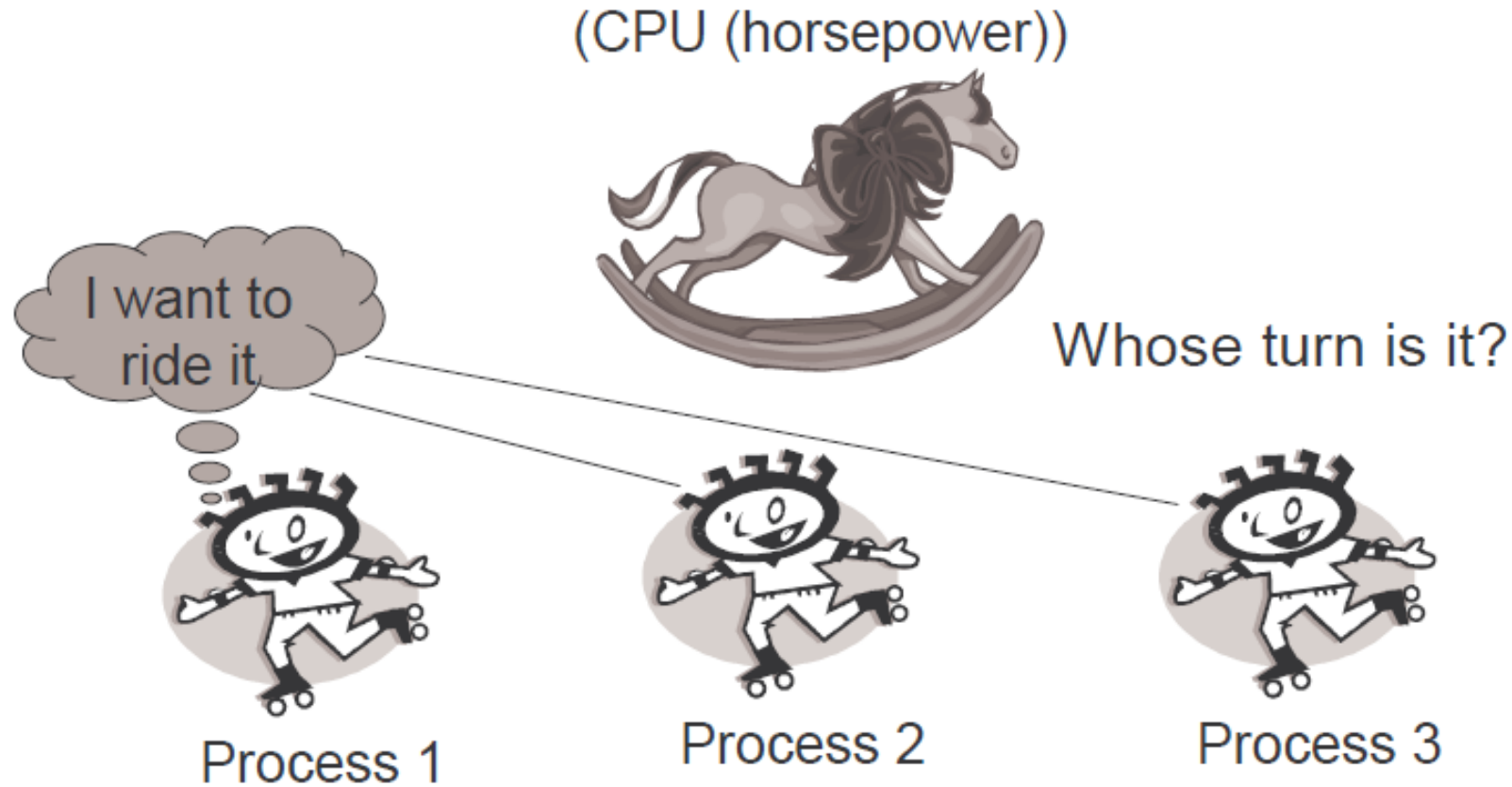


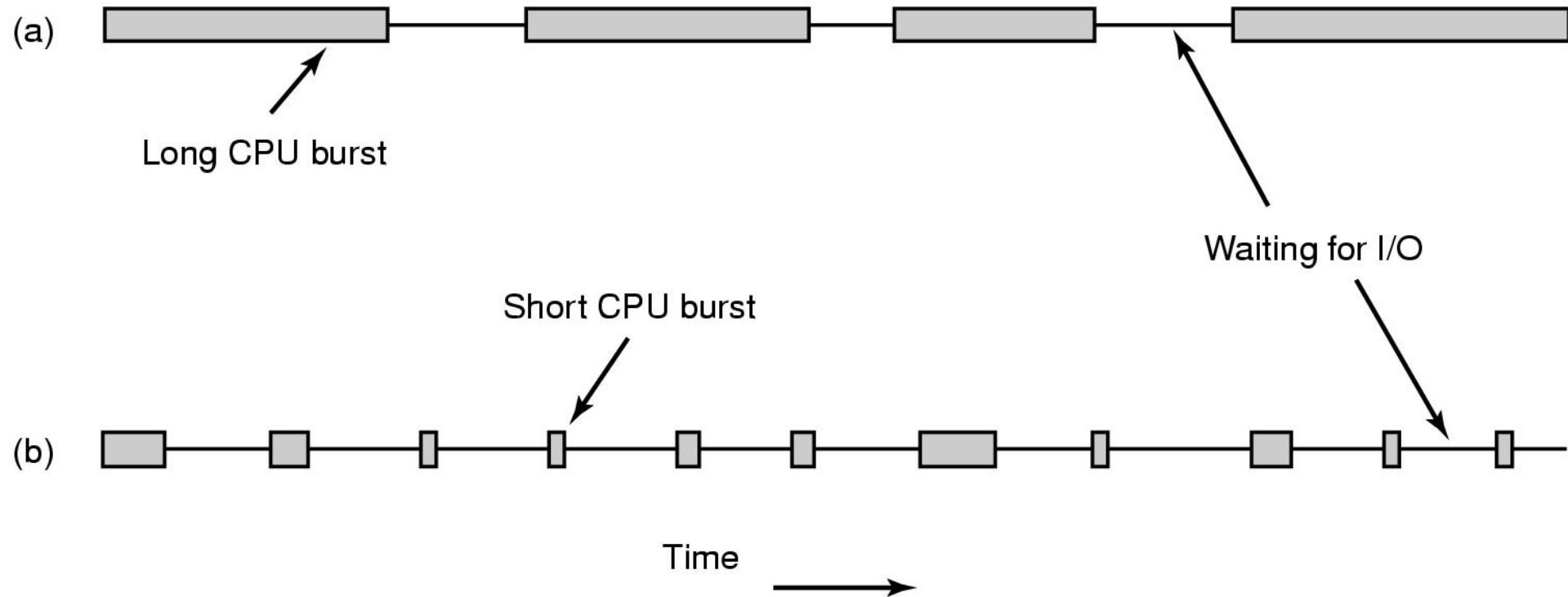
# Scheduling

Operating Systems

# Who gets the CPU?



# Process Behaviour



- a) **CPU-bound** processes spend most of their time computing
- b) **I/O bound** processes spend most of their time waiting for I/O

# Multiprogramming

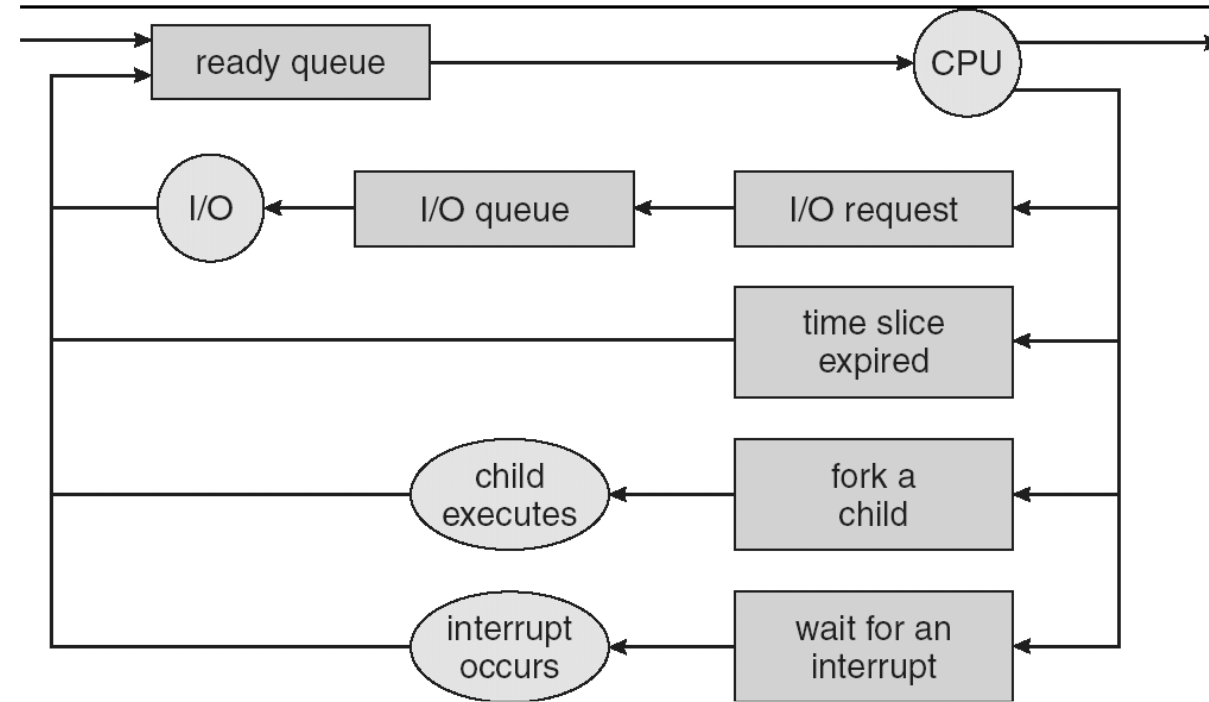
- Overlapping I/O and CPU activities
  - To increase CPU utilization and job throughput
- Previously covered the mechanisms of
  - Context switching
  - Process queues and process states
- But...
  - **which** process (thread) to run, for **how long**, etc. – scheduling

# Scheduling

- Choosing which process to run next, when two or more of them are simultaneously in the ready state
- Deciding which process should occupy the resource (CPU, disk, etc.)
- Done by **scheduler** using the **scheduling algorithm**
- Many of the same issues that apply to process scheduling also apply to thread scheduling, although some are different.
- **Jobs** - schedulable entities (processes, threads)

# When to schedule?

- when a job exits
- when a job blocks on I/O
- when time slice expired
  - a hardware clock provides periodic interrupts
- when a new job is created
  - whether to run the parent or the child
- when an I/O interrupt occurs
  - from an I/O device that has now completed its work for a waiting job



# Performance Criteria

- Throughput
  - number of jobs completed in unit time
- Turnaround time (elapse time)
  - Amount of time to execute a particular process from the time it entered
- Waiting time
  - Amount of time process has been waiting in ready queue
- Meeting deadlines
  - Avoid bad consequences



# Scheduling Objectives

- Fair
  - Everyone is happy
- Priority
  - Some are more important
- Efficiency
  - Make best use of equipment
- Encourage good behavior
  - Good boy/girl
- Support heavy load
  - Degrade gracefully
- Adapt to different environment
  - Interactive, real-time, multi-media



# Categories of Scheduling Algorithms

## 1. Batch

- Periodic tasks – payroll, bills, interest calculation (at banks)
- No users impatiently waiting
- Possible to run for long time periods for each process without switching

## 2. Interactive

- For environments with interactive users – personal computing, servers
- One process cannot be hogging the CPU and denying service to the others

## 3. Real-time

- Only programs that are intended to further the application at hand
- Processes may not run for long and usually do their work and block quickly
- So, it's okay to let them finish

# Preemptive vs. Non-preemptive

- Non-preemptive scheduling
  - The running process keeps the CPU until it **voluntarily** gives up the CPU
- Preemptive scheduling
  - The running process can be **interrupted** and must release the CPU

# Scheduling Algorithm Goals

## **All systems**

Fairness - giving each process a fair share of the CPU

Policy enforcement - seeing that stated policy is carried out

Balance - keeping all parts of the system busy

## **Batch systems**

Throughput - maximize jobs per hour

Turnaround time - minimize time between submission and termination

CPU utilization - keep the CPU busy all the time

## **Interactive systems**

Response time - respond to requests quickly

Proportionality - meet users' expectations

## **Real-time systems**

Meeting deadlines - avoid losing data

Predictability - avoid quality degradation in multimedia systems

# Scheduling Algorithms

- Batch Systems
  - First-Come, First-Served (FCFS)
  - Short Job First (SJF)
- Interactive Systems
  - Round-Robin Scheduling
  - Priority Scheduling
  - Multi-Queue & Multi-Level Feedback
- Real-time Systems
  - Earliest Deadline First Scheduling

# First-Come, First-Served (FCFS)

- “Real-world” scheduling of people in lines (e.g., supermarket)
- A single queue of ready jobs
- Jobs are scheduled in order of arrival to ready queue
- Typically non-preemptive (no context switching at market)
- Jobs treated equally, no starvation.
- When the running process blocks, the first process on the queue is run next.
- When a blocked process becomes ready, like a newly arrived job, it is put on the end of the queue, behind all waiting processes.

# First-Come, First-Served – Example

Process	Duration	Order	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0



P1 waiting time: 0  
P2 waiting time: 24  
P3 waiting time: 27

The average waiting time:  
 $(0+24+27)/3 = 17$

# First-Come, First-Served – Problems

- Average waiting time can be large
  - If small jobs wait behind long ones (high turnaround time)
  - Non-preemptive
  - You're stuck behind someone with a cart, when you only have two items
- Solution?
  - Express lane (10 items or less)



# Shortest Job First (SJF)

- Choose the job with the smallest expected duration first
  - Person with smallest number of items to buy
- Requirement
  - the job duration needs to be known in advance
- Used in Batch Systems
- Optimal for Average Waiting Time if all jobs are available simultaneously



# Shortest Job First – Example

Process	Duration	Order	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0



P4 waiting time: 0  
P1 waiting time: 3  
P3 waiting time: 9  
P2 waiting time: 16

The total time is: 24

The average waiting time (AWT):  
 $(0+3+9+16)/4 = 7$

# FCFS vs. SJF

Process	Duration	Order	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0



P1 waiting time: 0  
P2 waiting time: 6  
P3 waiting time: 14  
P4 waiting time: 21

The total time is the same (why?)  
The average waiting time (AWT):  
 $(0+6+14+21)/4 = 10.25$   
(comparing to 7)

# Shortest Job First – Problems

- Starvation
  - a job is waiting forever
- All jobs must be available at start
  - Suited for batch systems

Process	Duration	Order	Arrival Time
P1	10	1	0
P2	2	2	2



P1 waiting time: 0  
P2 waiting time: 8

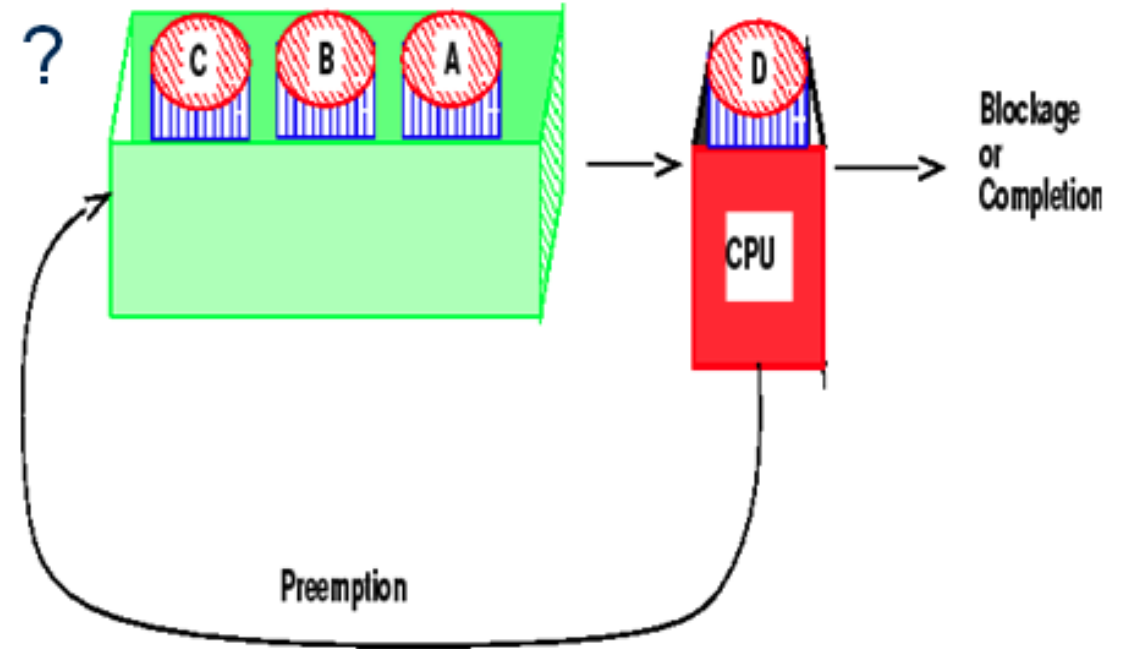
The average waiting time (AWT):  
 $(0+8)/2 = 4$

# Scheduling Algorithms

- Batch Systems
  - First-Come, First-Served (FCFS)
  - Short Job First (SJF)
- Interactive Systems
  - Round-Robin Scheduling
  - Priority Scheduling
  - Multi-Queue & Multi-Level Feedback
- Real-time Systems
  - Earliest Deadline First Scheduling

# Round-Robin Scheduling

- One of the oldest, simplest, most commonly used scheduling algorithm
- Select process/thread from ready queue in a round-robin fashion (take turns)



# Round-Robin Scheduling – Example

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is: 1 unit, P1, P2 & P3 never block

P1 P2 P3 P1 P2 P3 P1 P2 P3 P2



P1 waiting time: 4

P2 waiting time: 6

P3 waiting time: 6

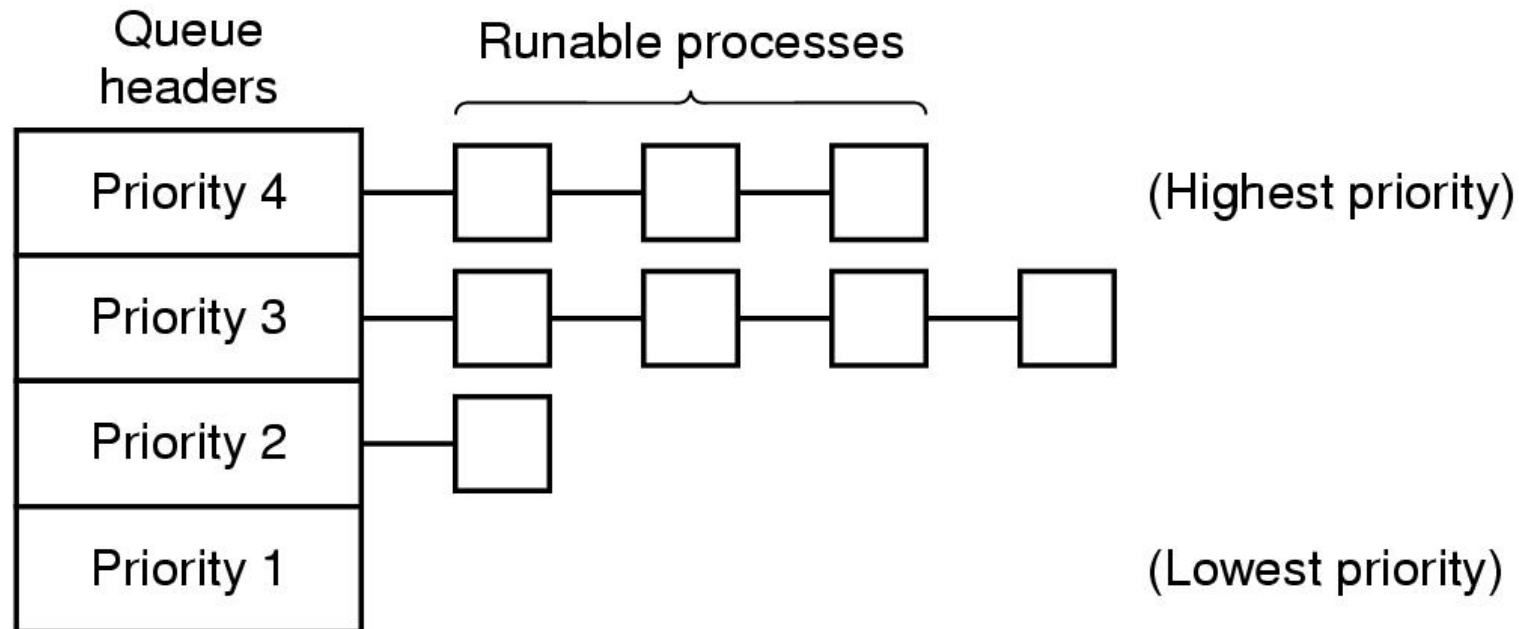
The average waiting time (AWT):  
 $(4+6+6)/3 = 5.33$

# Round-Robin Scheduling – Problems

- Time slice too large
  - FIFO behavior
  - Poor response to short interactive requests
- Time slice too small
  - Too many context switches (overheads)
  - Inefficient CPU utilization
- A quantum around 20–50 msec is often a reasonable compromise.

# Priority Scheduling

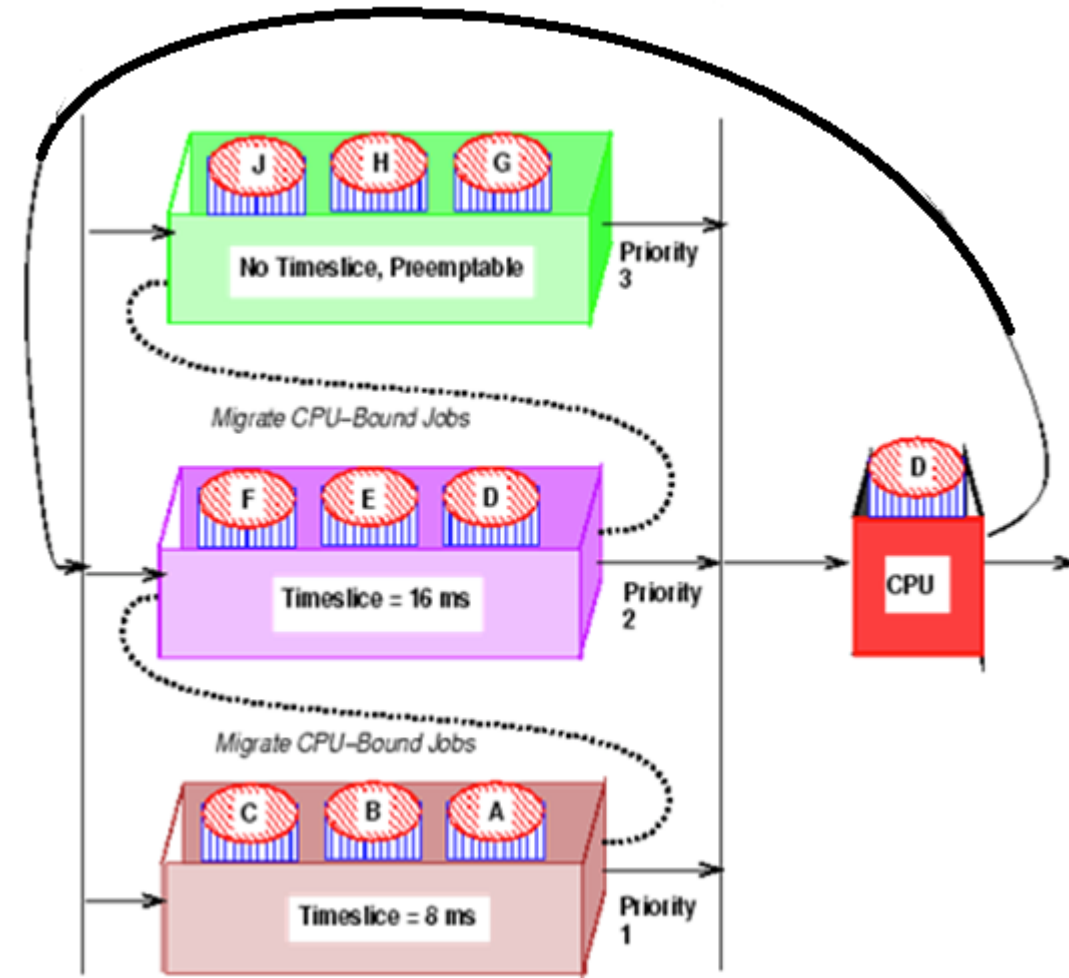
- Not all processes are equally important
- Need to consider external factors
- Email checking less priority than displaying video





# Multiple-level feedback queues (MLFQ)

- Scheduling algorithms can be combined
  - Have multiple queues
  - Use a different algorithm among queues
  - Move processes among queues
- Multiple queues representing different job types
  - Interactive, CPU-bound, batch, etc.
  - Queues have priorities
  - Jobs can move among queues based upon execution history

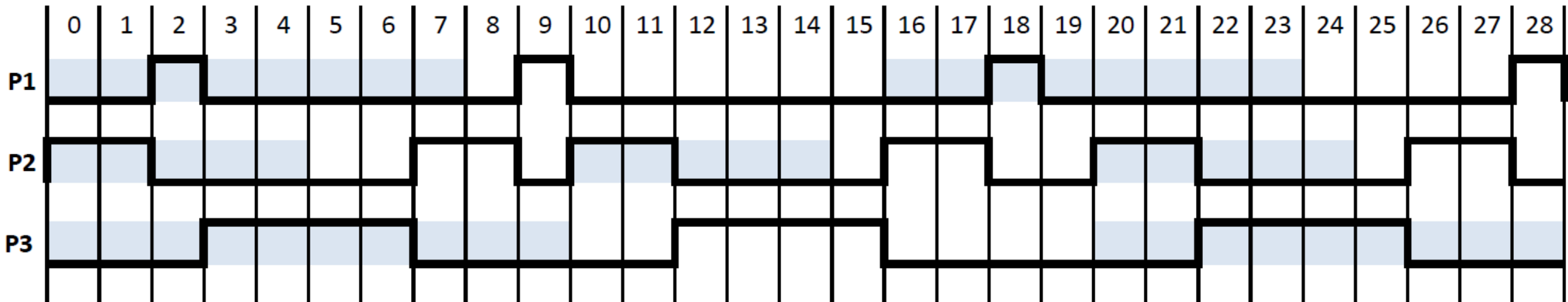


# Scheduling Algorithms

- Batch Systems
  - First-Come, First-Served (FCFS)
  - Short Job First (SJF)
- Interactive Systems
  - Round-Robin Scheduling
  - Priority Scheduling
  - Multi-Queue & Multi-Level Feedback
- Real-time Systems
  - Earliest Deadline First Scheduling

# Earliest Deadline First (EDF)

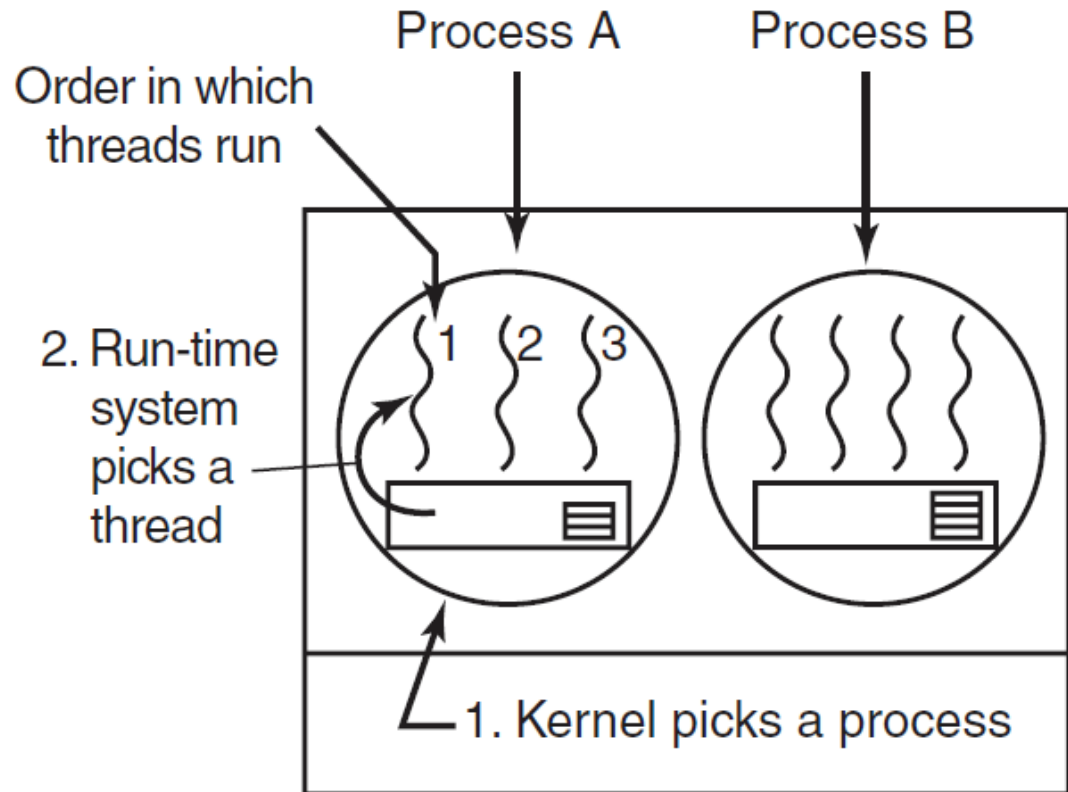
- Each job has an arrival time and a *deadline* to finish
  - Assignments, exams\*
- Always pick the job with the earliest deadline to run



# Thread Scheduling

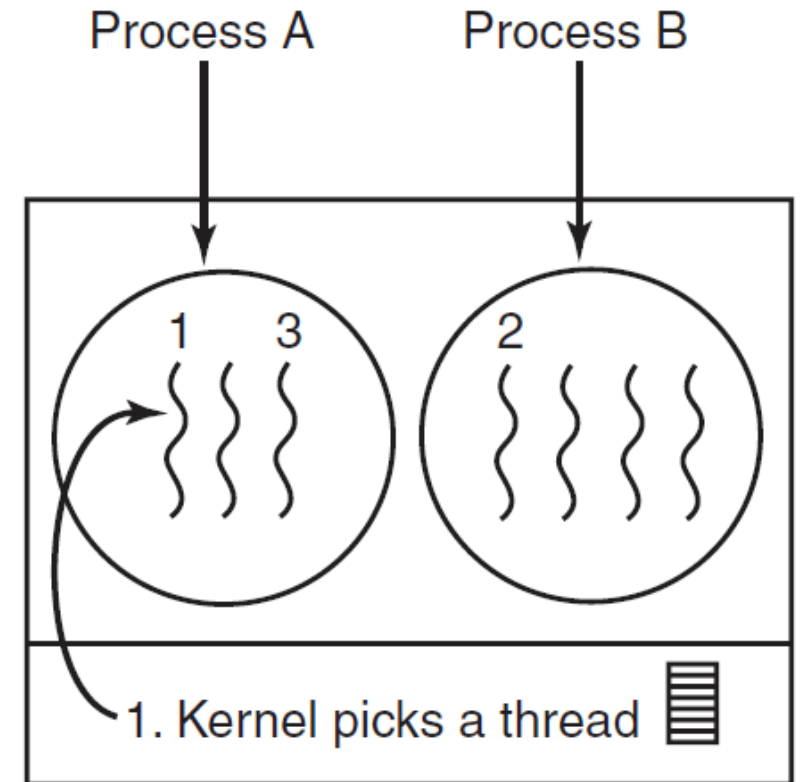
- Two levels of threads
  - User-level threads
  - Kernel-level threads
- User-level threads
  - Kernel picks the process
  - Scheduler inside process picks thread
- Kernel-level threads
  - Kernel picks a particular thread to run
  - Requires a full context switch

# Thread Scheduling



Possible: A1, A2, A3, A1, A2, A3  
Not possible: A1, B1, A2, B2, A3, B3

(a)



Possible: A1, A2, A3, A1, A2, A3  
Also possible: A1, B1, A2, B2, A3, B3

(b)

# Scheduling Summary

- Scheduler is the module that gets invoked when a context switch needs to happen
- Scheduling algorithm determines which process runs and where processes are placed on queues
- Scheduling algorithms have many goals
  - Utilization, throughput, wait time, response time, etc.
- Various algorithms to meet these goals
  - FCFS/FIFO, SJF, RR, Priority
- Can combine algorithms
  - Multiple-level feedback queues