a process as the process executes.
- The typical approach is to adjust the priority based on the level of expectation that the process will carry out a system call (typically an I/O request).
- However, this requires the CPU scheduler to predict future process requests.

Implementing Dynamic Priority

- The OS can apply an estimation or approximation (also known as a heuristic) that is based on observed program behavior
 - *A process will tend to carry out in the near future what it has done in the recent past.*

Heuristic Algorithm

1. Allocate the CPU to the highest priority process.
2. When a process is selected for execution, assign it a time-slice.
3. If the process requests an I/O operation before the time-slice expires, raise its priority (i.e. assume it will carry out another I/O request soon)
4. If the time-slice expires, lower its priority (i.e., assume it is now in a CPU burst) and allocate the CPU to the highest priority ready process.

Real-Time Systems

- A system that maintains an on-going interaction with its environment
- One of the requirements of the system is its strict timing constraints
- The system depends on priorities and preemption

Real-Time Scheduling Policies

- These scheduling policies attempt to maintain the CPU allocated to the high-priority real-time processes.
- One of the goals for this kind of scheduling is to guarantee fast response of the real-time processes.

Real-Time Processes

- The real-time processes, each with its own service demand, priority and deadline, compete for the CPU.
- Real-time processes must complete their service before their deadlines expire.
- The second general goal of a real-time scheduler is to guarantee that the processes can be scheduled in some manner in order to meet their individual deadlines.
- The performance of the system is based on this guarantee.

Real-Time Processes (2)

- The real-time processes can be:
 - Periodic processes need to be executed every specific interval (known as the period)
 - Sporadic processes can be started by external random events.
- The operating system uses a real-time scheduling policy based on priorities and preemption.

Real-Time Scheduling Policies

- There are two widely known real-time scheduling policies:
 - The rate monotonic scheduling (RMS)
 - The earliest deadline first scheduling (EDFS).

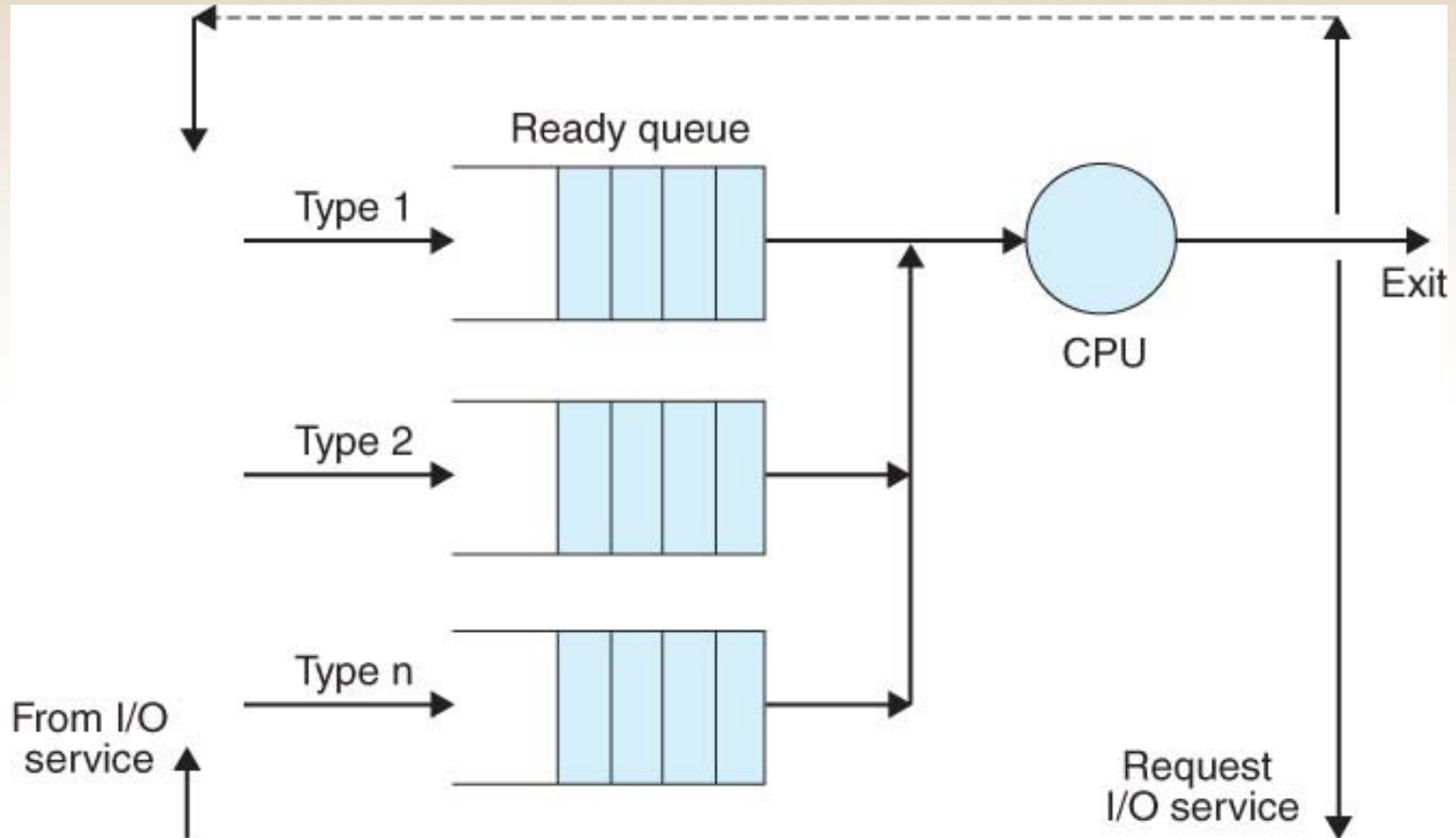
Multi - Queue Scheduling

- The ready queue is partitioned into separate queues, based on some property of the processes
- Each queues has its own scheduling algorithm
- For example, one queue for each of the following types of processes:
 - System processes
 - Interactive processes
 - Editing processes
 - Batch processes
 - Other low priority processes

Multilevel Queues

- In multi-class systems there are several classes of processes and each class of process is assigned a different priority.
- Multilevel queues are needed when the system has different categories of processes.
- Each category needs a different type of scheduling.
- For example, one category of process requires interactive processing and another category requires batch processing.
- For each category there may be more than one priority used.

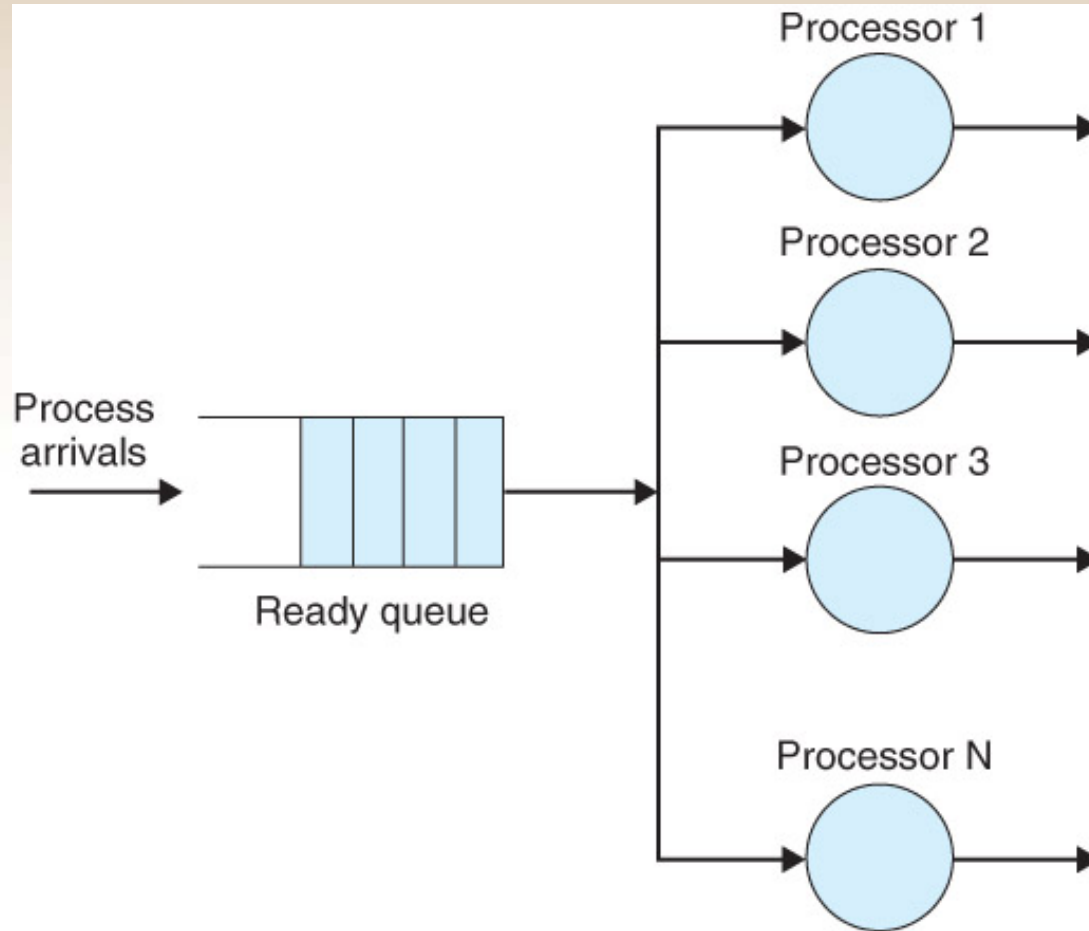
Multiple Ready Queues



Multiple Processors

- For multiple-processor systems, there are several ways a system is configured.
- The scheduling of the processes on any of the processors is the simplest approach to use. This assumes that the processors are tightly coupled, that is, they share memory and other important hardware/software resources.
- More advanced configurations and techniques for example, parallel computing, are outside the scope of this book.

Single Queue-Multiple Processors



Multiple Queues-Multiple Processors

