Operating Systems: Internals and Design Principles

Chapter 2 Operating System Overview

Seventh Edition
By William Stallings

Operating Systems: Internals and Design Principles

Operating systems are those programs that interface the machine with the applications programs. The main function of these systems is to dynamically allocate the shared system resources to the executing programs. As such, research in this area is clearly concerned with the management and scheduling of memory, processes, and other devices. But the interface with adjacent levels continues to shift with time. Functions that were originally part of the operating system have migrated to the hardware. On the other side, programmed functions extraneous to the problems being solved by the application programs are included in the operating system.

—WHAT CAN BE AUTOMATED?: THE COMPUTER SCIENCE AND ENGINEERING RESEARCH STUDY, MIT Press, 1980

Operating System

- An interface between applications and computer
- A program that controls the execution of application programs and the allocation of system resources

Main objectives of an OS:

- Convenience
- Efficiency
- Ability to evolve

a User/Computer Interface

- The OS provides abstractions of the computer hardware, making it more convenient for applications to use the computer’s capabilities
- It does this through a set of interfaces and services.

Computer Hardware and Software Infrastructure

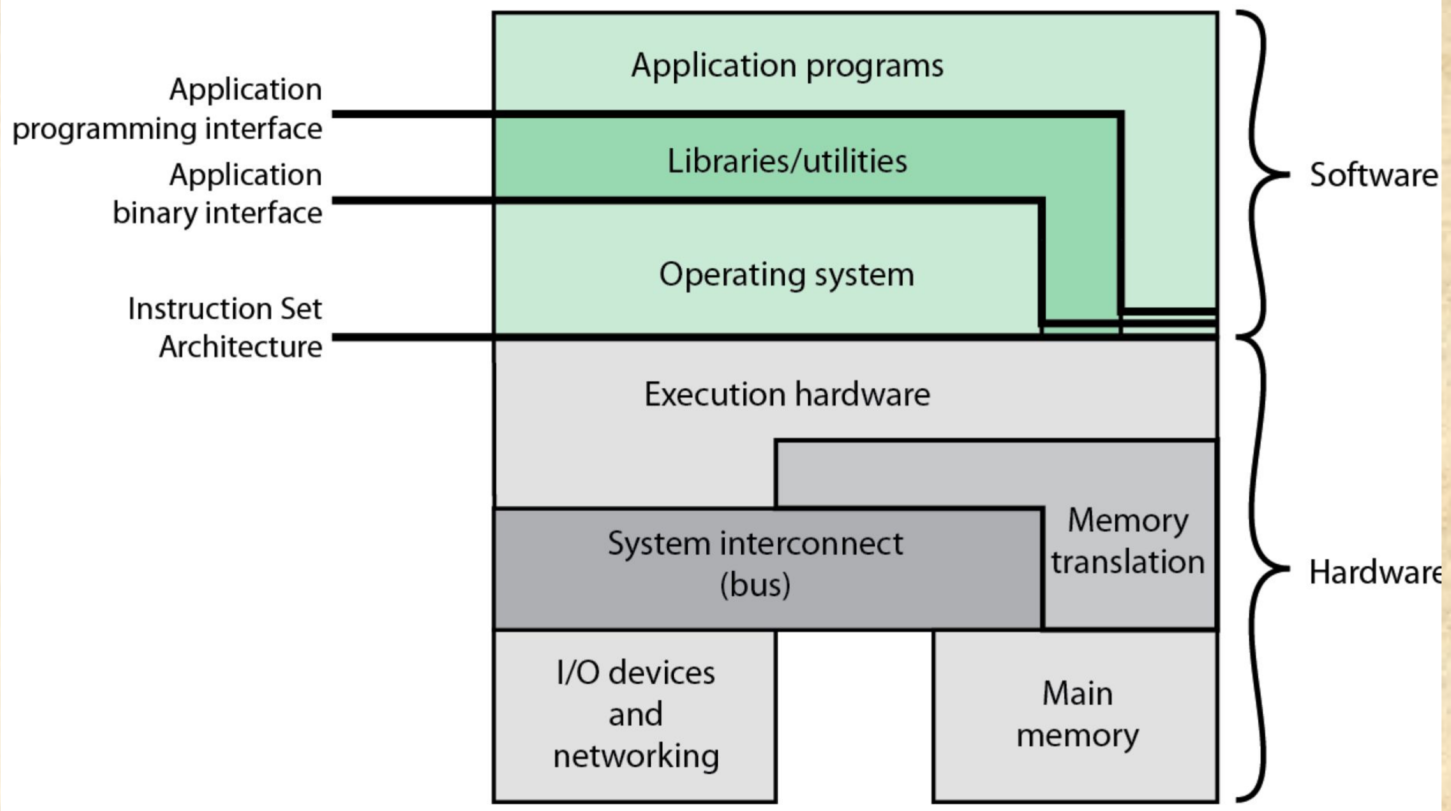

Application programming interface
Application binary interface
Instruction Set Architecture
Application programs
Libraries/utilities
Operating system
Execution hardware
Memory translation
System interconnect (bus)
I/O devices and networking
Main memory
Software
Hardware

Figure 2.1 Computer Hardware and Software Infrastructure

Key Interfaces

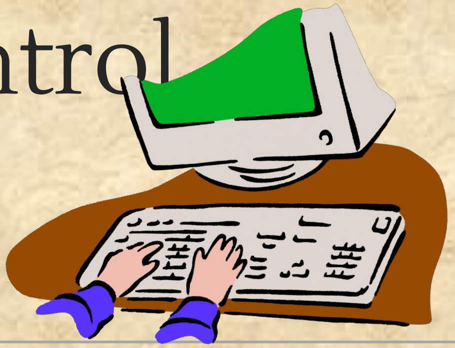
- Instruction set architecture (ISA)
- Application binary interface (ABI)
  supports portability of applications in binary forms across different system platforms and environments
- Application programming interface (API) applications can interface to OS through system calls

Operating System Services

- Program development utilities – not strictly OS
- Program execution
- Access I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

Efficiency:The Operating System As a Resource Manager

- A computer is a set of resources for moving, storing, & processing data
- The OS is responsible for managing these resources
- The OS exercises its control through software



Operating System as Software

- Functions in the same way as ordinary computer software
- Program, or suite of programs, executed by the processor
- Frequently relinquishes control and must be able to regain control to decide on the next thing the processor should do.

Operating System as Resource Manager

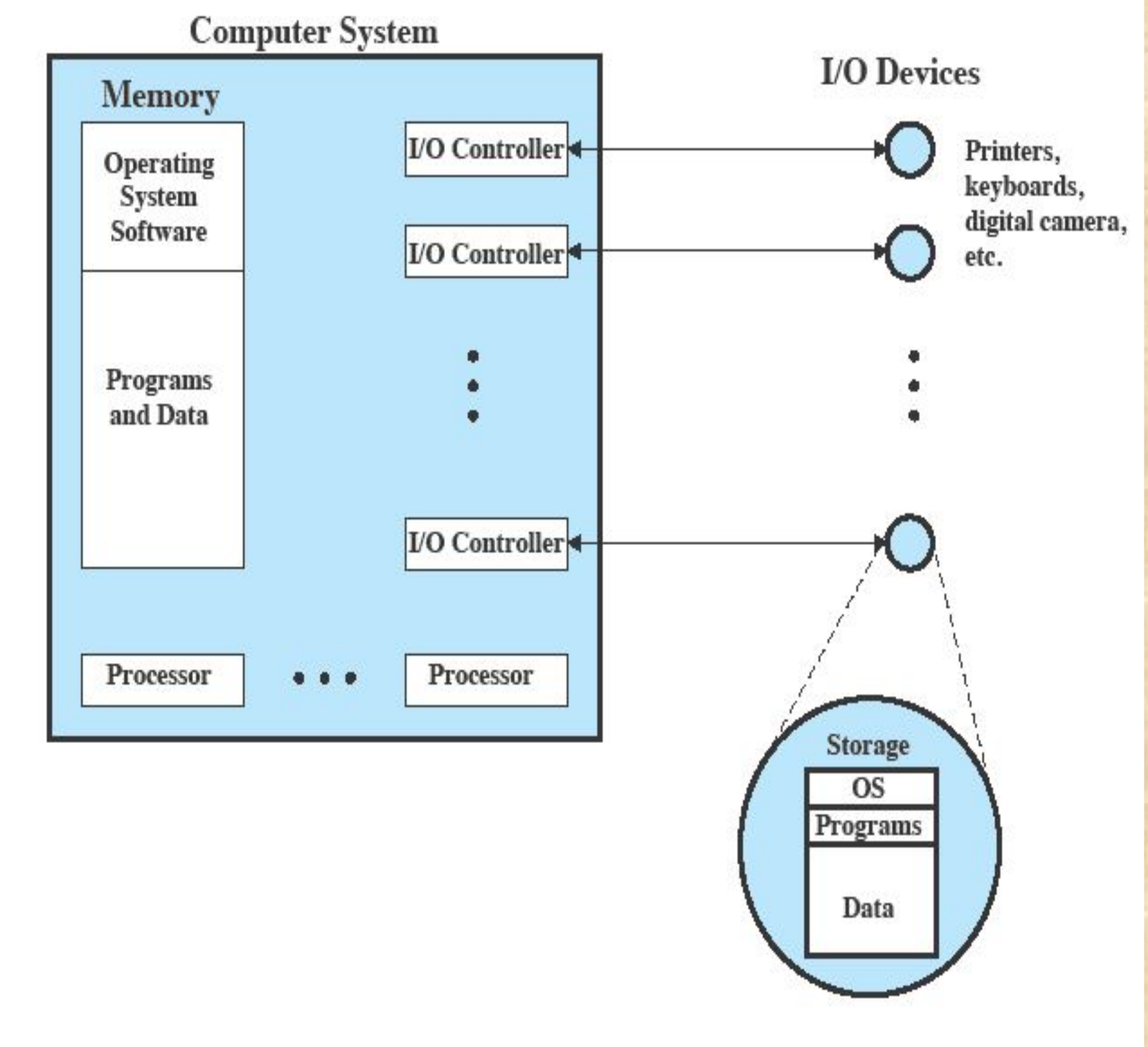


Figure 2.2 The Operating System as Resource Manager

Evolution of Operating Systems

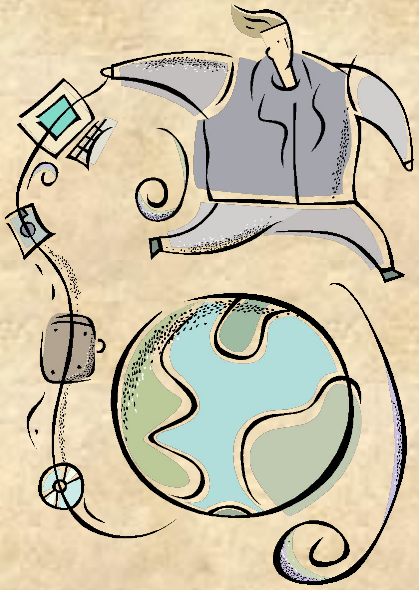
- A major OS will evolve over time for a number of reasons:

Hardware upgrades

New types of hardware

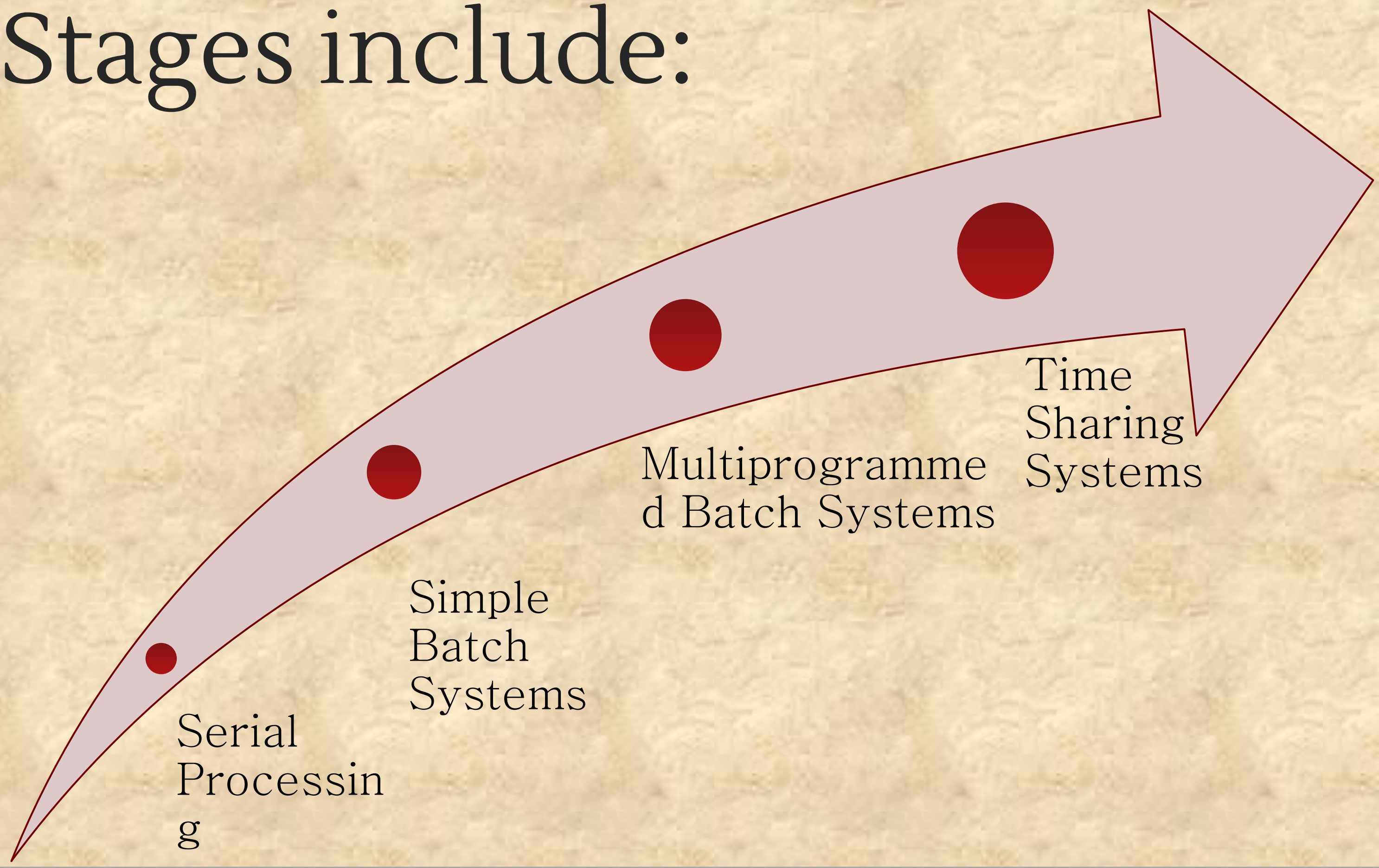
New services

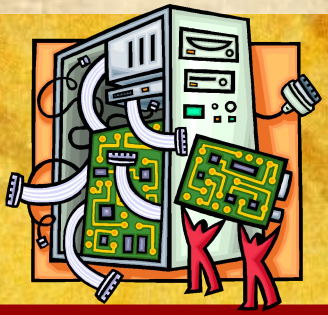
Fixes



Evolution of Operating Systems

- Stages include:





Serial Processing

Earliest Computers:

- No operating system
  - programmers interacted directly with the computer hardware
- Computers ran from a console with display lights, toggle switches, some form of input device, and a printer
- Users had access to the computer in “series”

Problems:

- Scheduling:
  - most installations used a hardcopy sign-up sheet to reserve computer time
    - time allocations could run short or long, resulting in wasted computer time
- Setup time
  - a considerable amount of time was spent just on setting up the program to

Simple Batch Systems

- Early computers were very expensive
  - important to maximize processor utilization
- Monitor (primitive operating system)
  - user no longer has direct access to processor
  - job is submitted to computer operator who batches them together and places them on an input device

Monitor Point of View

- Monitor controls the sequence of events
- *Resident Monitor* is software always in memory
- Monitor reads in job and gives control
- Job returns control to monitor

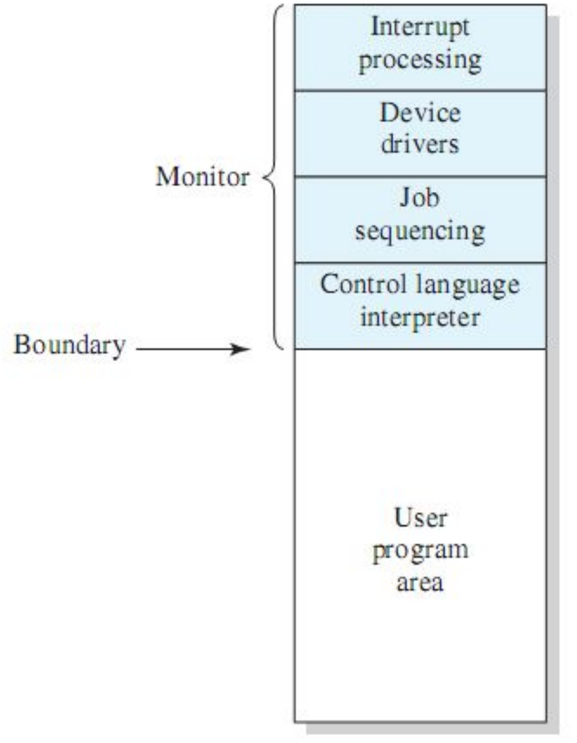


Figure 2.3 Memory Layout for a Resident Monitor

Processor Point of View

- Processor executes instruction from the memory containing the monitor
- Executes the instructions in the user program until it encounters an ending or error condition
- "*control is passed to a job*" means processor is fetching and executing instructions in a user program
- "*control is returned to the monitor*" means that the processor is fetching and executing instructions from the monitor program

Job Control Language (JCL)

Special type of programming language used to provide instructions to the monitor

what compiler to use

what data to use

Desirable Hardware Features

Memory protection for monitor

- while the user program is executing, it must not alter the memory area containing the monitor

Timer

- prevents a job from monopolizing the system

Privileged instructions

- can only be executed by the monitor

Interrupts

- gives OS more flexibility in controlling user programs

Modes of Operation

User Mode

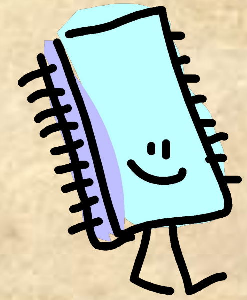
- user program executes in user mode
- certain areas of memory are protected from user access
- certain instructions may not be executed

Kernel Mode

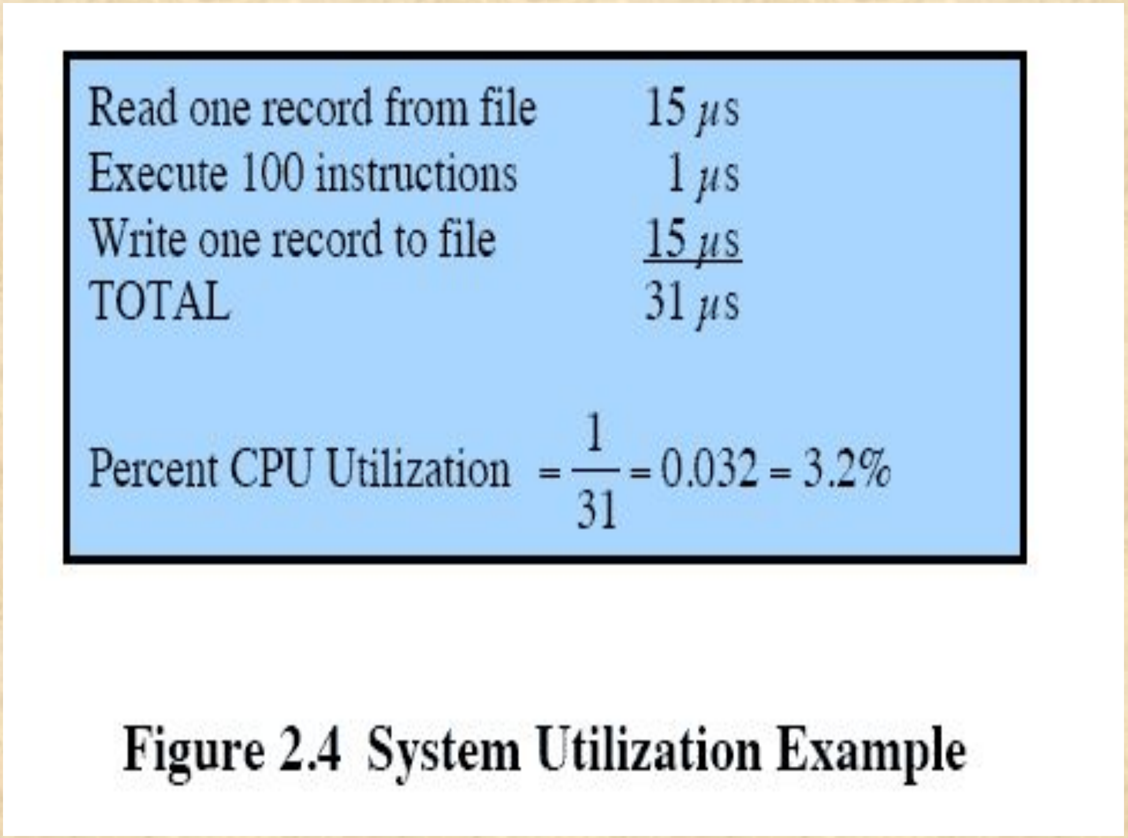
- monitor executes in kernel mode
- privileged instructions may be executed
- protected areas of memory may be accessed

Simple Batch System Overhead

- Processor time alternates between execution of user programs and execution of the monitor
- Sacrifices:
  - some main memory is now given over to the monitor
  - some processor time is consumed by the monitor
- Despite overhead, the simple batch system improves utilization of the computer.



Multiprogrammed Batch Systems



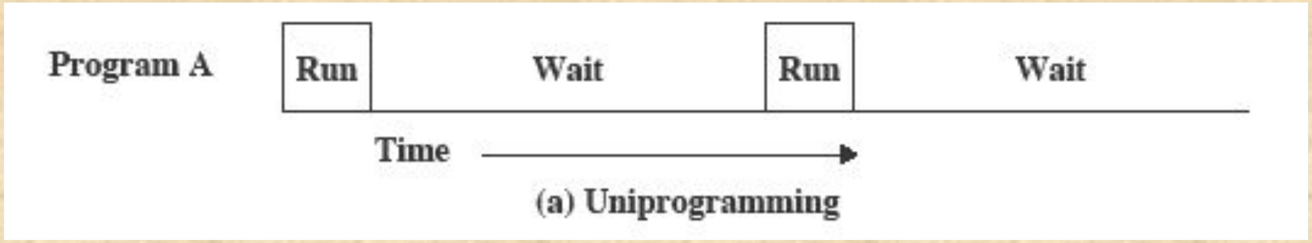
Read one record from file	15 µs
Execute 100 instructions	1 µs
Write one record to file	15 µs
TOTAL	31 µs

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

Figure 2.4 System Utilization Example

- Processor is often idle
  - even with automatic job sequencing
  - I/O devices are slow compared to processor

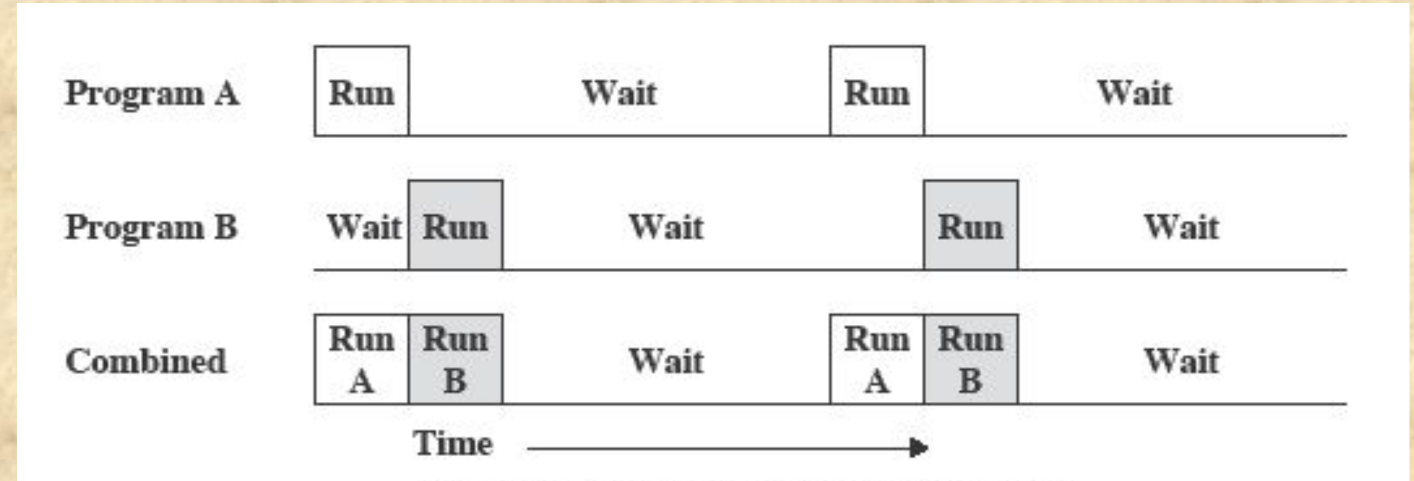
Uniprogramming



(a) Uniprogramming

- The processor spends a certain amount of time executing, until it reaches an I/O instruction; it must then wait until that I/O instruction concludes before proceeding

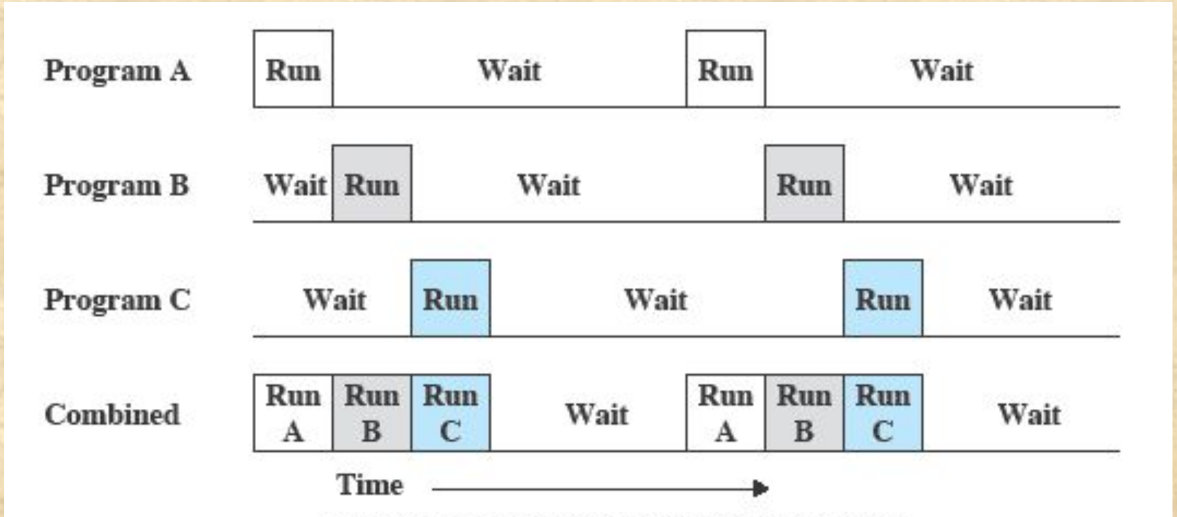
Multiprogramming



(b) Multiprogramming with two programs

- What if there's enough memory to hold the OS (resident monitor) and *two* user programs.
- When one job needs to wait for I/O, the processor can switch to the other job, which may not be waiting.

Multiprogramming



(c) Multiprogramming with three programs

- Multiprogramming
  - also known as multitasking
  - memory is expanded to hold three, four, or more programs and switch among all of them

Multiprogramming Example

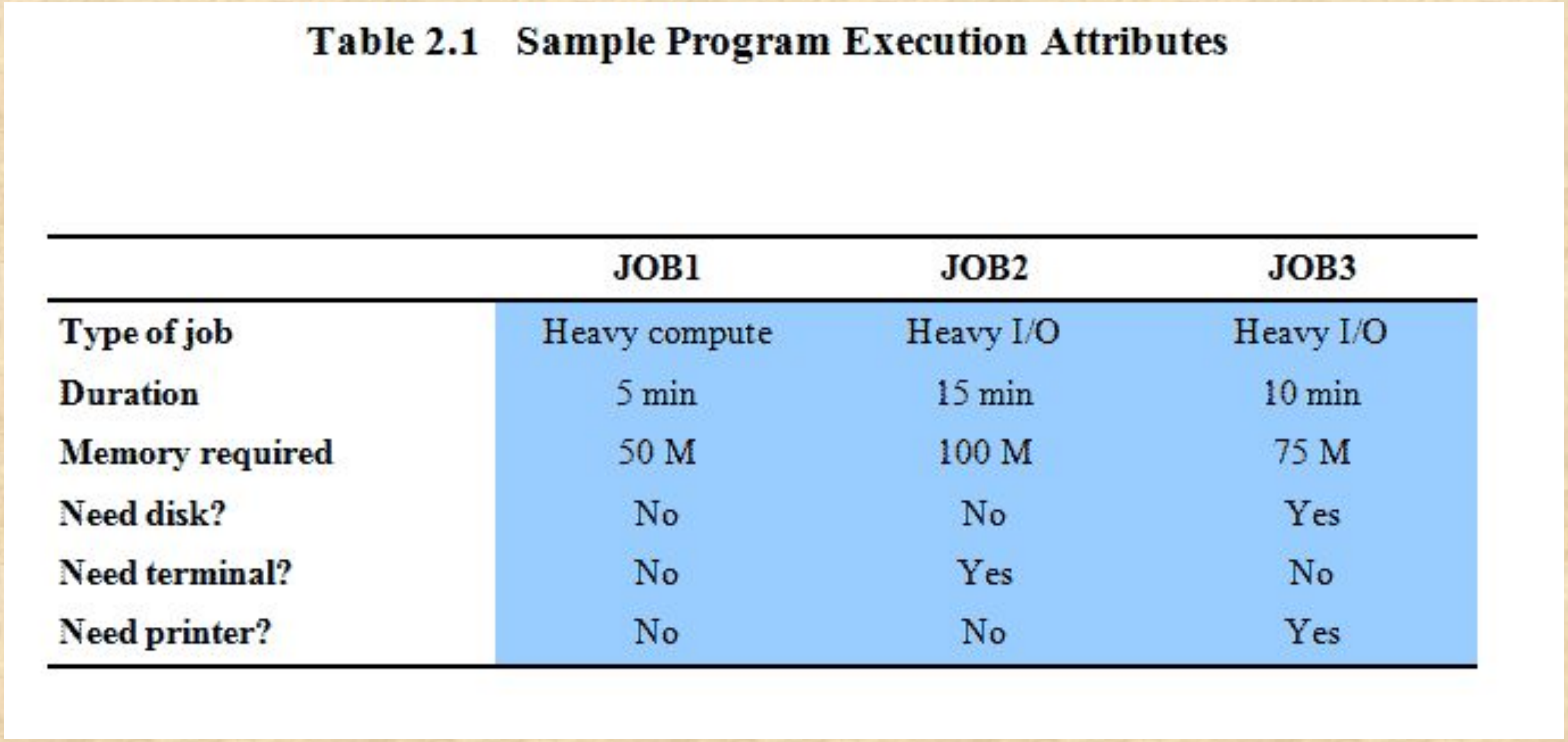
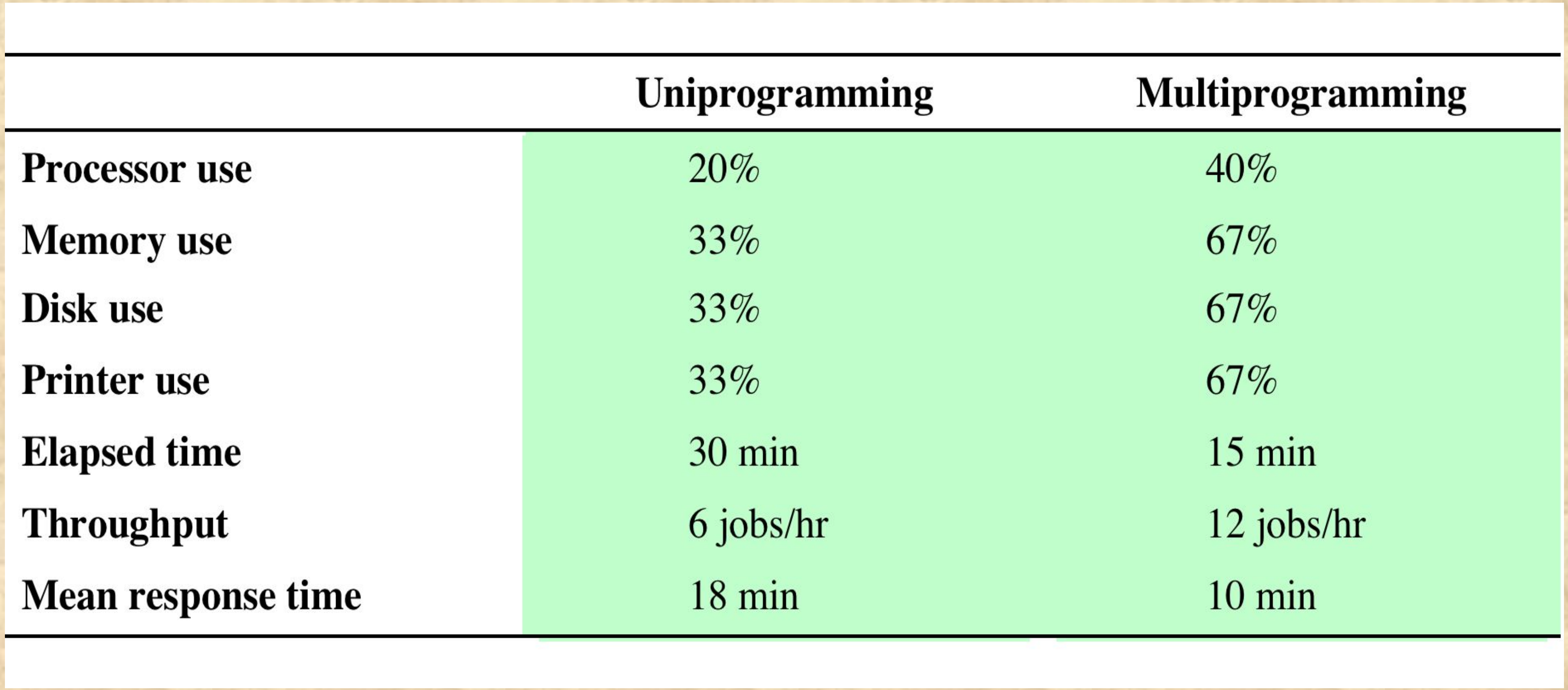


Table 2.1 Sample Program Execution Attributes

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

Effects on Resource Utilization



	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Table 2.2 Effects of Multiprogramming on Resource Utilization

Utilization Histograms

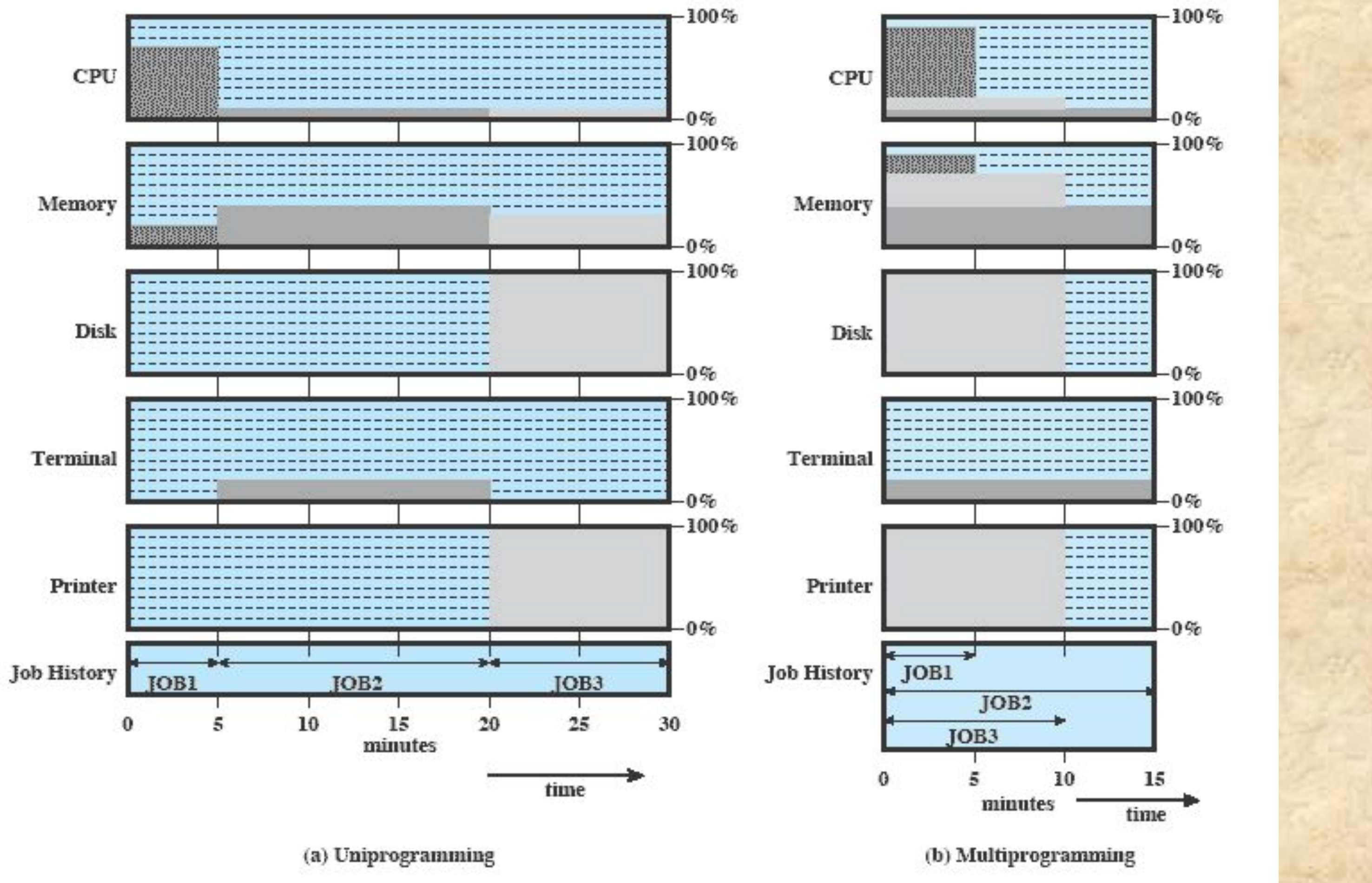
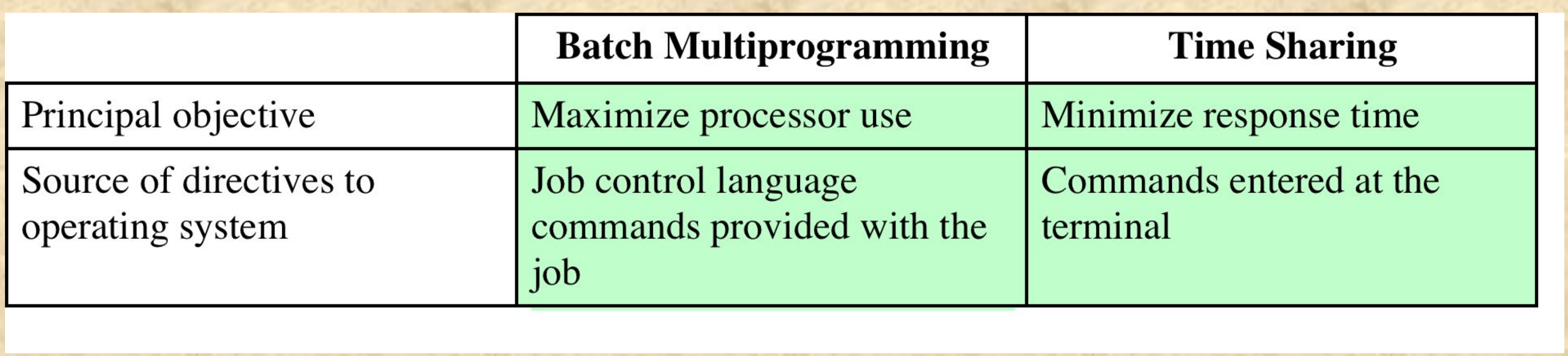


Figure 2.6 Utilization Histograms

Time-Sharing Systems

- Can be used to handle multiple *interactive* jobs
- Processor time is shared among multiple users
- Origin: multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation

Batch Multiprogramming vs. Time Sharing



	Batch Multiprogramming	**Time Sharing**
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

Table 2.3 Batch Multiprogramming versus Time Sharing

Compatible Time-Sharing Systems

CTSS

- One of the first time-sharing operating systems
- Developed at MIT by a group known as Project MAC for IBM 709/7094
- Ran on a computer with 32,000 *36*-bit words of main memory, with the resident monitor consuming 5000 of that
- To simplify both the monitor and memory management a program was always loaded to start at the location of the 5000th word

Time Slicing

- System clock generates interrupts at a rate of approximately one every 0.2 seconds
- At each interrupt OS regained control and could assign processor to another user
- At regular time intervals the current user would be preempted and another user loaded in
- Old user programs and data were written out to disk
- Old user program code and data were restored in main memory when that program was next

CTSS Operation

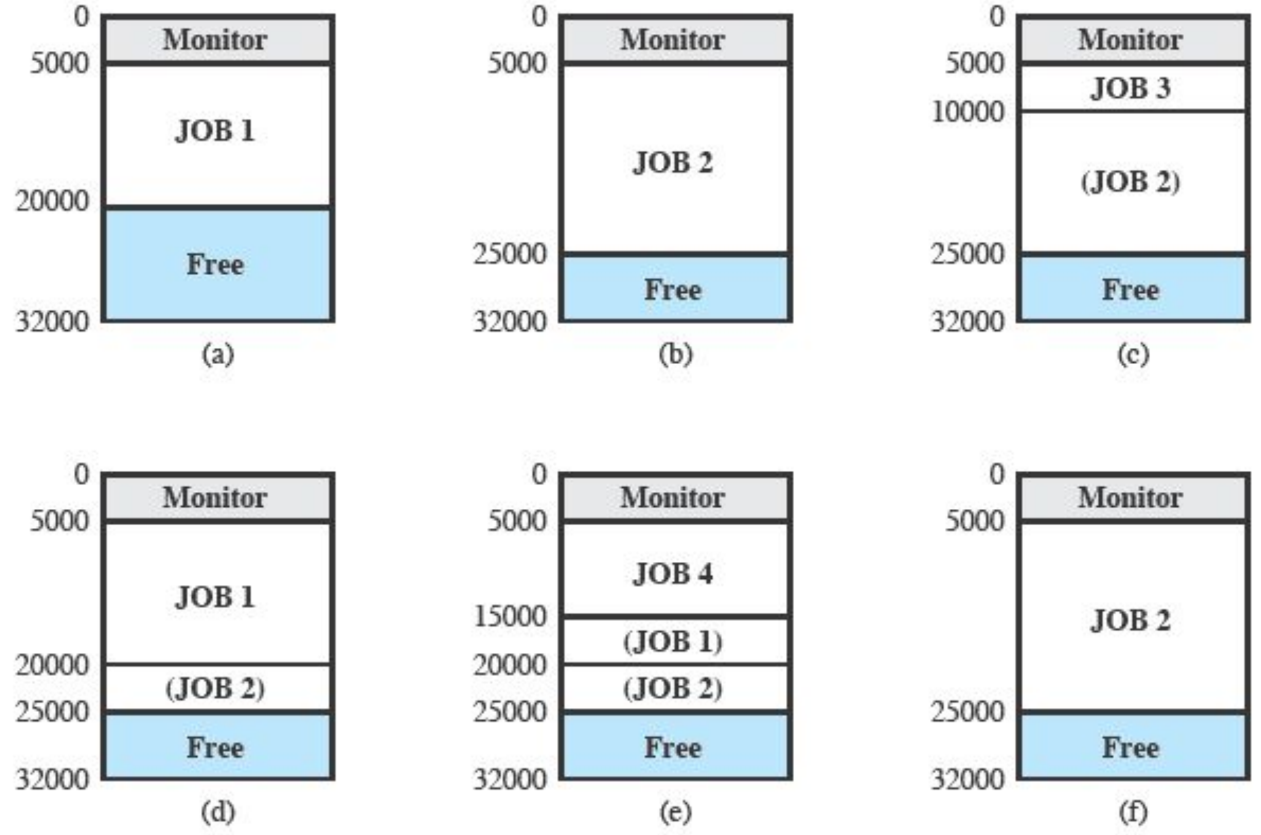
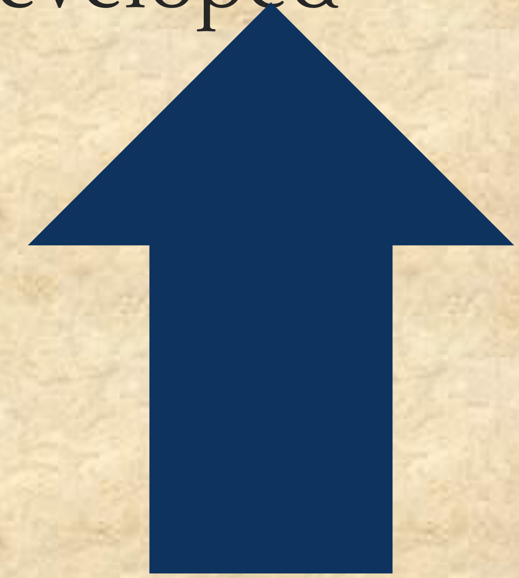


Figure 2.7 CTSS Operation

Major Advances

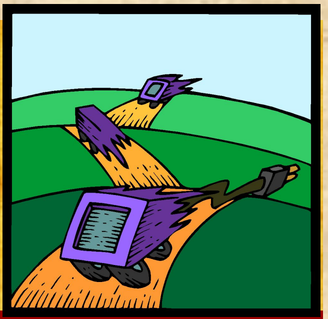
- Operating Systems are among the most complex pieces of software ever developed



Major advances in development include:

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

Process



- Fundamental to the structure of operating systems

A *process* can be defined as:

a program in execution

an instance of a running program

the entity that can be assigned to, and executed on, a processor

a unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

Development of the Process

- Three major lines of computer system development created problems in timing and synchronization that contributed to the development:

multiprogramming batch operation

- processor is switched among the various programs residing in main memory

time sharing

- be responsive to the individual user but be able to support many users simultaneously

real-time transaction systems

- a number of users are entering queries or updates against a database

Causes of Errors

- Improper synchronization
  - a program must wait until the data are available in a buffer
  - improper design of the signaling mechanism can result in loss or duplication
- Failed mutual exclusion
  - more than one user or program attempts to make use of a shared resource at the same time
  - only one routine at at time allowed to perform an update against a given file

- Nondeterminate program operation
  - program execution is interleaved by the processor when memory is shared
  - the order in which programs are scheduled may affect their outcome
- Deadlocks
  - it is possible for two or more programs to be hung up waiting for each other
  - may depend on the chance timing of resource allocation and release

Components of a Process

- A process contains three components:
  - an executable program
  - the associated data needed by the program (variables, work space, buffers, etc.)
  - the execution contex (or "process state") the program

- The execution context is essential:
  - it is the internal data by which the OS is able to supervise and control the process
  - includes the contents of the various process registers
  - includes information such as the priority of the process and whether the process is waiting for the completion of a particular I/O event

Process Management

- The entire state of the process at any instant is contained in its context
- New features can be designed and incorporated into the OS by expanding the context to include any new information needed to support the feature

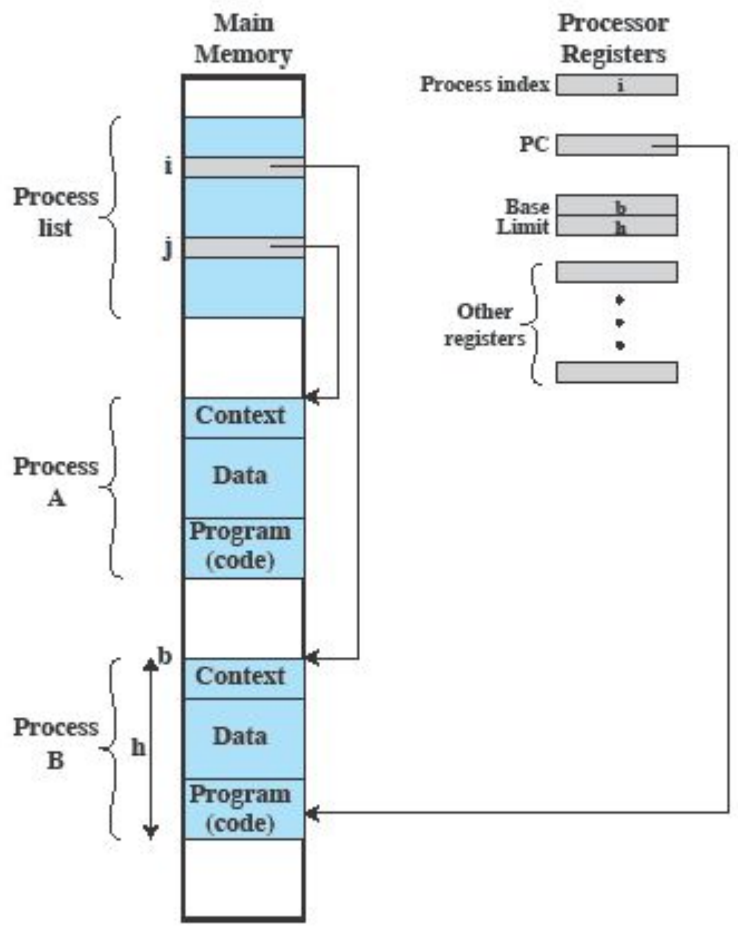


Figure 2.8 Typical Process Implementation

Memory Management

- The OS has five principal storage management responsibilities:

process isolation	automatic allocation and management	support of modular programmin g	protection and access control	long-term storage

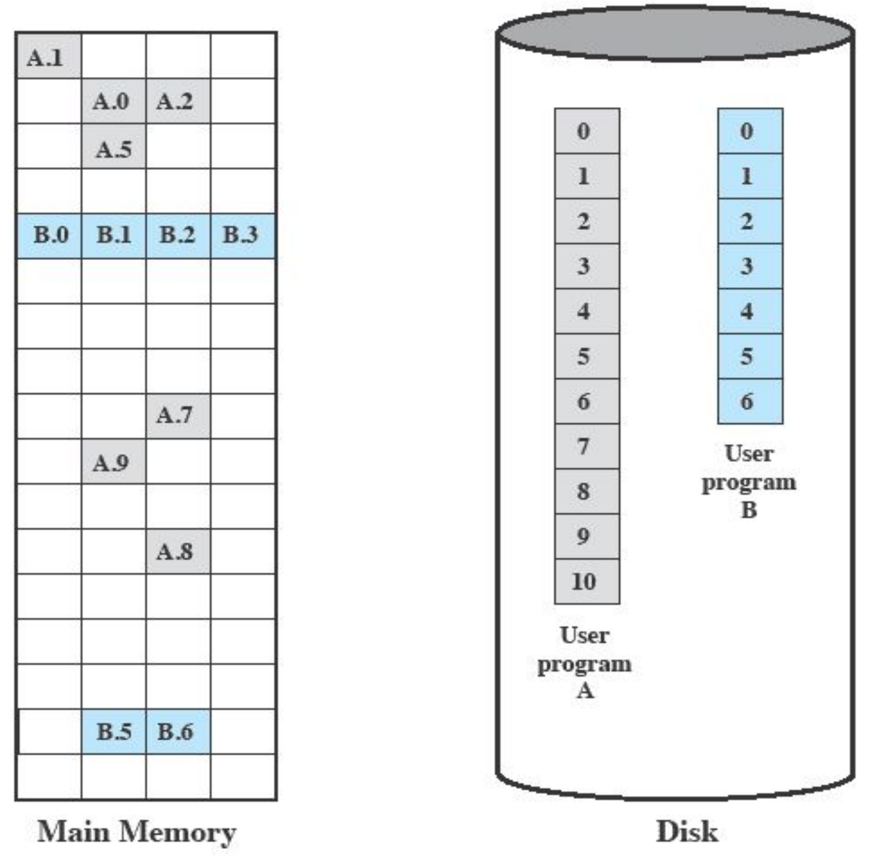
Virtual Memory

- A facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available
- Conceived to meet the requirement of having multiple user jobs reside in main memory concurrently

Paging

- Allows processes to be comprised of a number of fixed-size blocks, called pages
- Program references a word by means of a virtual address
  - consists of a page number and an offset within the page
  - each page may be located anywhere in main memory
- Provides for a dynamic mapping between the virtual address used in the program and a real (or physical) address in main memory

Virtual Memory



Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Figure 2.9 Virtual Memory Concepts

Virtual Memory Addressing

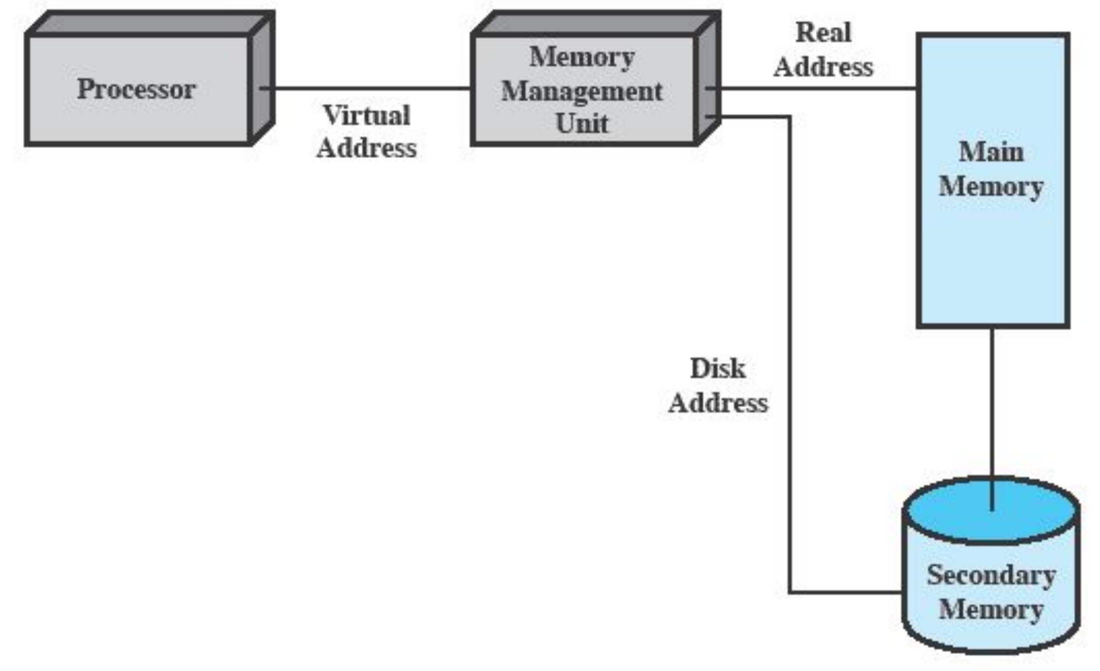
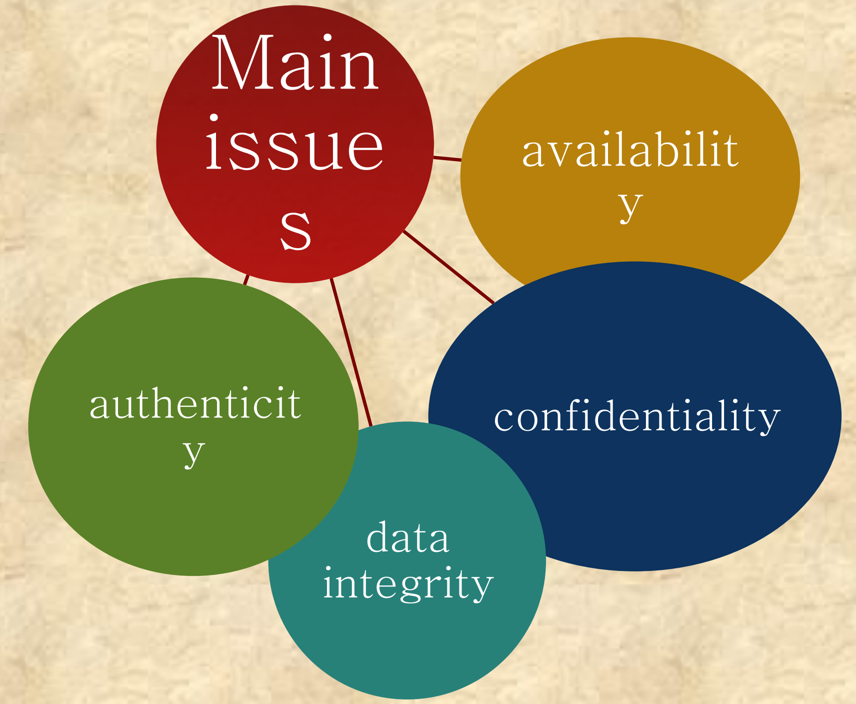


Figure 2.10 Virtual Memory Addressing

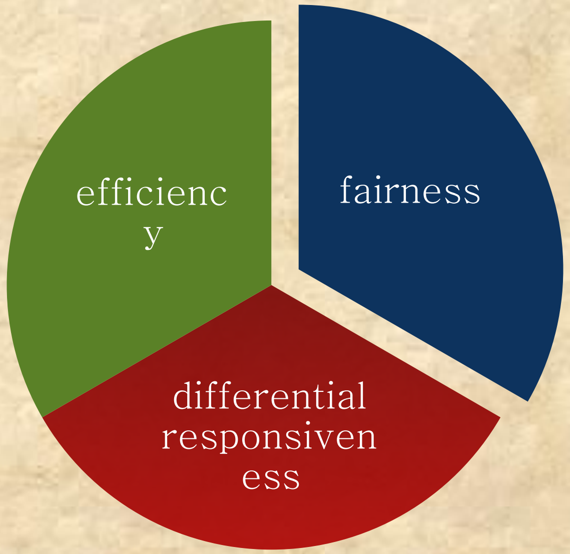
Information Protection and Security

- The nature of the threat that concerns an organization will vary greatly depending on the circumstances
- The problem involves controlling access to computer systems and the information stored in them



Scheduling and Resource Management

- Key responsibility of an OS is managing resources
- Resource allocation policies must consider:



Key Elements of an Operating System

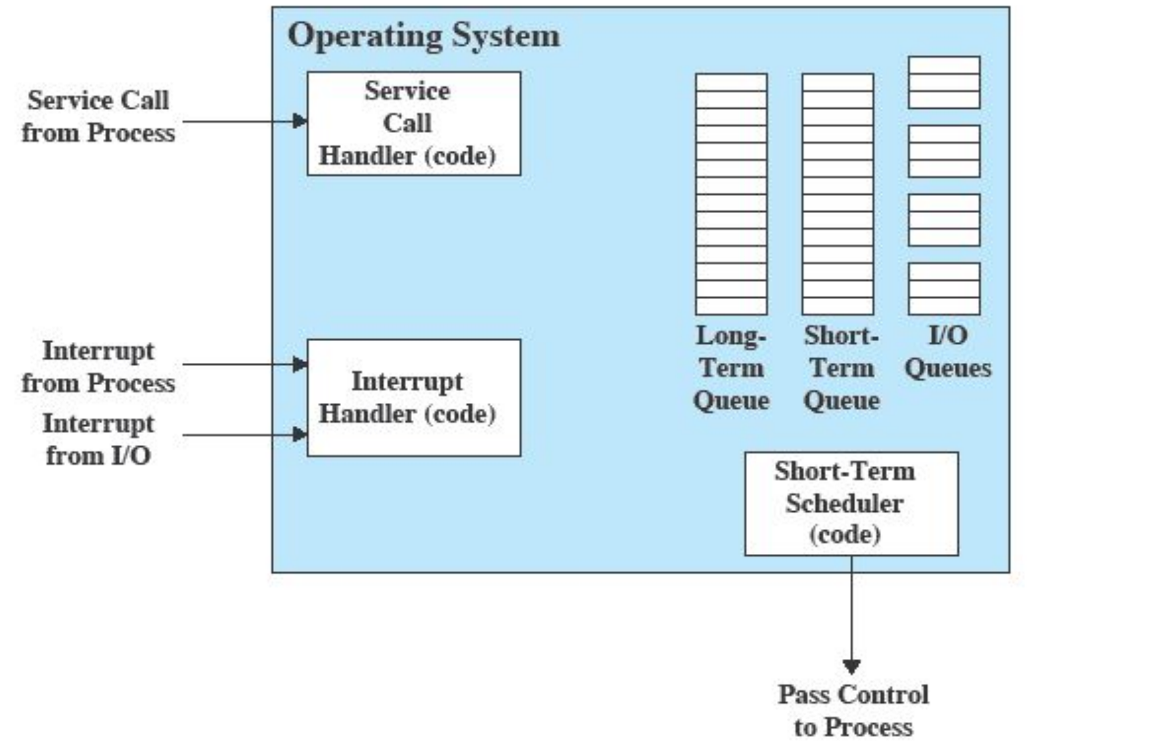


Figure 2.11 Key Elements of an Operating System for Multiprogramming