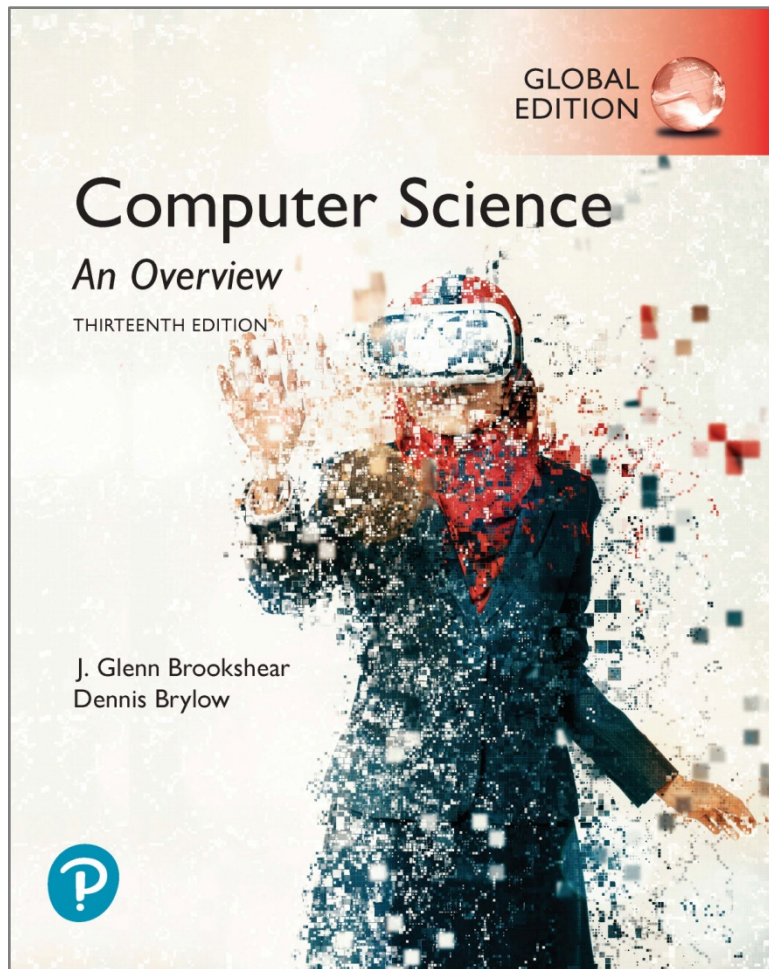# Computer Science An Overview

## 13th Edition, Global Edition

# Chapter 3

Operating Systems

# Chapter 3: Operating Systems

- 3.1 The History of Operating Systems

- 3.2 Operating System Architecture

- 3.3 Coordinating the Machine's Activities

- 3.4 Handling Competition Among Processes

- 3.5 Security

# Examples of Operating Systems

- Windows
- UNIX
- Mac OS
- Solaris (Sun/Oracle machines)
- Linux

# Smartphone Operating Systems

- Apple iOS

- Windows Phone

- BlackBerry OS

- Nokia Symbian OS

- Google Android

# Functions of Operating Systems

- Oversee operation of computer

- Store and retrieve files

- Provide the user interface to request execution of programs

- Coordinate the execution of programs

# 3.1 History of Operating Systems

- Each program is called a "job"

- Early computers required significant setup time

- Each "job" required its own setup

- Operating Systems began as systems for simplifying setup and transitions between jobs

# 3.1 History of Operating Systems

- Batch processing (job queue)

- Interactive processing (real time)

- Time-sharing (one machine, many users)

- Multitasking (one user, many tasks)

- Multiprocessor machines (load balancing)

- Embedded Systems (specific devices)
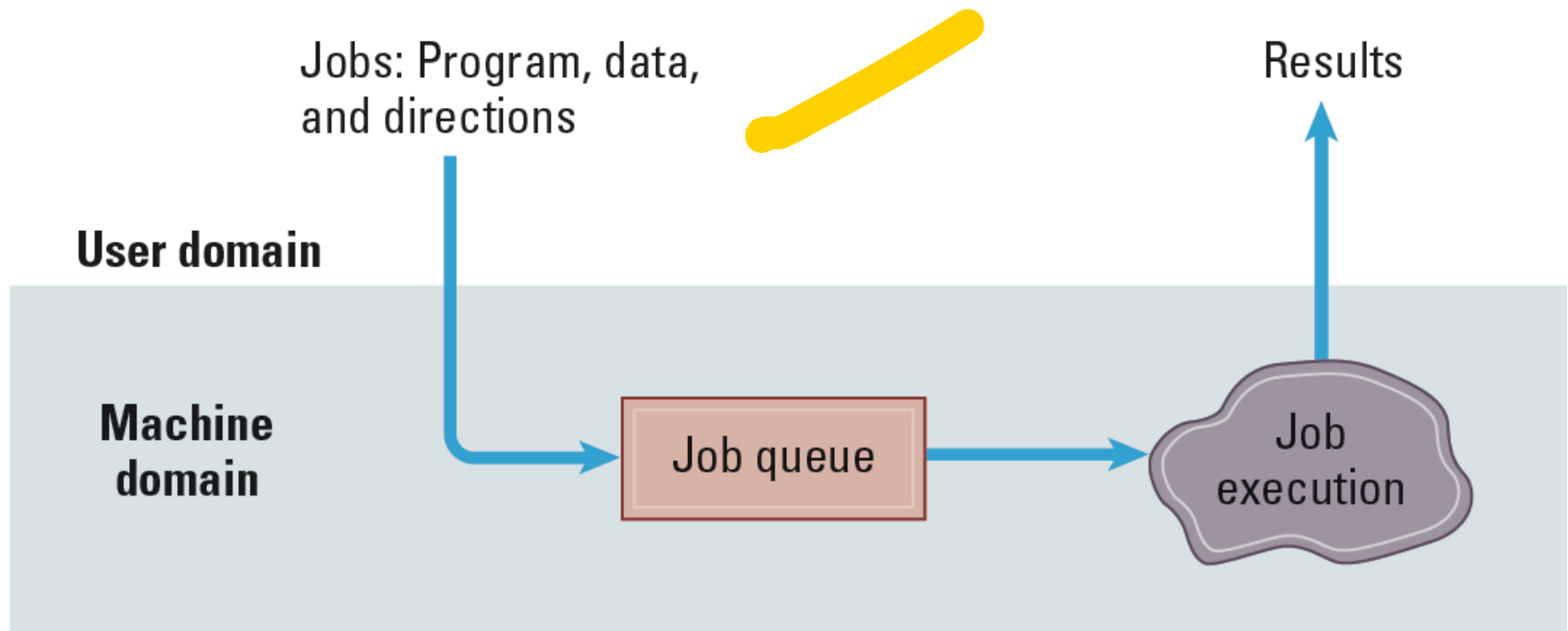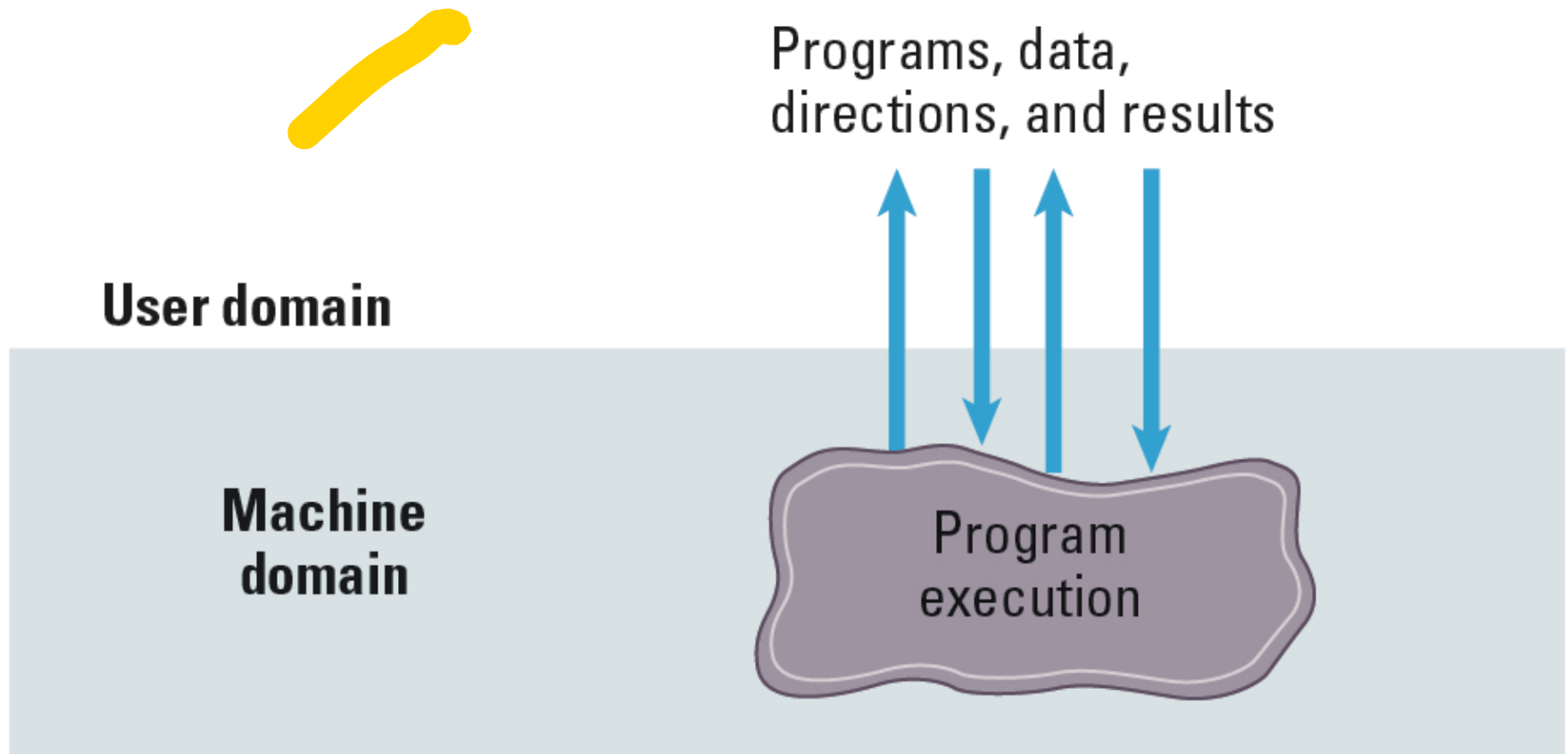
# Figure 3.1 Batch processing

# Figure 3.2 Interactive processing



Programs, data, directions, and results

User domain

Machine domain

Program execution

# 3.2 Operating System Architecture

- Application software
  - Performs specific tasks for users (productivity, games, software development)

- System software
  - Provides infrastructure for application software
  - Consists of operating system and utility software

# Figure 3.3 Software classification
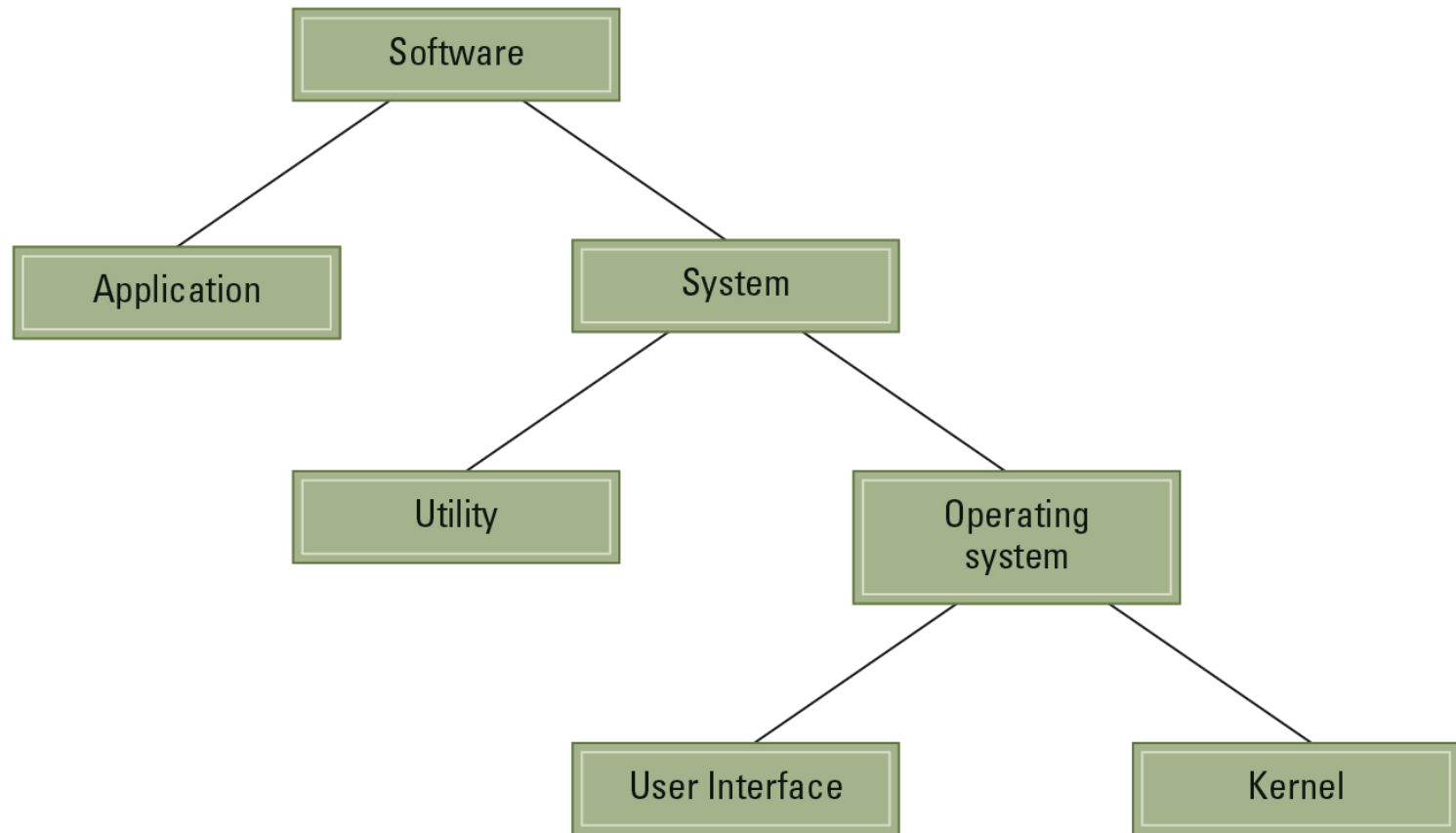
# Operating System Components

- **User Interface:** Communicates with users
    - Text based (Shell)
    - Graphical user interface (GUI)

- **Kernel:** Performs basic required functions
    - File manager
    - Device drivers
    - Memory manager
    - Scheduler and dispatcher

# Figure 3.4 The user interface acts as an intermediary between users and the operating system's kernel
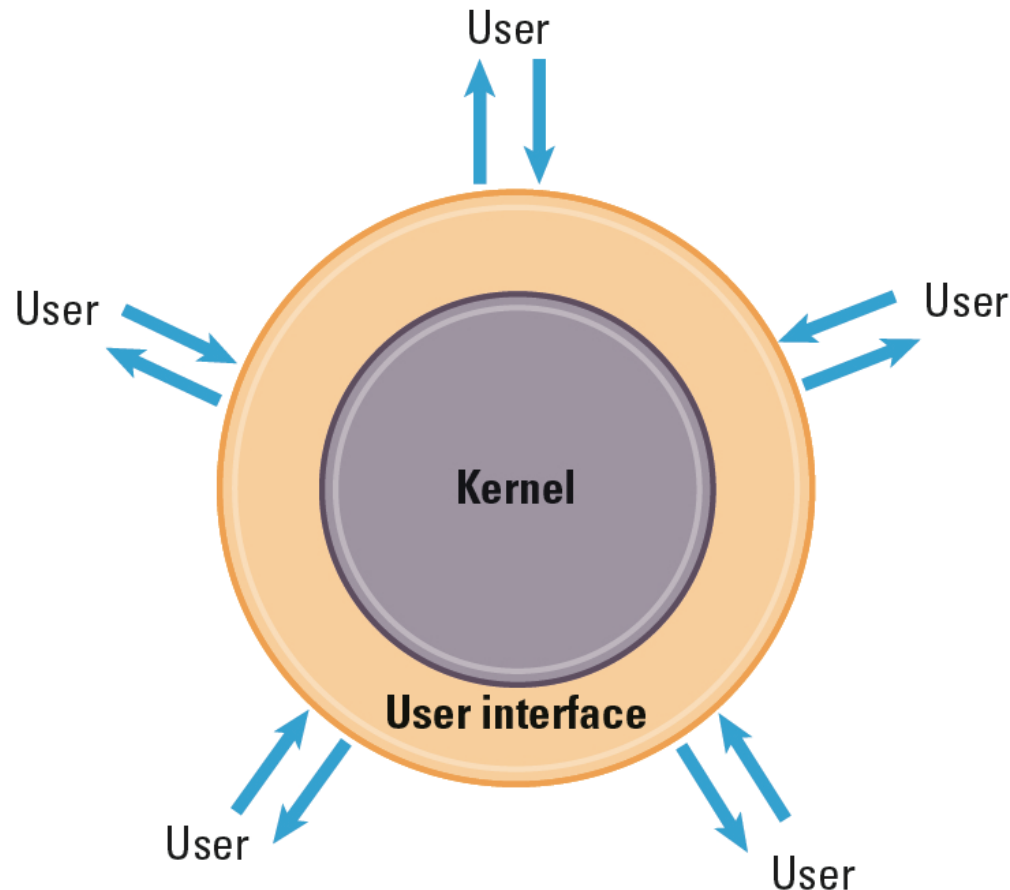
# File Manager

- **Directory** (or **Folder**): A user-created bundle of files and other directories (subdirectories)

- **Directory Path:** A sequence of directories within directories

# Memory Manager

- Allocates space in main memory

- May create the illusion that the machine has more memory than it actually does (**virtual memory**) by playing a "shell game" in which blocks of data (**pages**) are shifted back and forth between main memory and mass storage

# Getting it Started (Bootstrapping)

- **Boot loader:** Program in ROM (example of firmware)
  - Run by the CPU when power is turned on
  - Transfers operating system from mass storage to main memory
  - Executes jump to operating system

Pearson

# Figure 3.5  The booting process

**Main memory**

| ROM | Boot loader |
|---|---|
| Volatile memory | |

**Disk storage**
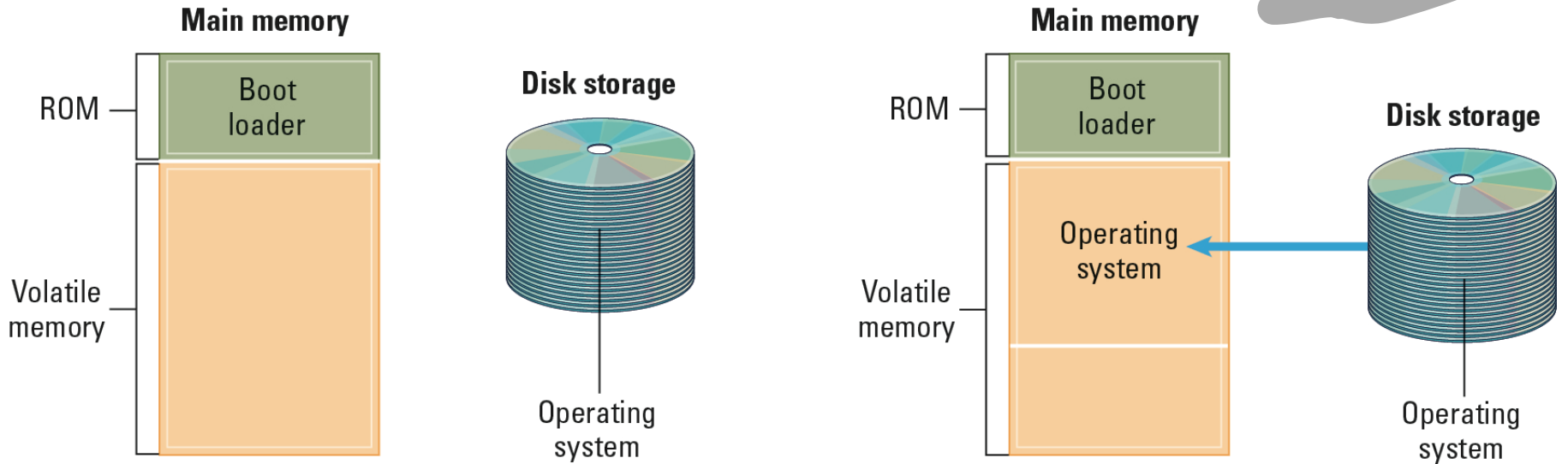
Operating system

**Step 1:** Machine starts by executing the boot loader program already in memory. Operating system is stored in mass storage.

**Main memory**

| ROM | Boot loader |
|---|---|
| Volatile memory | Operating system |

**Disk storage**

Operating system

**Step 2:** Boot loader program directs the transfer of the operating system into main memory and then transfers control to it.

# 3.3 Coordinating the Machine's Activities

An operating system coordinates the execution of application software, utility software, and units within the operating system itself.

# The Concept of a Process

- **Process:** The activity of executing a program

- **Process State:** Current status of the activity
  - Program counter
  - General purpose registers
  - Related portion of main memory

# Process Administration

- **Scheduler:** Adds new processes to the process table and removes completed processes from the process table

- **Dispatcher:** Controls the allocation of time slices to the processes in the process table
  - The end of a time slice is signaled by an interrupt.

# Figure 3.6  Multiprogramming between process A and process B
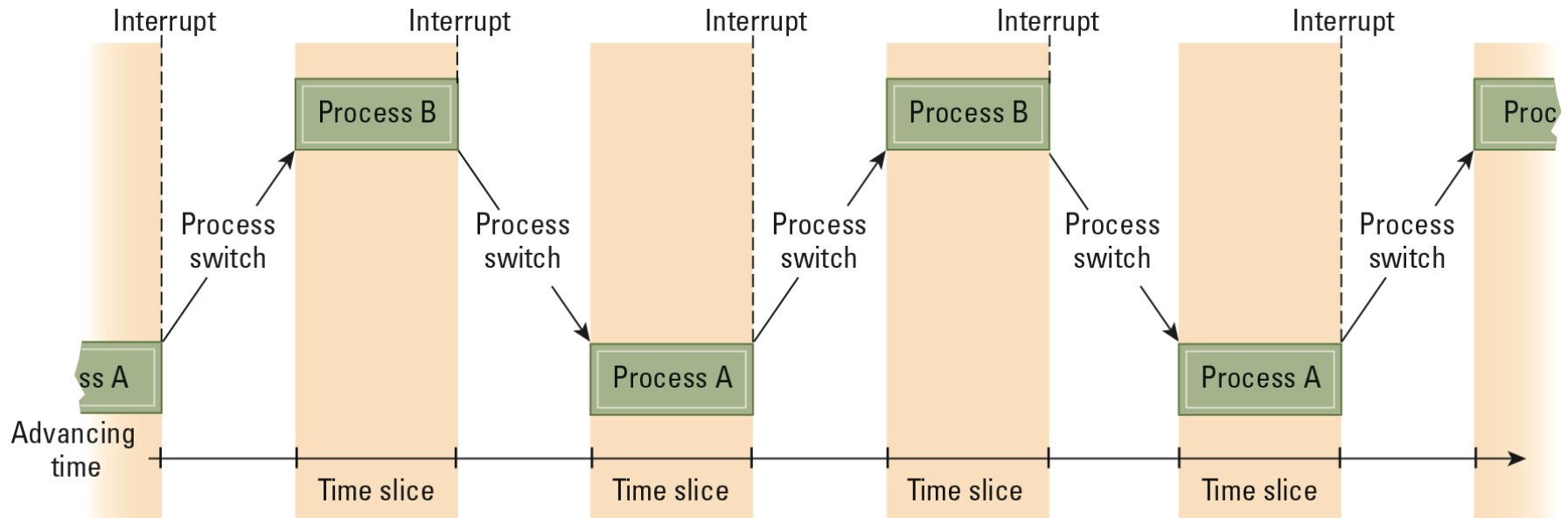
# 3.4 Handling Competition Among Processes

- **Semaphore:** A "control flag"

- **Critical Region:** A group of instructions that should be executed by only one process at a time

- **Mutual exclusion:** Requirement that only one process at a time be allowed to execute a Critical Region

# Deadlock

- Processes block each other from continuing because each is waiting for a resource that is allocated to another

- Conditions required for deadlock
  1. Competition for non-sharable resources
  2. Resources requested on a partial basis
  3. An allocated resource can not be forcibly retrieved

# Figure 3.7 A deadlock resulting from competition for nonshareable railroad intersections
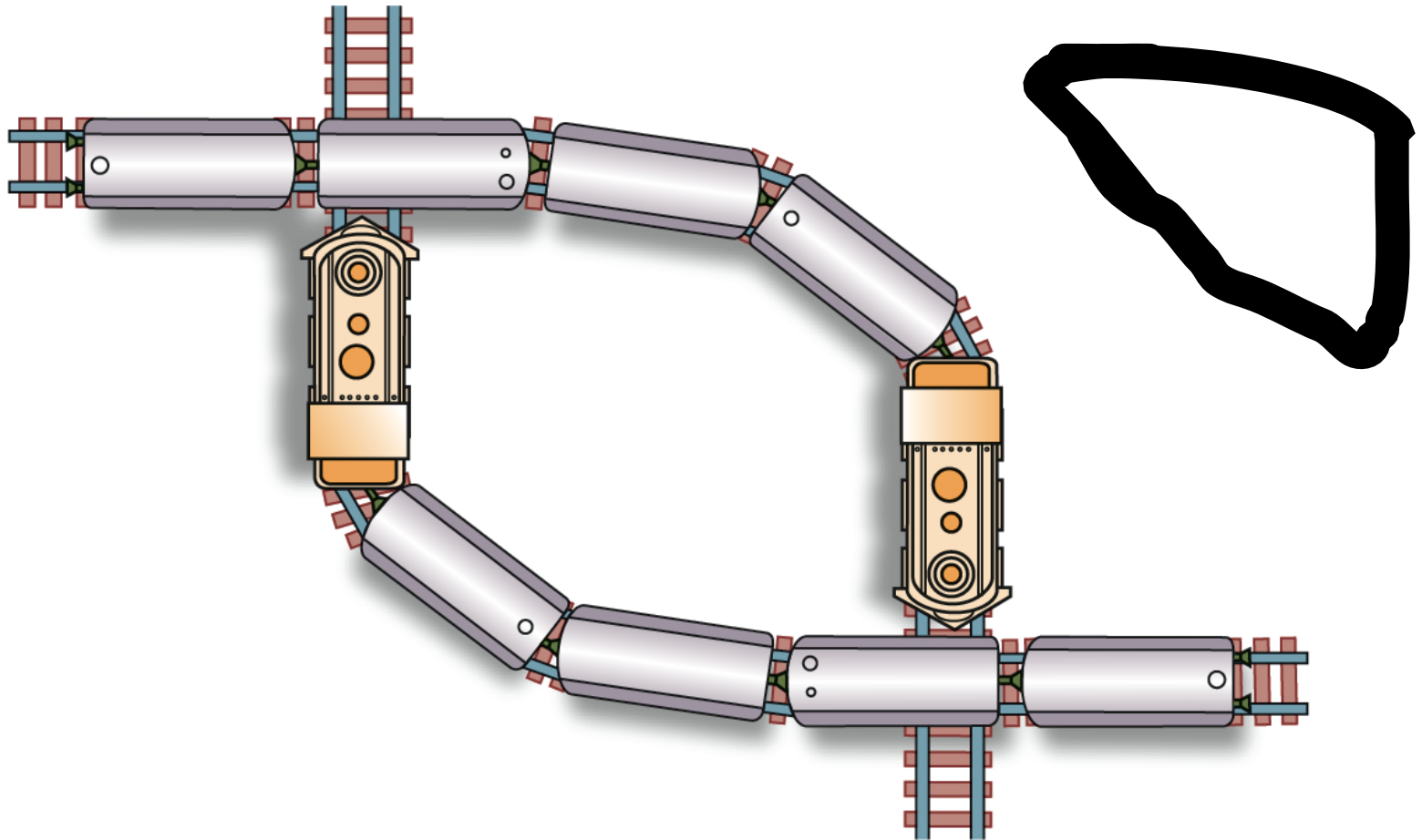
# 3.5 Security

- Attacks from outside
  - Problems
    - Insecure passwords
    - Sniffing software
  - Counter measures
    - Auditing software

# Security (continued)

- Attacks from within
  - Problem: A process that gains access to memory outside its designated area
  - Counter measures: Control process activities via privilege levels and privileged instructions