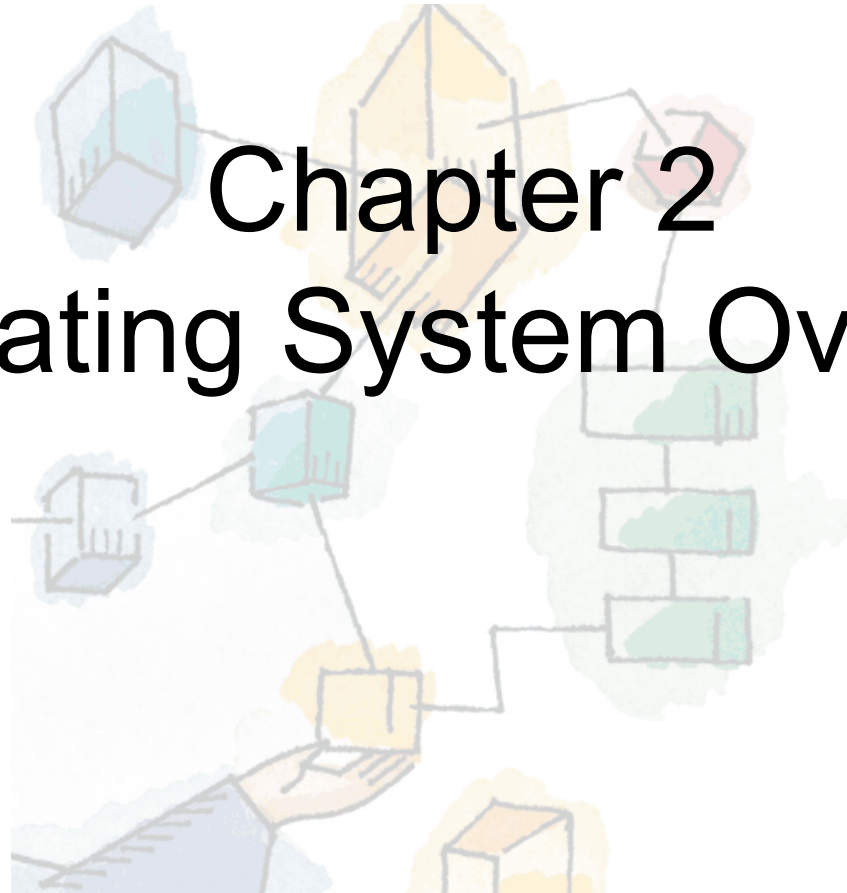
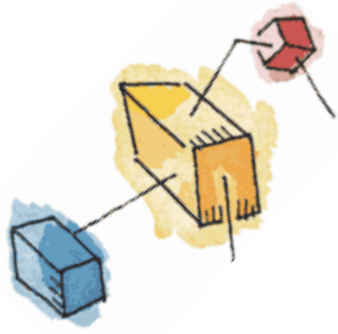


*Operating Systems:  
Internals and Design Principles, 6/E*  
William Stallings

# Chapter 2

# Operating System Overview

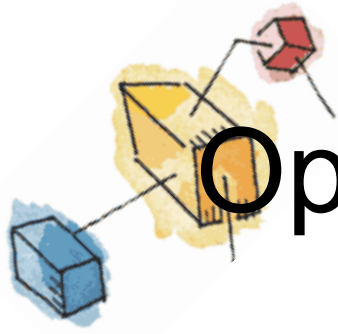




# Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware





# Operating System Objectives

- Convenience
- Efficiency
- Ability to evolve



# Layers and Views

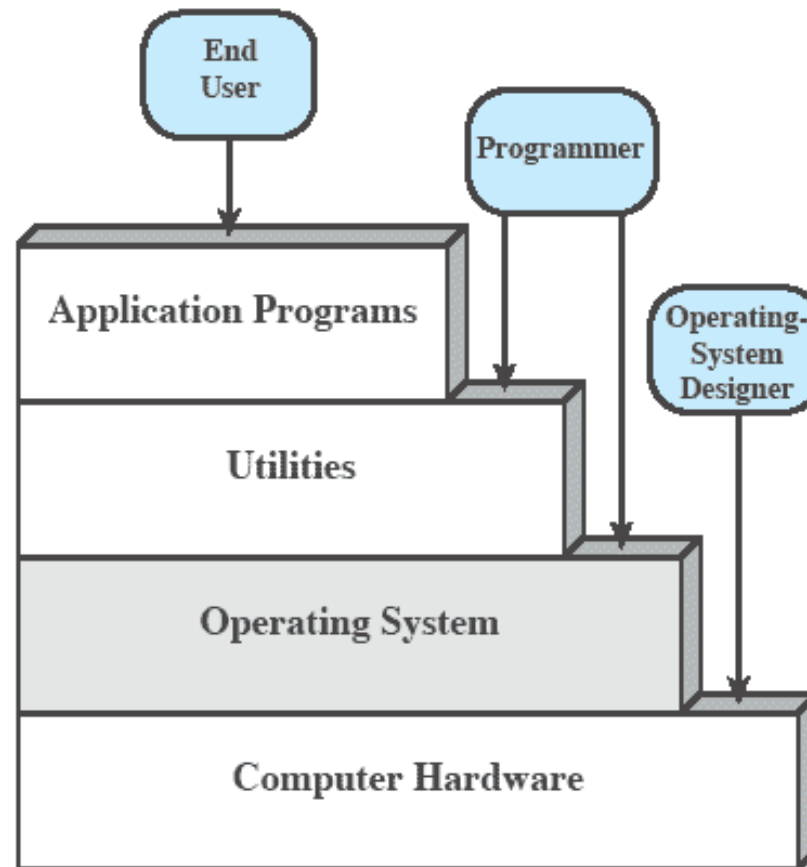
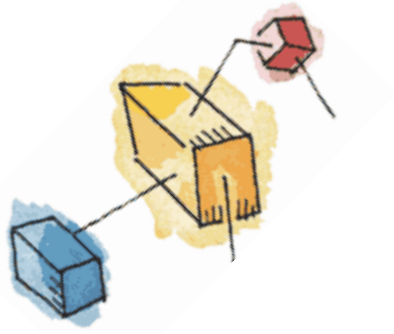
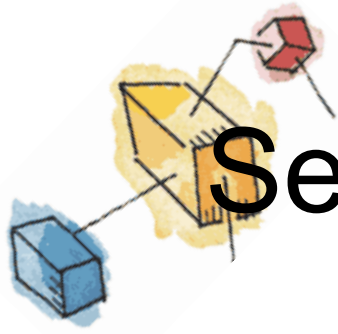


Figure 2.1 Layers and Views of a Computer System

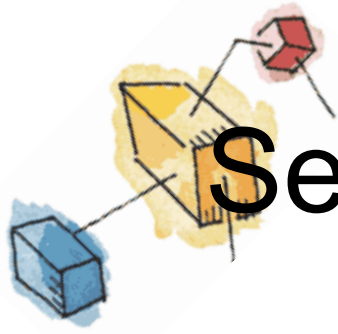




# Services Provided by the OS

- Program development
  - Editors and debuggers
- Program execution
- Access I/O devices

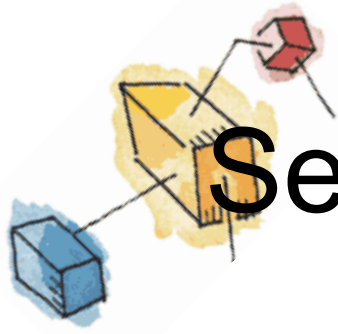




# Services Provided by the OS

- Controlled access to files
- System access

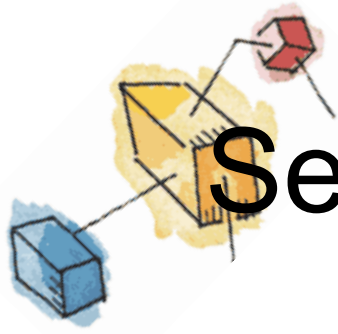




# Services Provided by the OS

- Error detection and response
  - Internal and external hardware errors
  - Software errors
  - Operating system cannot grant request of application

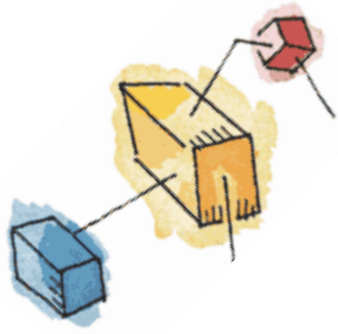




# Services Provided by the OS

- Accounting
  - Collect usage statistics
  - Monitor performance
  - Used to anticipate future enhancements
  - Used for billing purposes

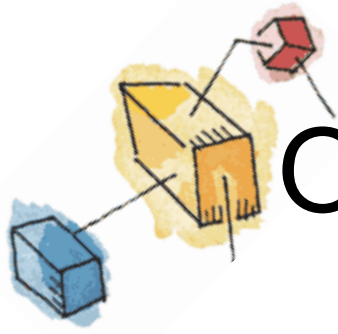




# Operating System

- Responsible for managing resources
- Functions same way as ordinary computer software
  - It is a program that is executed
- Operating system relinquishes control of the processor





# OS as Resource Manager

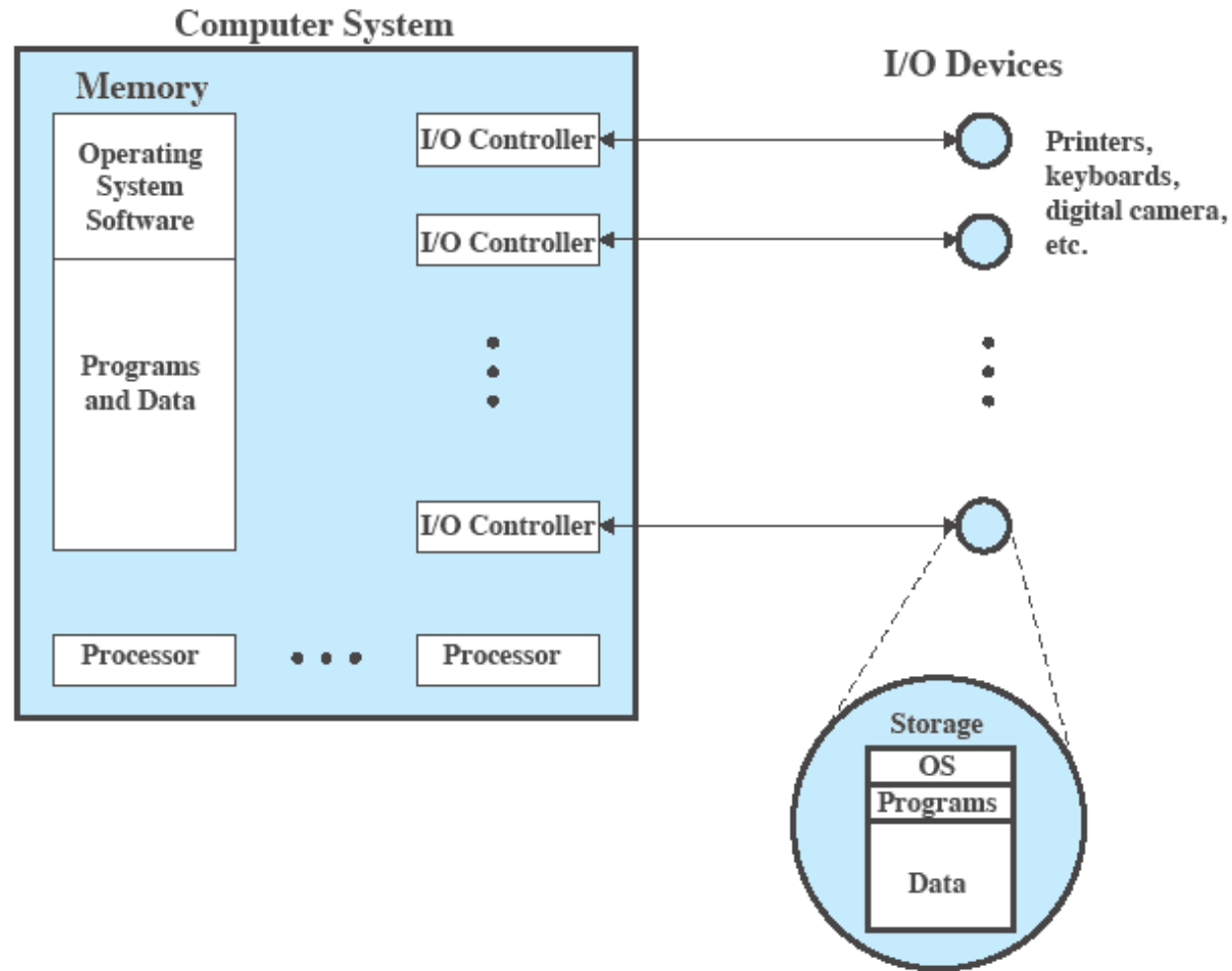
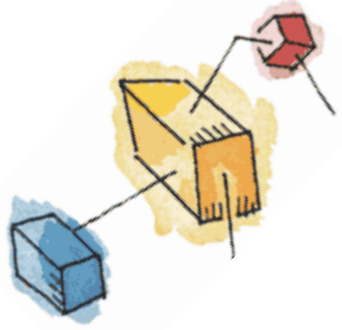


Figure 2.2 The Operating System as Resource Manager

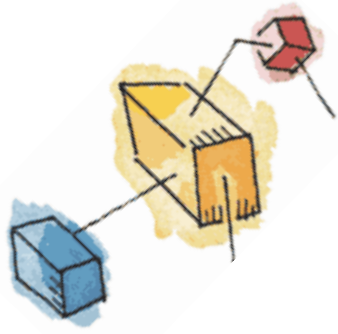


# Kernel



- Portion of operating system that is in main memory
- Contains most frequently used functions
- Also called the nucleus

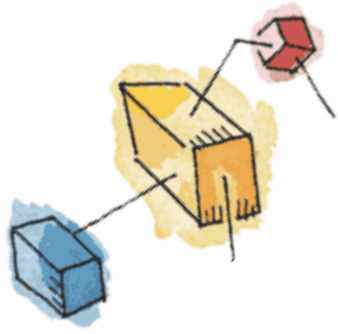




# Evolution of Operating Systems

- Hardware upgrades plus new types of hardware
- New services
- Fixes

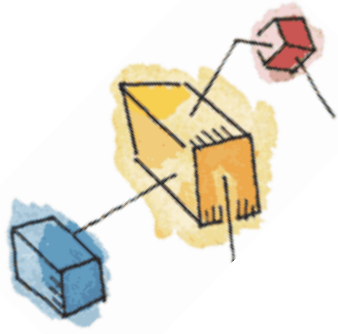




# Evolution of Operating Systems

- Serial processing
  - No operating system
  - Machines run from a console with display lights, toggle switches, input device, and printer

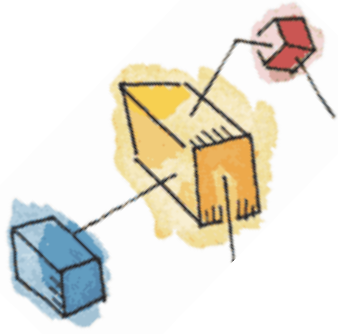




# Evolution of Operating Systems

- Serial processing
  - Schedule time
  - Setup included loading the compiler, source program, saving compiled program, and loading and linking





# Evolution of Operating Systems

- Simple batch system
  - Monitor
    - Software that controls the sequence of events
    - Batch jobs together
    - Program returns control to monitor when finished



# Resident Monitor

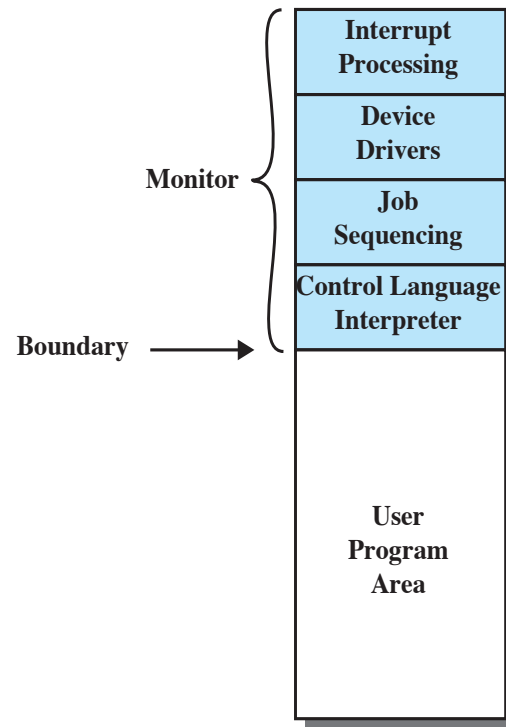
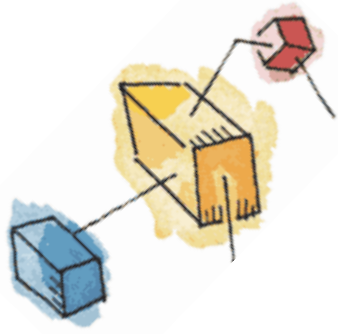


Figure 2.3 Memory Layout for a Resident Monitor





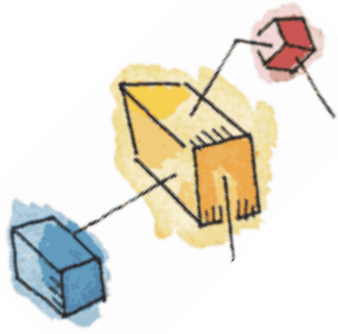
# Job Control Language

- Special type of programming language
- Provides instruction to the monitor
  - What compiler to use
  - What data to use

## **JCL:** Job Control Language

```
$JOB          $RUN
$FTN          ...
...          $END
$LOAD
```

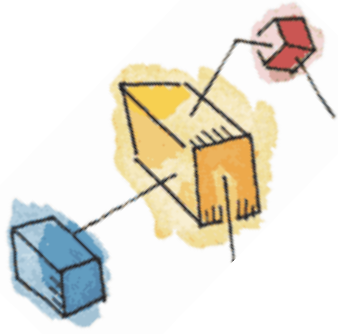




# Hardware Features

- Memory protection
  - Do not allow the memory area containing the monitor to be altered
- Timer
  - Prevents a job from monopolizing the system

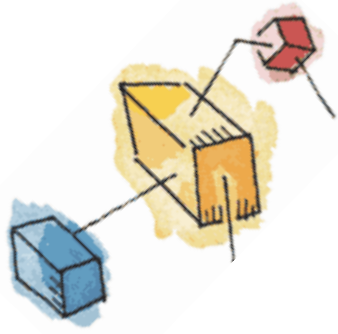




# Hardware Features

- Privileged instructions
  - Certain machine level instructions can only be executed by the monitor
- Interrupts
  - Early computer models did not have this capability

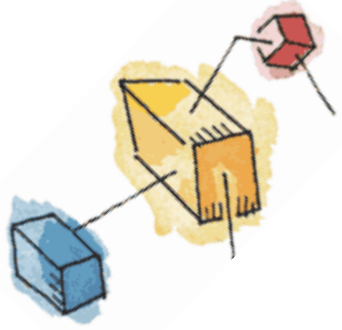




# Memory Protection

- User program executes in user mode
  - Certain instructions may not be executed

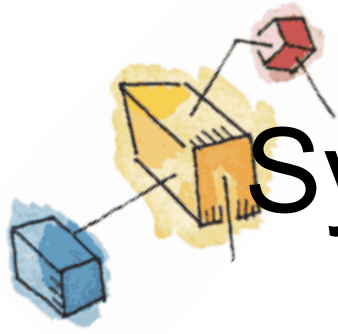




# Memory Protection

- Monitor executes in system mode
  - Kernel mode
  - Privileged instructions are executed
  - Protected areas of memory may be accessed





# System Utilization Example

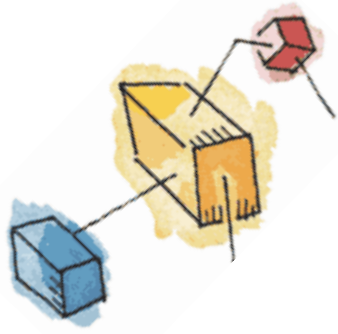
Read one record from file	15 $\mu$ s
Execute 100 instructions	1 $\mu$ s
Write one record to file	<u>15 <math>\mu</math>s</u>
TOTAL	31 $\mu$ s

Percent CPU Utilization	$= \frac{1}{31} = 0.032 = 3.2\%$
-------------------------	----------------------------------

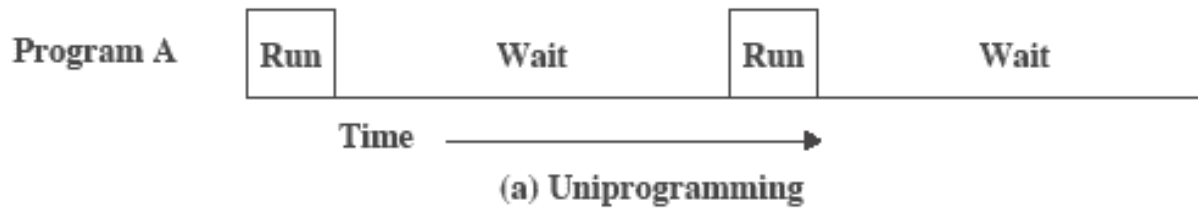
**Figure 2.4 System Utilization Example**

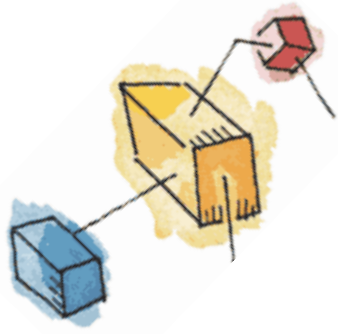




# Uniprogramming

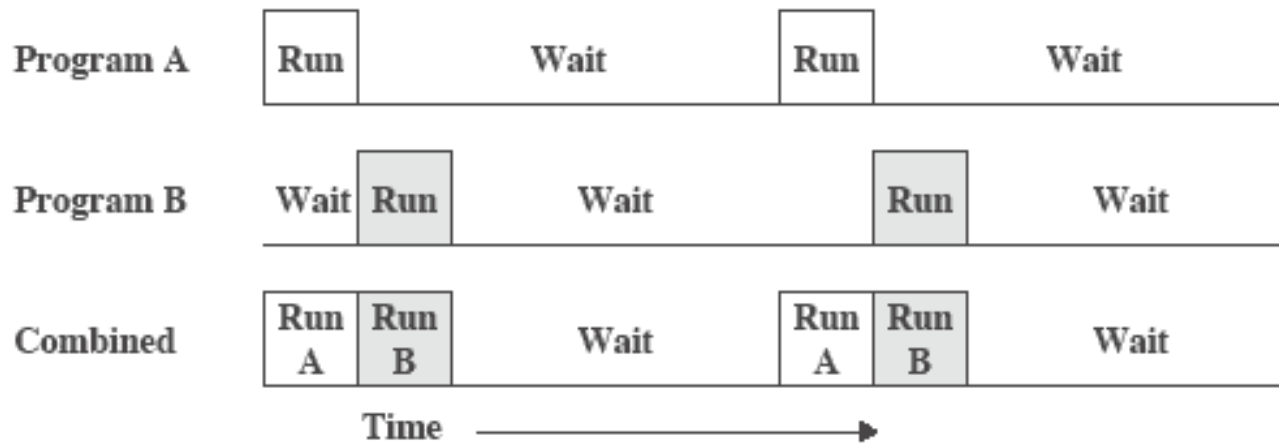
- Processor must wait for I/O instruction to complete before proceeding





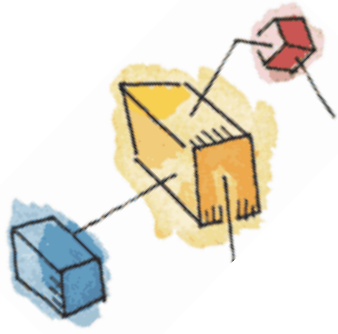
# Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job

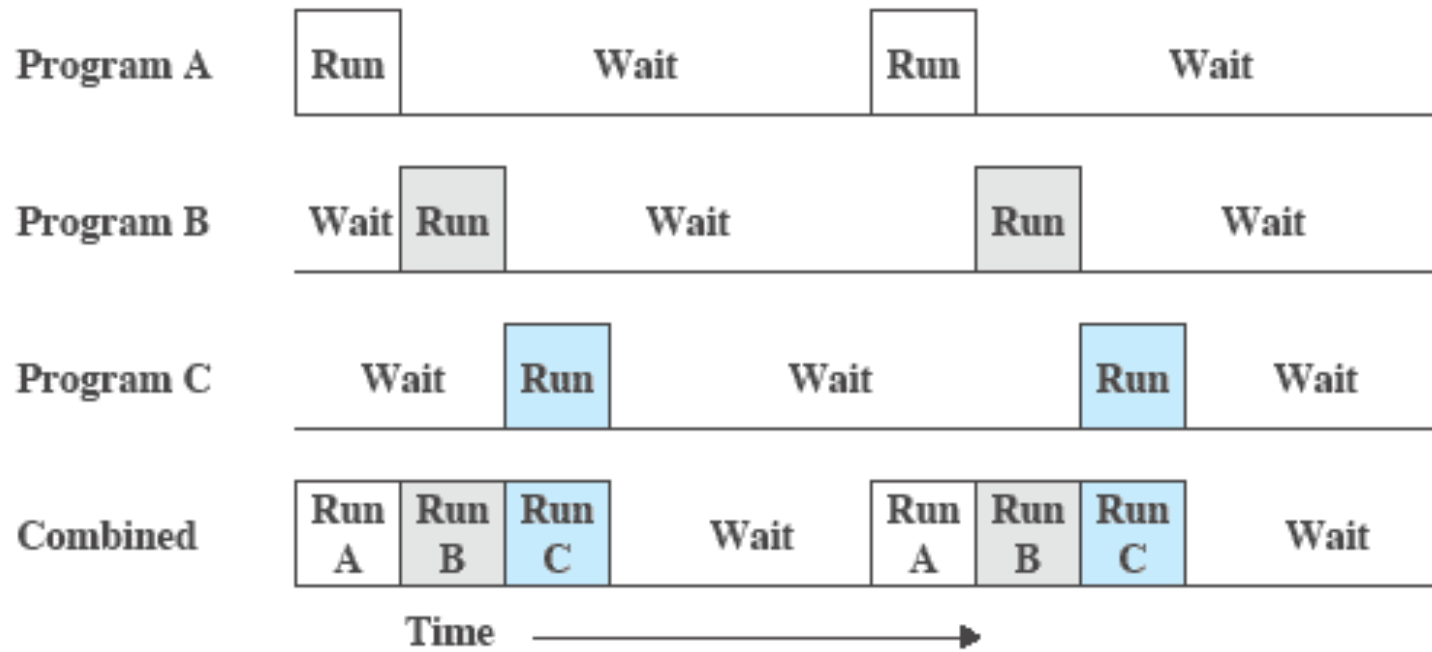


(b) Multiprogramming with two programs





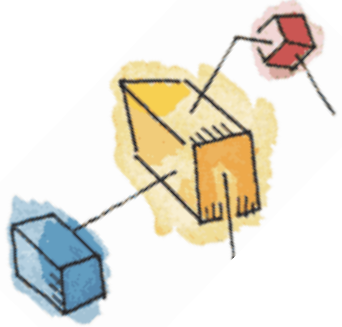
# Multiprogramming



(c) Multiprogramming with three programs



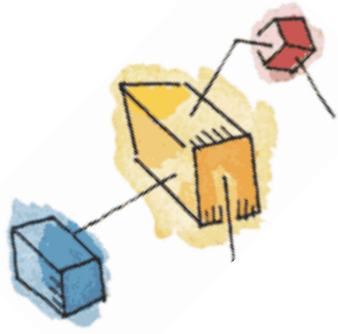
# Example



**Table 2.1 Sample Program Execution Attributes**

	<b>JOB1</b>	<b>JOB2</b>	<b>JOB3</b>
<b>Type of job</b>	Heavy compute	Heavy I/O	Heavy I/O
<b>Duration</b>	5 min	15 min	10 min
<b>Memory required</b>	50 M	100 M	75 M
<b>Need disk?</b>	No	No	Yes
<b>Need terminal?</b>	No	Yes	No
<b>Need printer?</b>	No	No	Yes





# Utilization Histograms

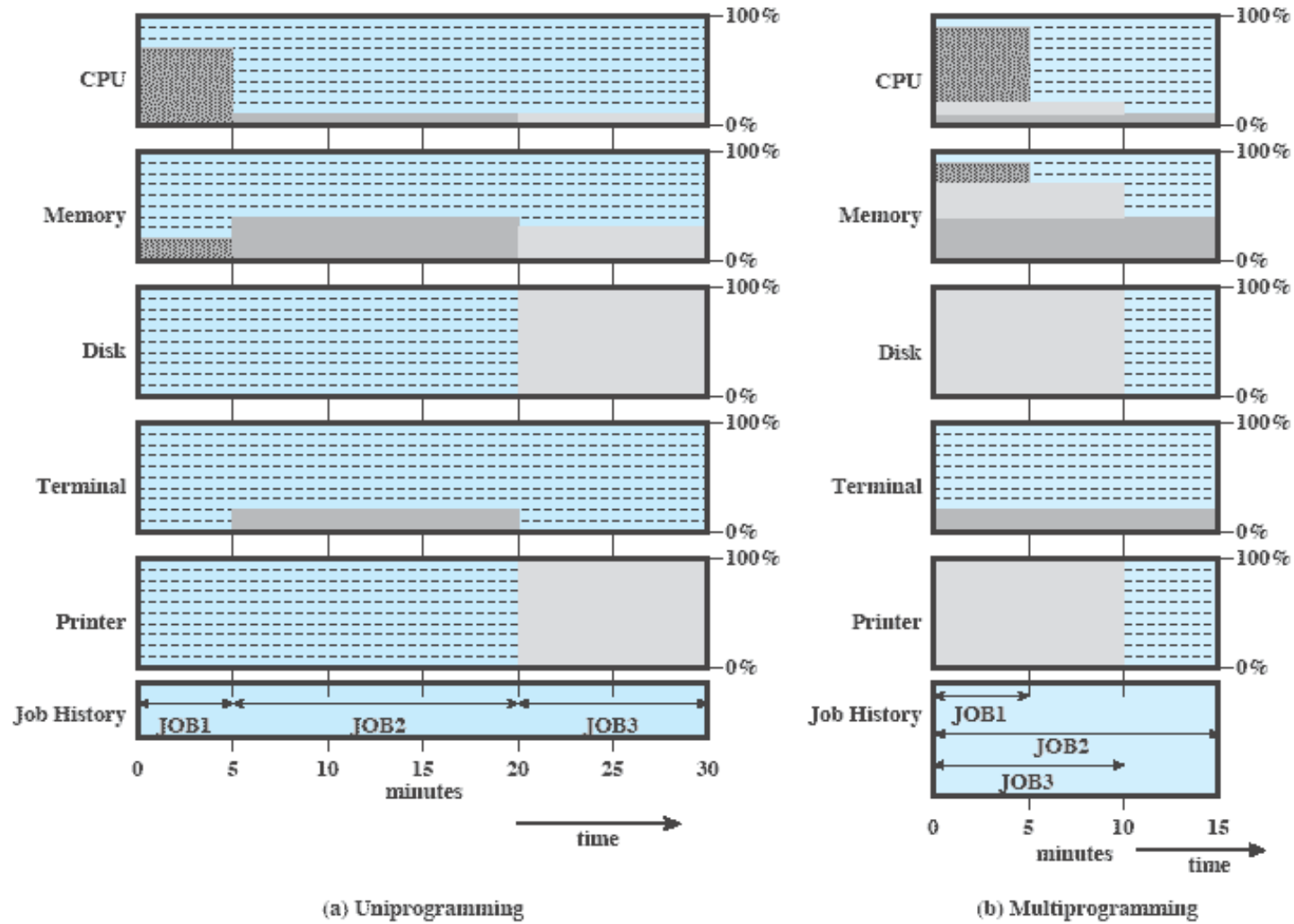
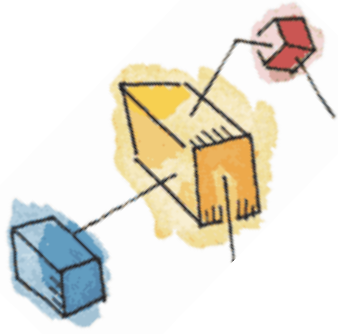


Figure 2.6 Utilization Histograms

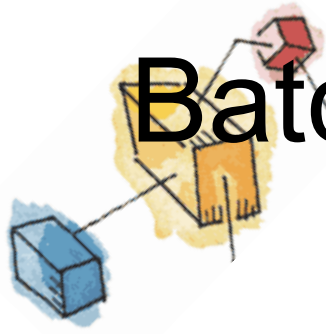




# Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals





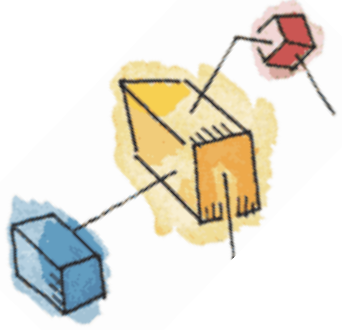
# Batch Multiprogramming versus Time Sharing

**Table 2.3 Batch Multiprogramming versus Time Sharing**

	<b>Batch Multiprogramming</b>	<b>Time Sharing</b>
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal



# Example CTSS



- Job1: 15K words
- Job2: 20K words
- Job3: 5K words
- Job4: 10K words



# CTSS Operation

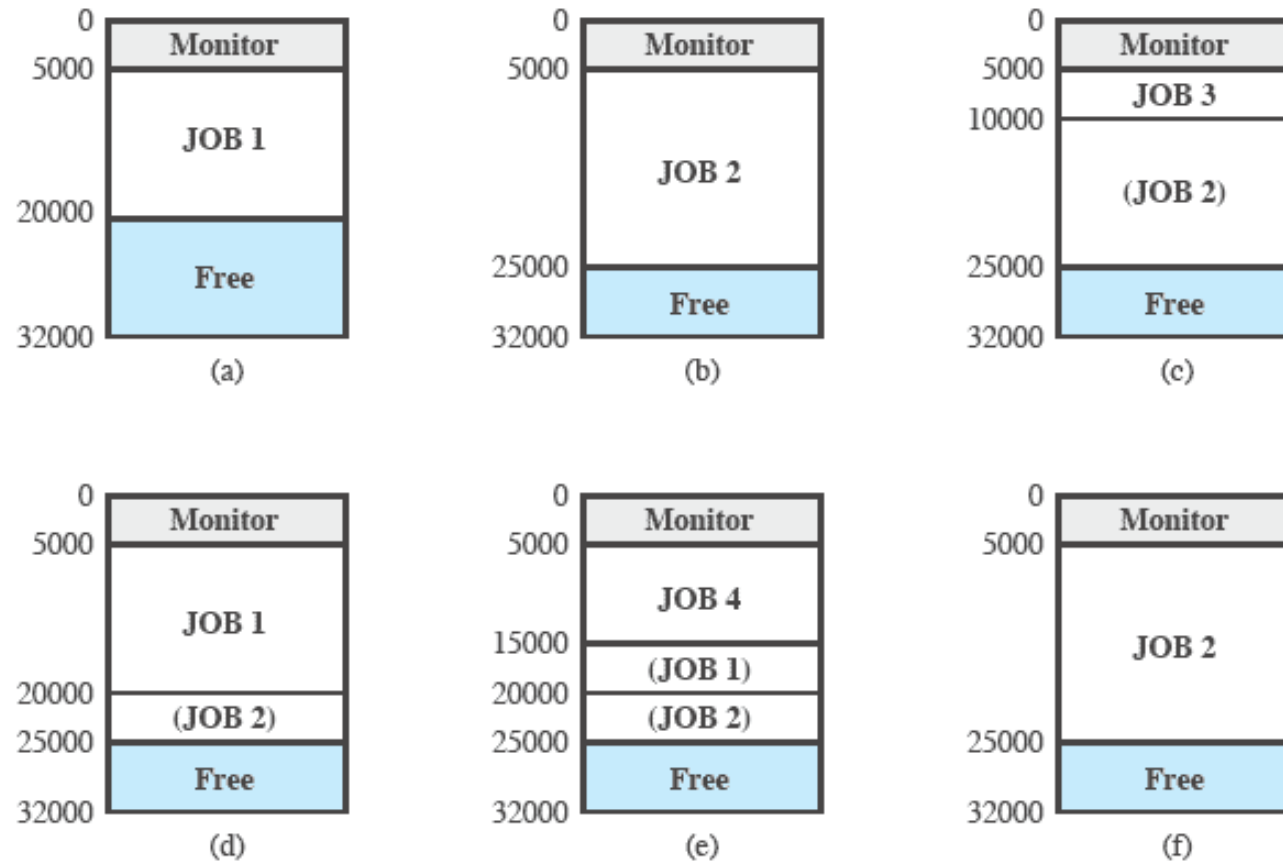
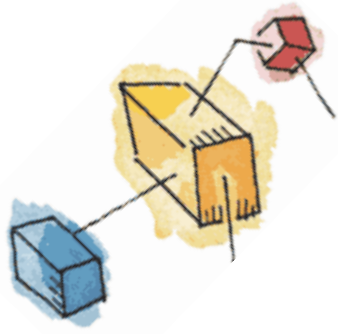
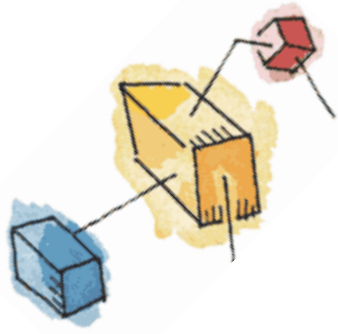


Figure 2.7 CTSS Operation

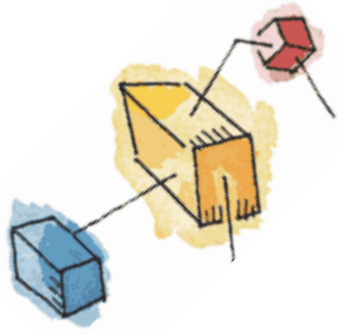




# Major Achievements

- Processes
- Memory management
- Information protection and security
- Scheduling and resource management
- System structure



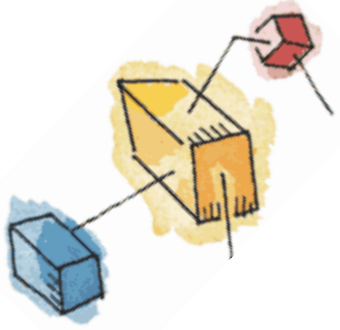


# Process

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor

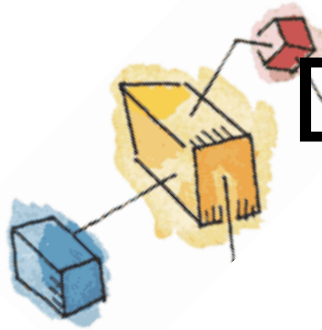


# Process



- A unit of activity characterized by
  - A single sequential thread of execution
  - A current state
  - An associated set of system resources



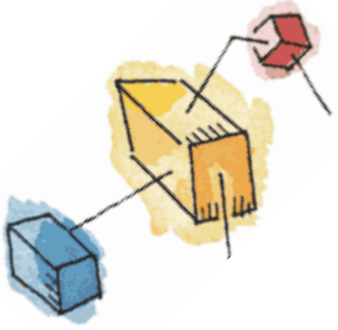


# Difficulties with Designing System Software

- Improper synchronization
- Failed mutual exclusion
- Nondeterminate program operation
- Deadlocks



# Process



- Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process



# Process

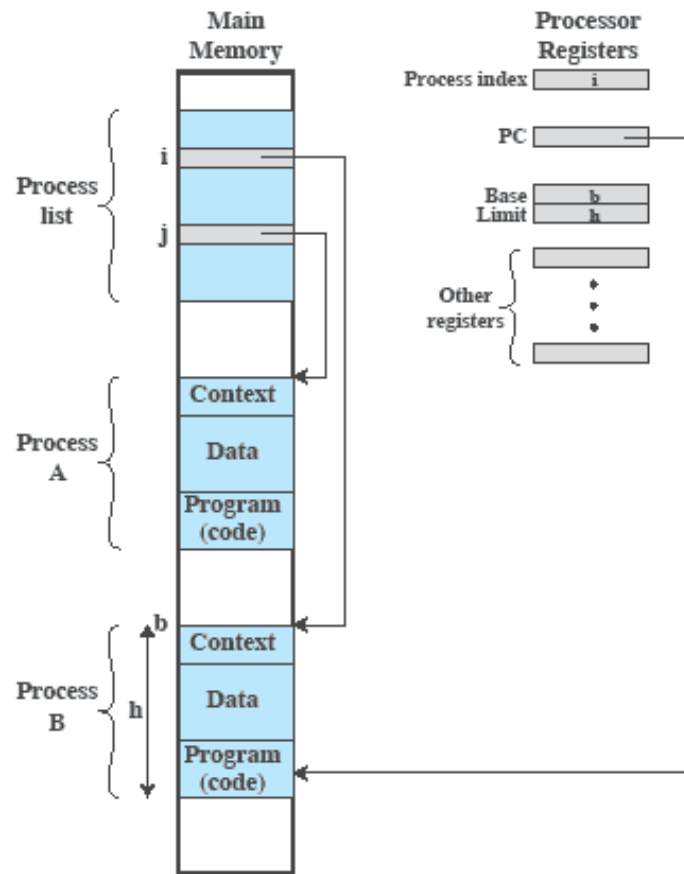
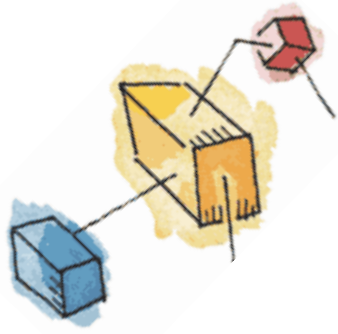


Figure 2.8 Typical Process Implementation

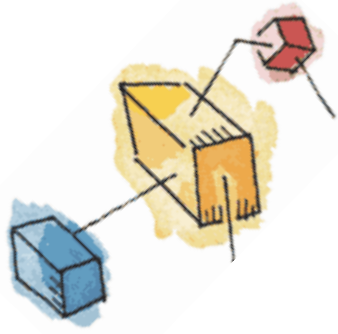




# Memory Management

- Process isolation
- Automatic allocation and management
- Support of modular programming
- Protection and access control
- Long-term storage

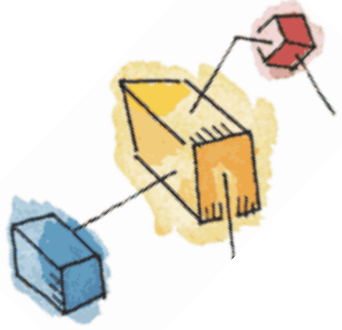




# Virtual Memory

- Implements long-term store
- Information stored in named objects called files
- Allows programmers to address memory from a logical point of view



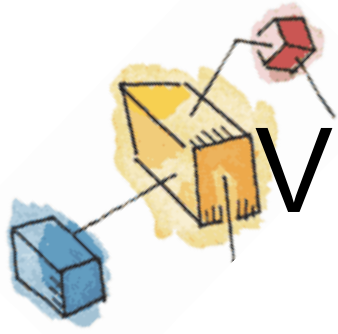


# Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages
- Virtual address is a page number and an offset within the page
- Each page may be located any where in main memory
- Real address or physical address in main memory







# Virtual Memory Addressing

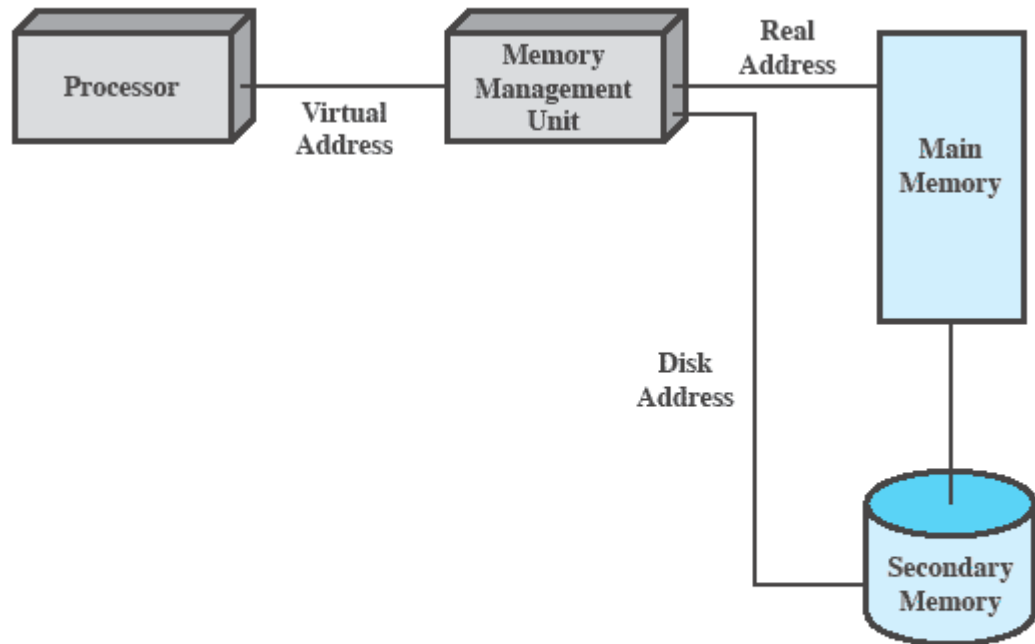


Figure 2.10 Virtual Memory Addressing

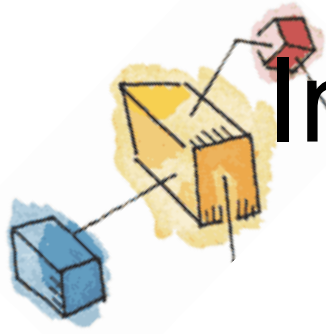




# Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized





# Information Protection and Security

- Data integrity
  - Protection of data from unauthorized modification
- Authenticity
  - Concerned with the proper verification of the identity of users and the validity of messages or data





# Scheduling and Resource Management

- Fairness
  - Give equal and fair access to resources
- Differential responsiveness
  - Discriminate among different classes of jobs

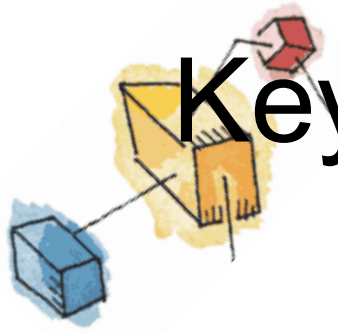




# Scheduling and Resource Management

- Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible





# Key Elements of an Operating System

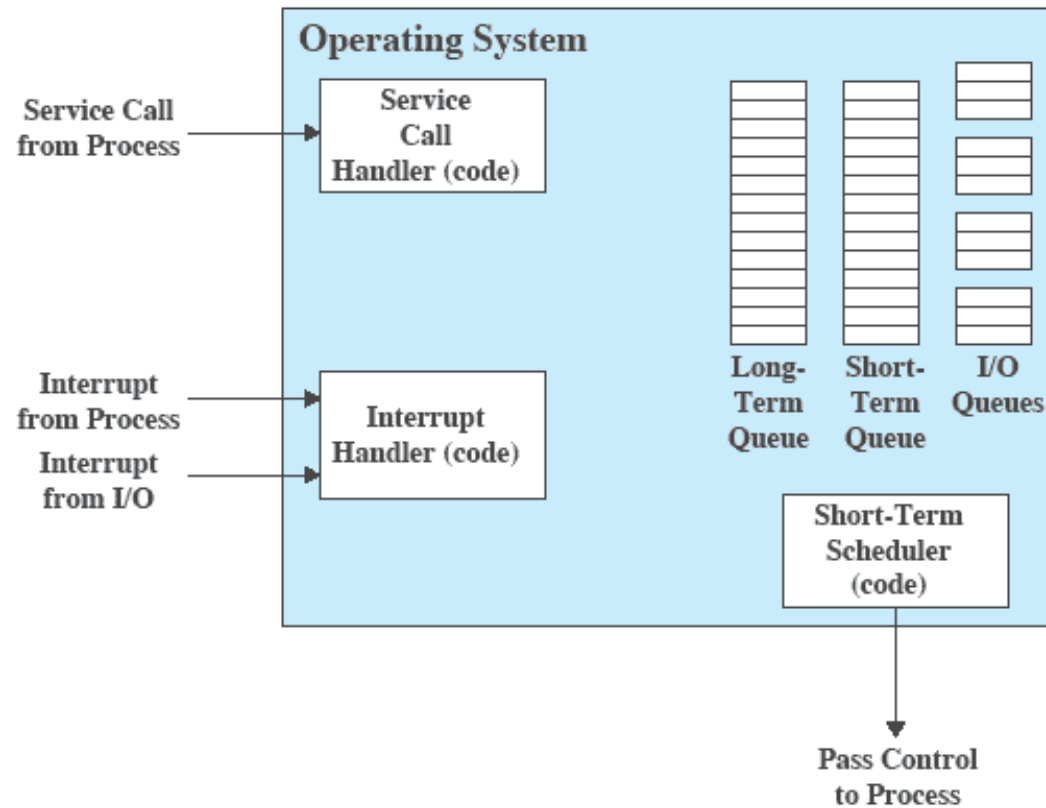
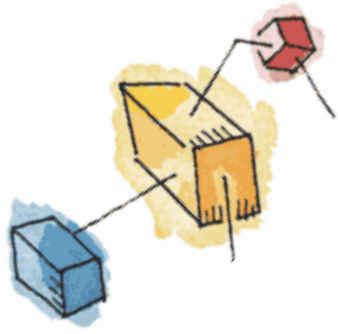


Figure 2.11 Key Elements of an Operating System for Multiprogramming



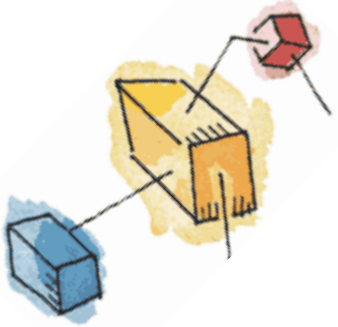


# System Structure

- View the system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems



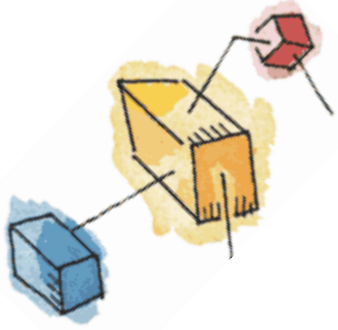
# Levels



- Level 1
  - Electronic circuits
  - Objects are registers, memory cells, and logic gates
  - Operations are clearing a register or reading a memory location
- Level 2
  - Processor's instruction set
  - Operations such as add, subtract, load, and store

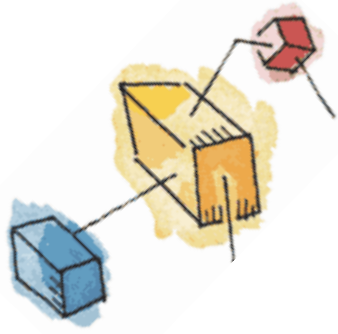


# Levels



- Level 3
  - Adds the concept of a procedure or subroutine, plus call/return operations
- Level 4
  - Interrupts



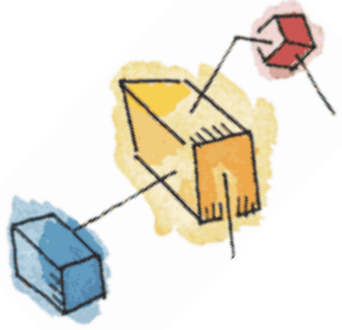


# Concepts with Multiprogramming

- Level 5
  - Process as a program in execution
  - Suspend and resume processes
- Level 6
  - Secondary storage devices
  - Transfer of blocks of data

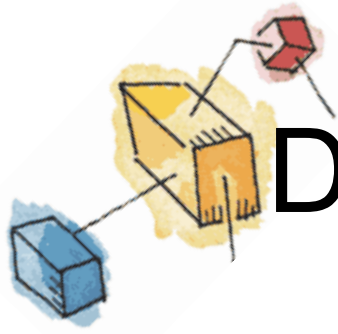


# Concepts with Multiprogramming



- Level 7
  - Creates logical address space for processes
  - Organizes virtual address space into blocks

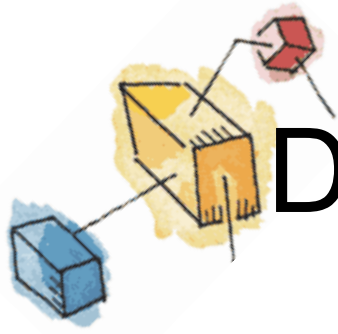




# Deal with External Objects

- Level 8
  - Communication of information and messages between processes
- Level 9
  - Supports long-term storage of named files
- Level 10
  - Provides access to external devices using standardized interfaces

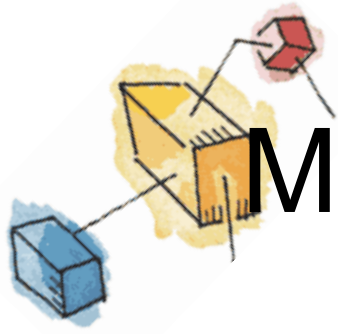




# Deal with External Objects

- Level 11
  - Responsible for maintaining the association between the external and internal identifiers
- Level 12
  - Provides full-featured facility for the support of processes
- Level 13
  - Provides an interface to the OS for the user

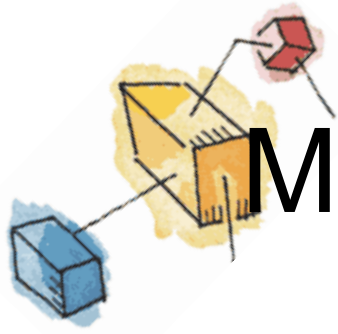




# Modern Operating Systems

- Microkernel architecture
  - Assigns only a few essential functions to the kernel
    - Address spaces
    - Interprocess communication (IPC)
    - Basic scheduling

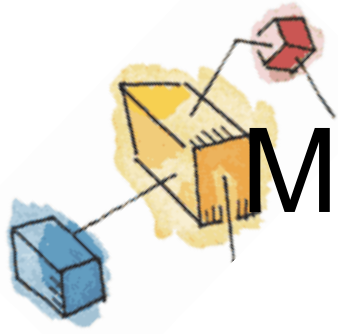




# Modern Operating Systems

- Multithreading
  - Process is divided into threads that can run concurrently
    - Thread
      - Dispatchable unit of work
      - executes sequentially and is interruptable
    - Process is a collection of one or more threads

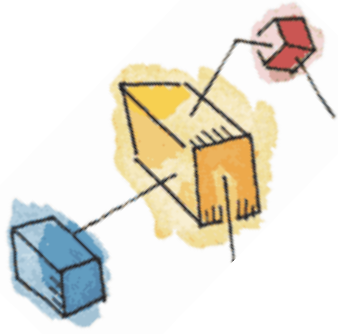




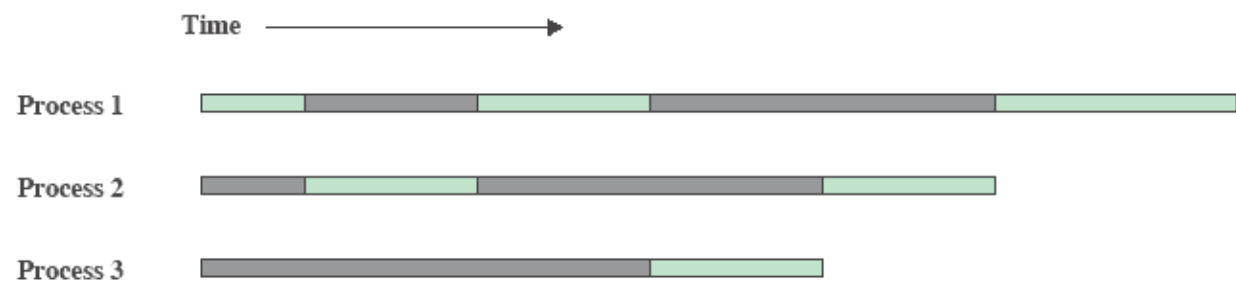
# Modern Operating Systems

- Symmetric multiprocessing (SMP)
  - There are multiple processors
  - These processors share same main memory and I/O facilities
  - All processors can perform the same functions

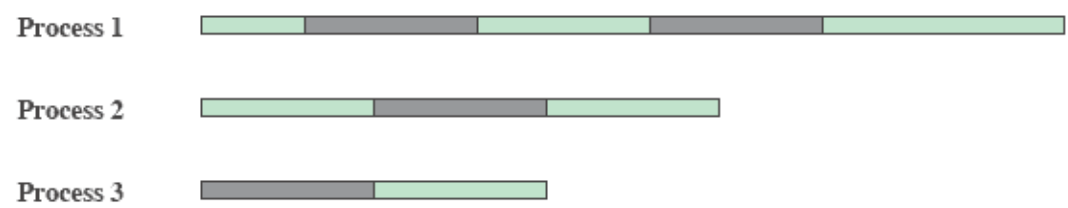




# Multiprogramming and Multiprocessing



(a) Interleaving (multiprogramming, one processor)



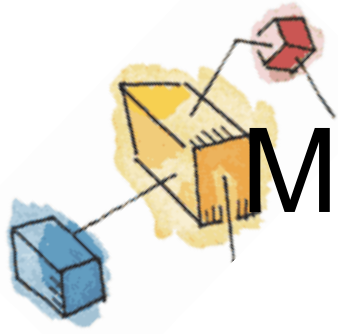
(b) Interleaving and overlapping (multiprocessing; two processors)

Blocked Running



Figure 2.12 Multiprogramming and Multiprocessing

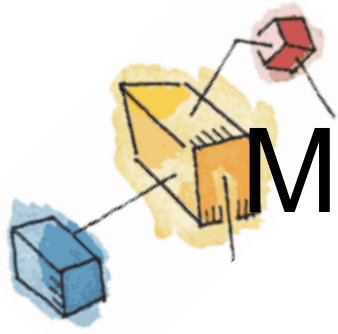




# Modern Operating Systems

- Distributed operating systems
  - Provides the illusion of a single main memory space and single secondary memory space





# Modern Operating Systems

- Object-oriented design
  - Used for adding modular extensions to a small kernel
  - Enables programmers to customize an operating system without disrupting system integrity

