Chapter 2 Operating System Overview

Operating Systems: Internals and Design Principles

Eighth Edition, William Stallings

Operating System

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

Main Objectives of an OS
Convenience
Efficiency
Ability to be updated

System Structure

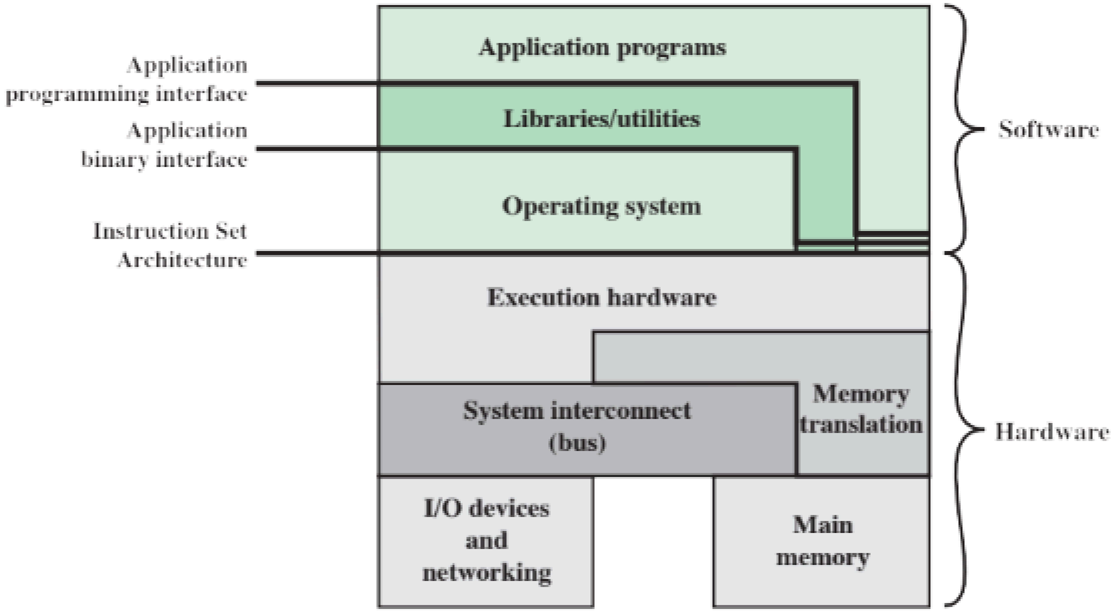


Figure 2.1 Computer Hardware and Software Structure

Operating System Services

- Program development
- Program execution
- Access to I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

Key Interfaces

- Instruction Set Architecure (ISA)
- Application Binary Interface (ABI)
- Application Programming Interface (API)

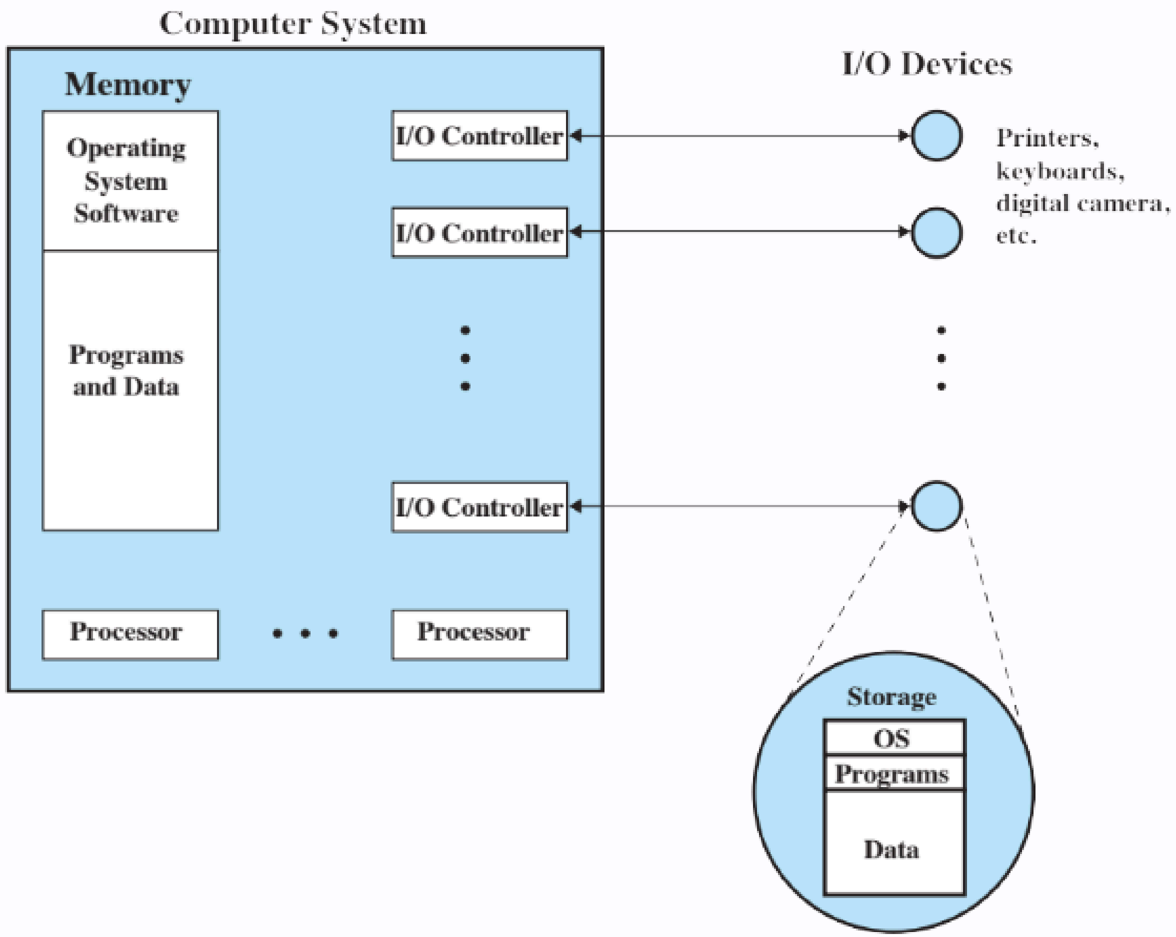
The Role of an OS

- A computer is a set of resources for the movement, storage, and processing of data
- The OS is responsible for managing these resources

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 depend on the processor to allow it to regain control

OS as a Resource Manager



Operating System Updates

- An OS will need to be updated over time for a number of reasons:

Hardware upgrades

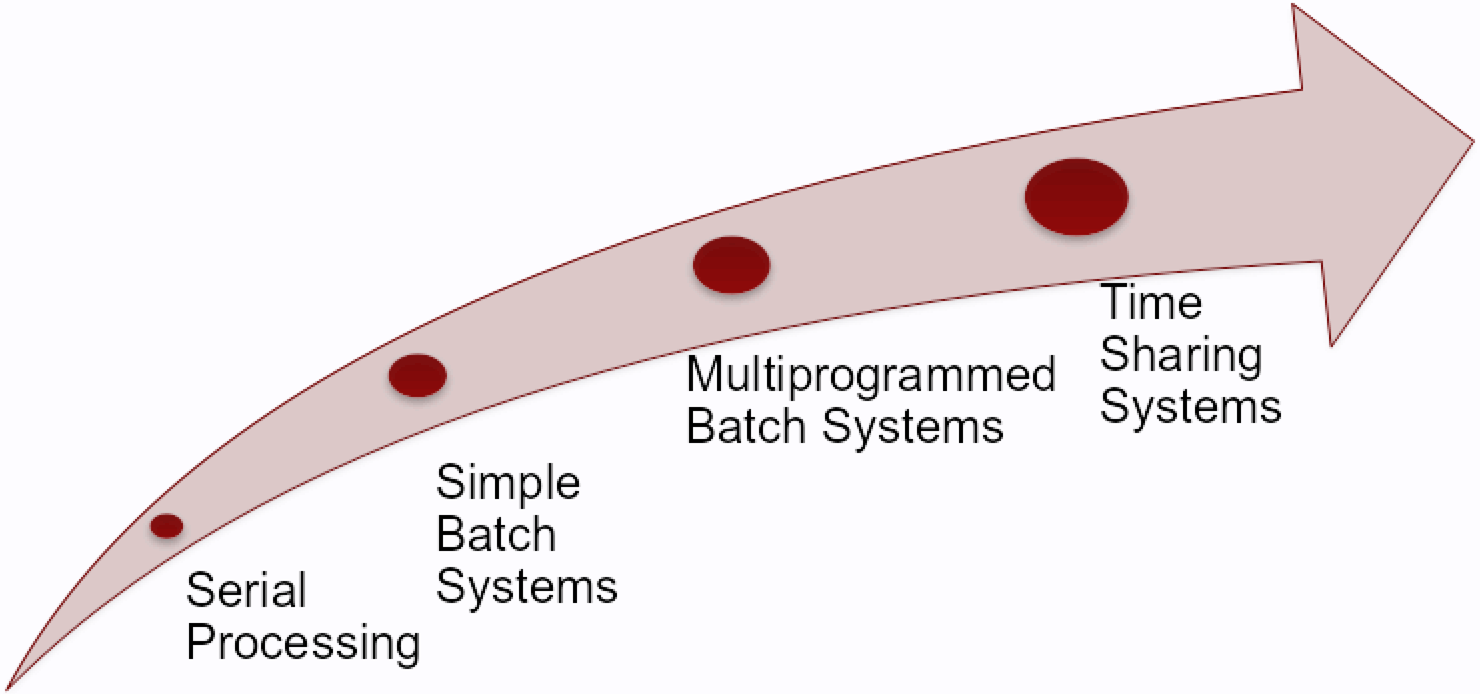
new types of hardware

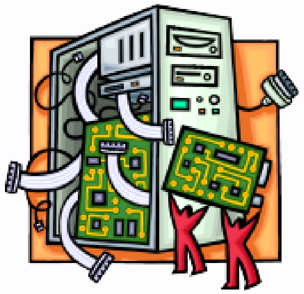
new services

Fixes

How Operating Systems have Changed Over Time

- 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

Problems:

- Scheduling:
  - most installations used a hardcopy sign-up sheet to reserve computer time
  - time allocations could run short or long,

Simple Batch Systems

- Early computers were very expensive
  - important to maximize processor utilization
- Monitor
  - 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
  - program branches back to the monitor when finished

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

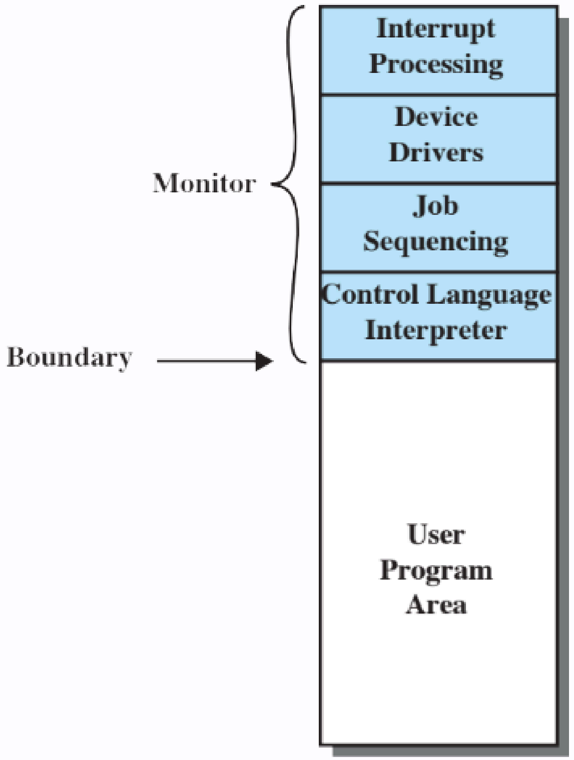


Figure 2.3 Memory Layout for a Resident Monitor

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

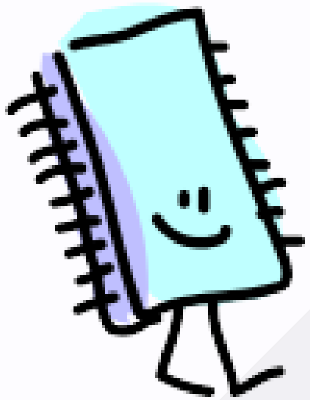
- user program executes in user mode
- certain areas of memory are protected from user access
- certain instructions may not be executed

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

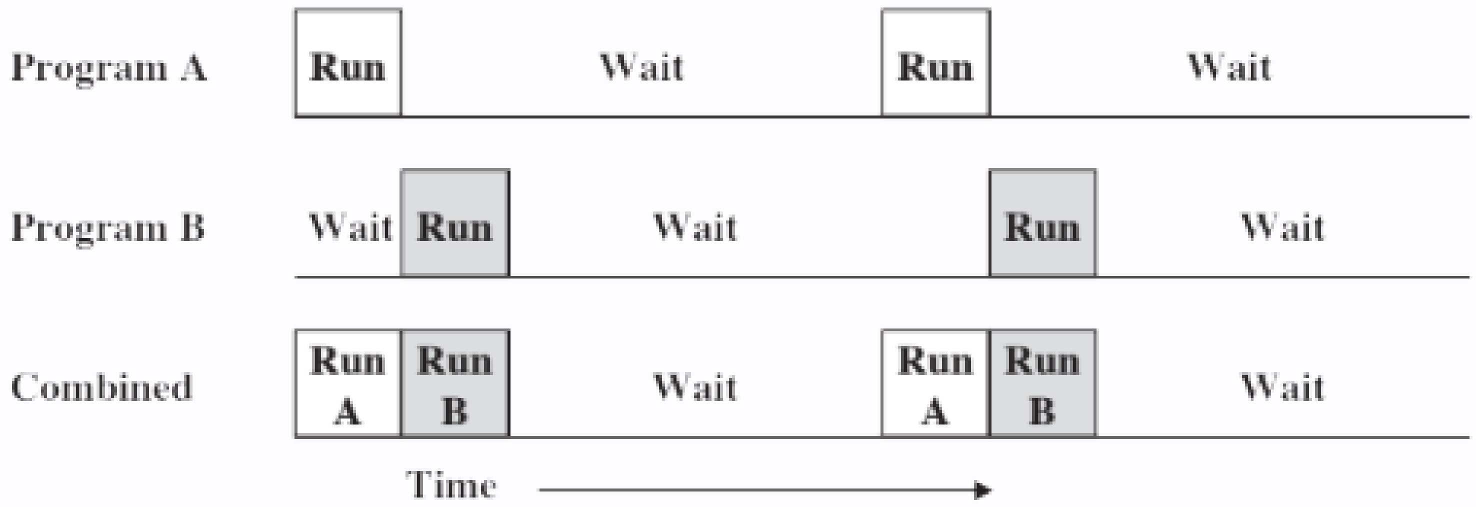
- Processor is often idle
  - even with automatic job sequencing
  - I/O devices are slow compared to processor

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

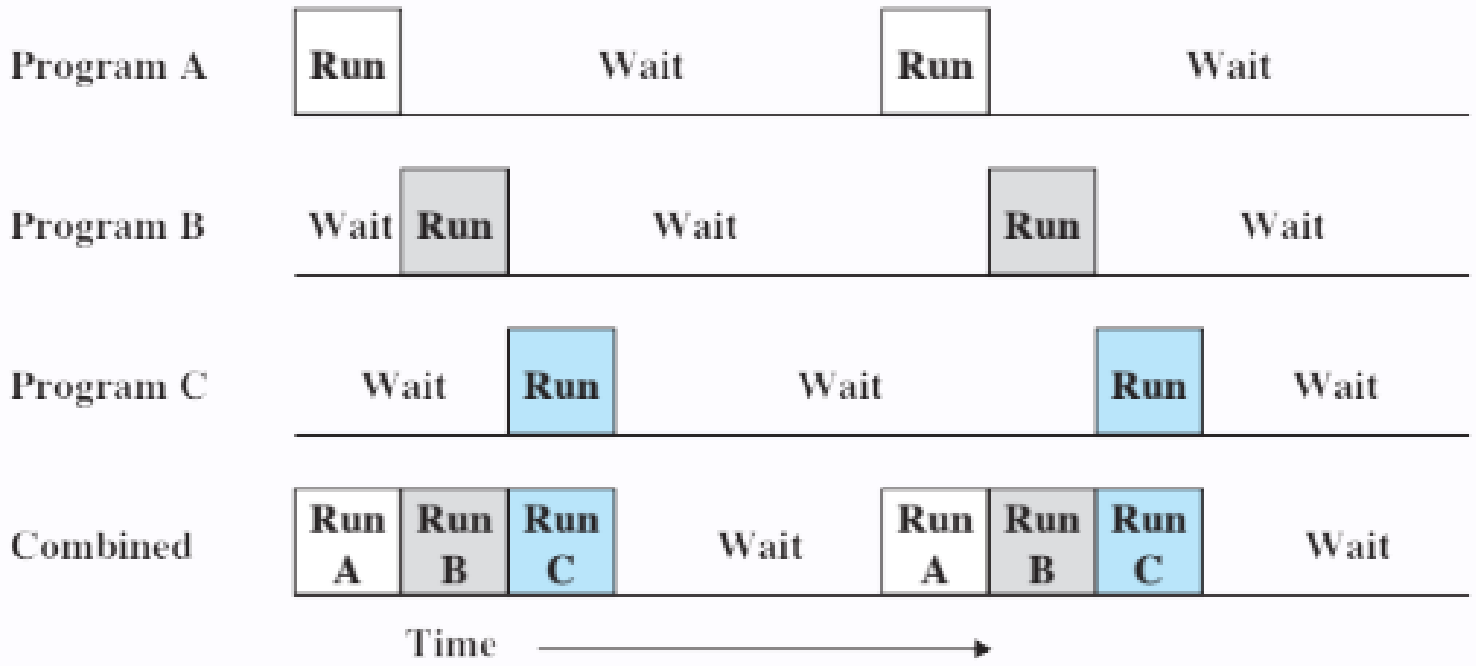
Multiprogramming



(b) Multiprogramming with two programs

- There must be enough memory to hold the OS (resident monitor) and one user program
- When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O

Multiprogramming (cont)



(c) Multiprogramming with three programs

- Multiprogramming
  - also known as *multitasking*

Multiprogramming Example

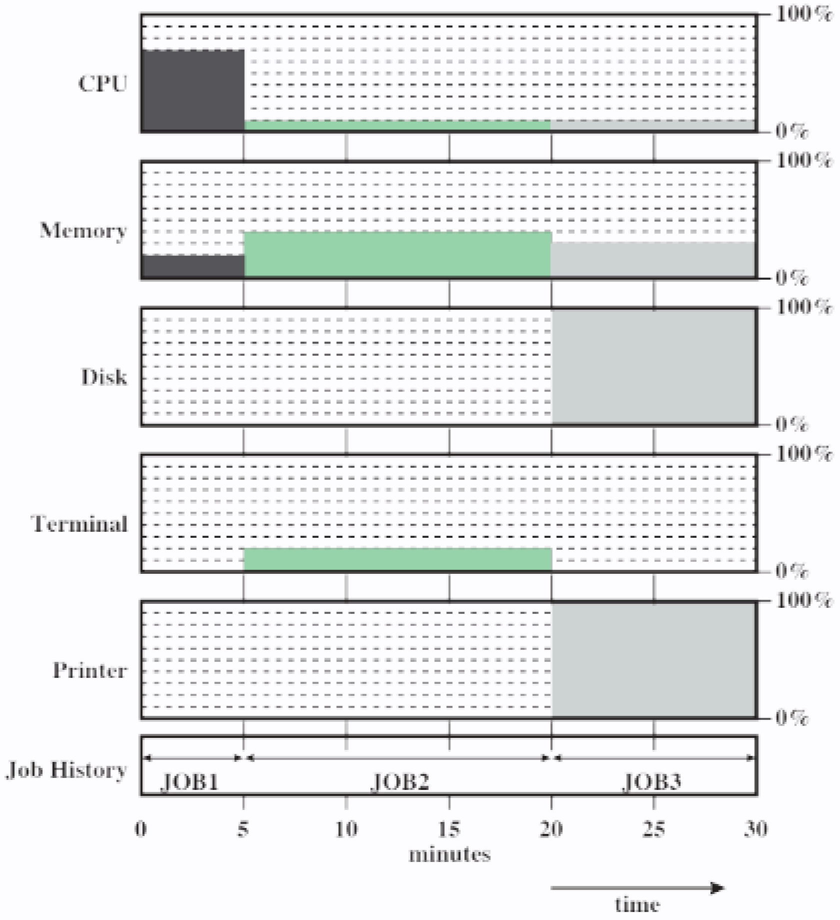
	Job 1	Job 2	Job 3
Type of Job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 minutes	15 minutes	10 Minutes
Memory Required	5 M	100 M	75 M
Need Disk?	No	No	Yes
Need Terminal?	No	Yes	No
Need Printer?	No	No	Yes

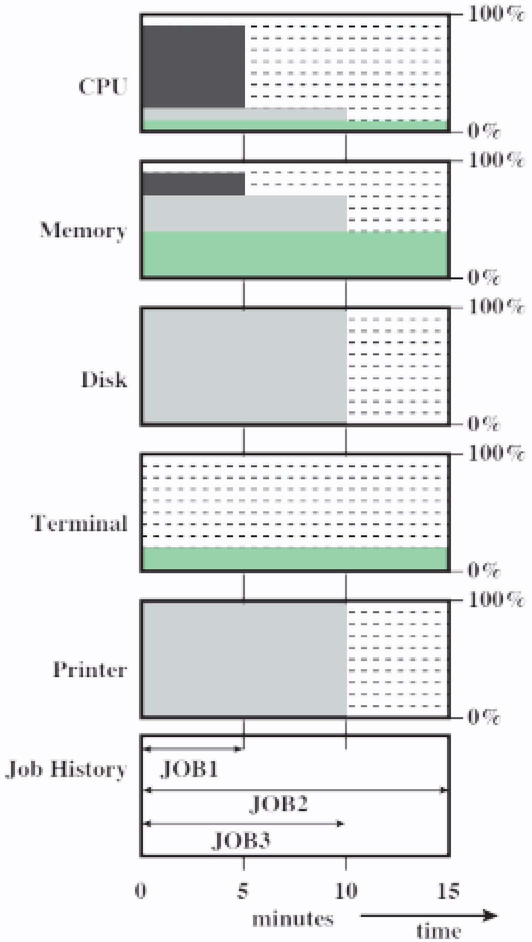
Effects on Resource Utilization

	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





Time-Sharing Systems

- Can be used to handle multiple interactive jobs
- Processor time is shared among multiple users
- 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

- operating systems
    - Developed at MIT by a group known as Project MAC
    - 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

CTSS Operation

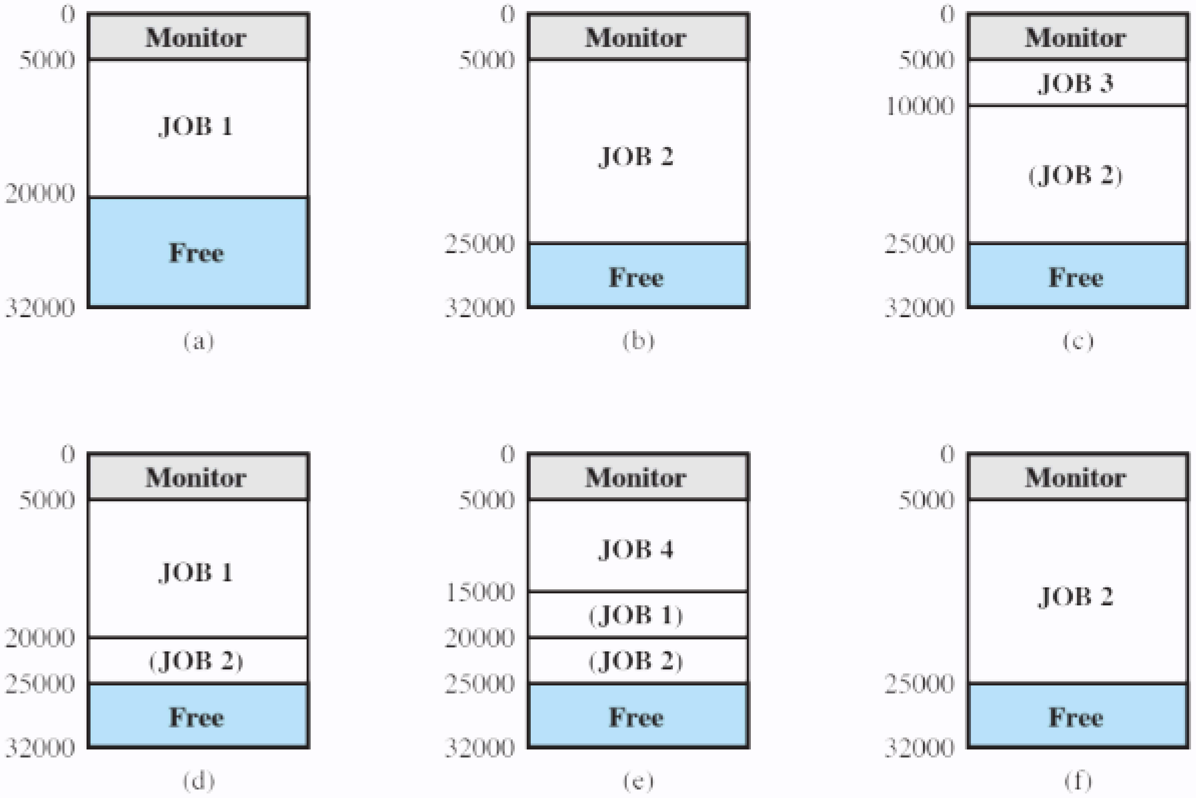


Figure 2.7 CTSS Operation

Major Achievements

- 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

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

multiprogramming batch operation

- 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

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 a time allowed to perform an update against the file
- 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
- 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

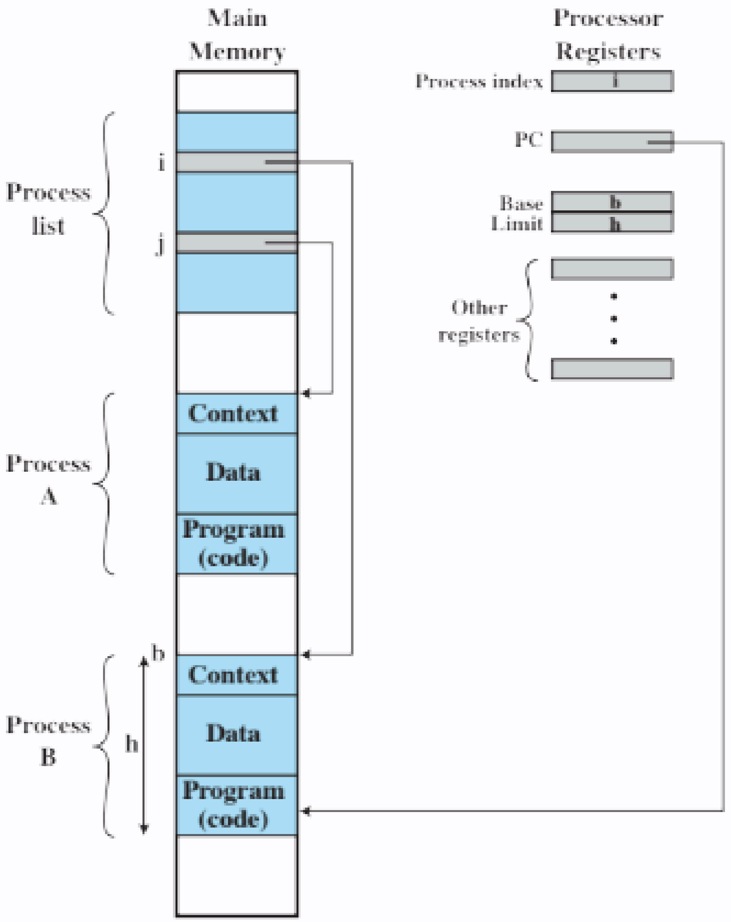
Components of a Process

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



Memory Management

- The OS has five principal storage management responsibilities:

process isolation	automatic allocation and management	support of modular programming	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 Concepts

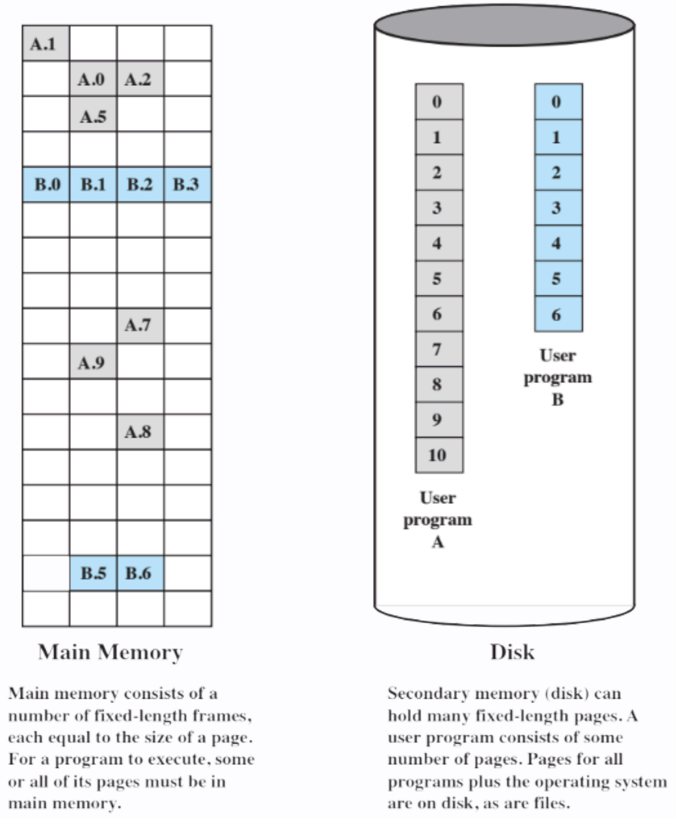


Figure 2.9 Virtual Memory Concepts

Virtual Memory Addressing

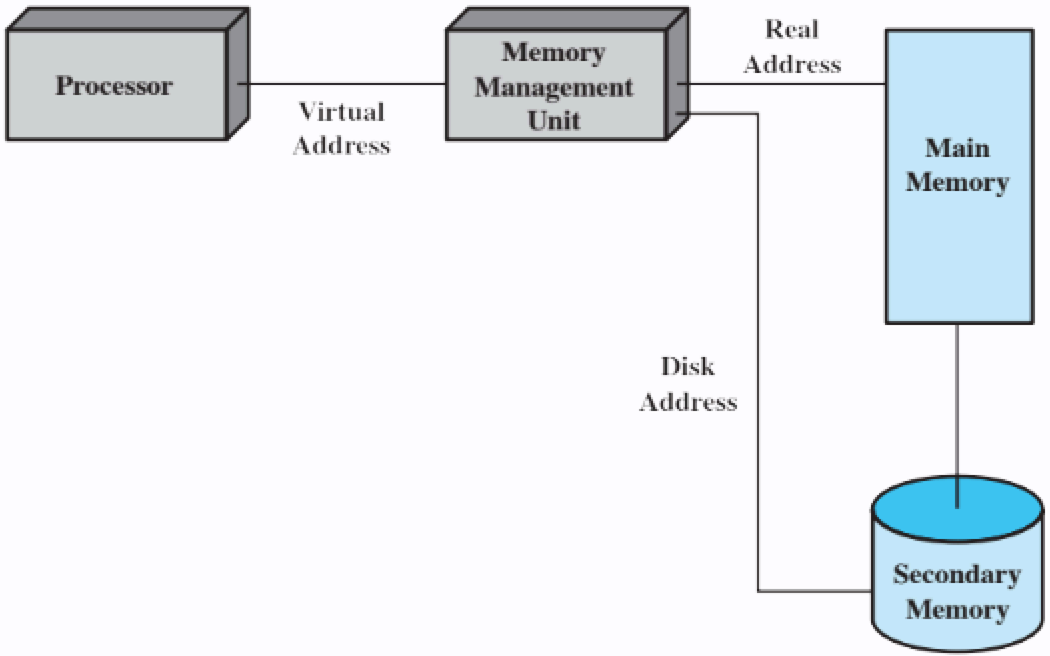
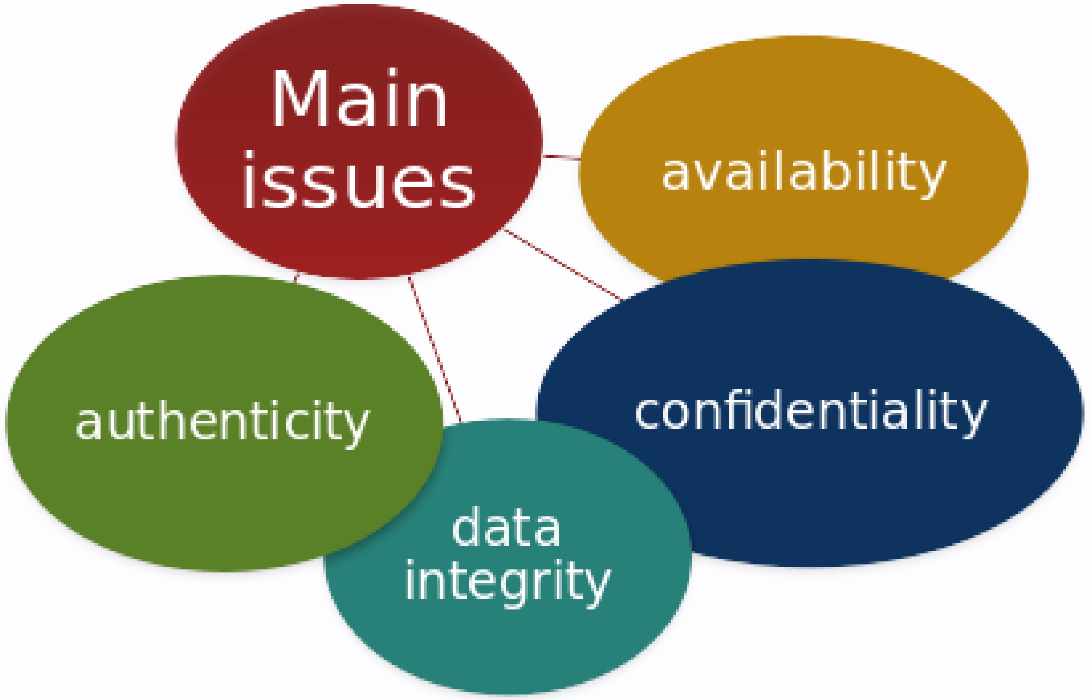


Figure 2.10 Virtual Memory Addressing

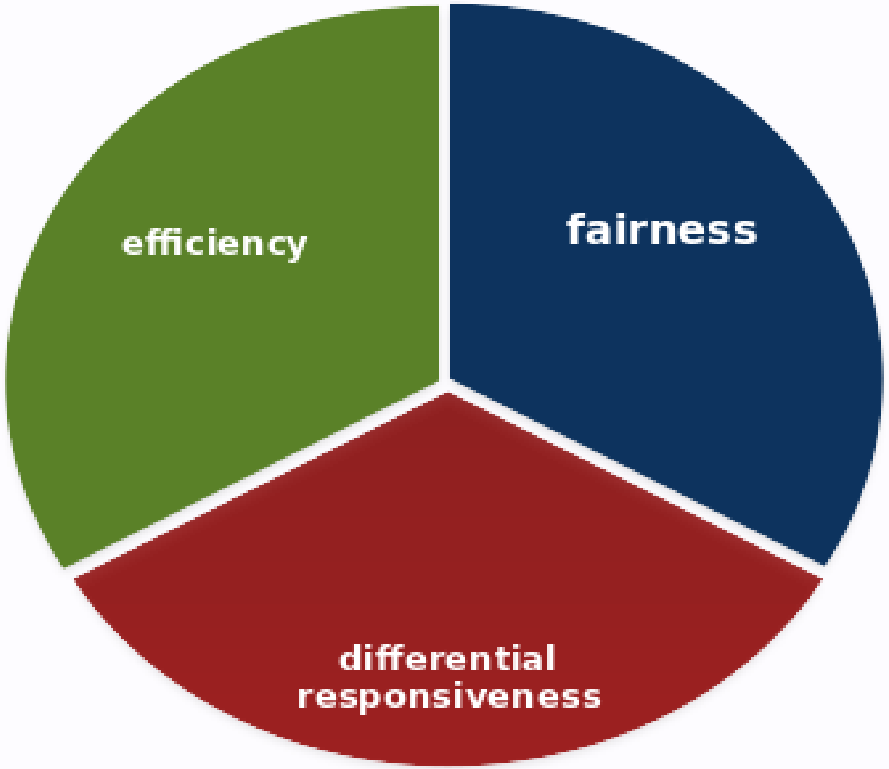
Information Protection and Security

- 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:



Elements of a Multiprogramming OS

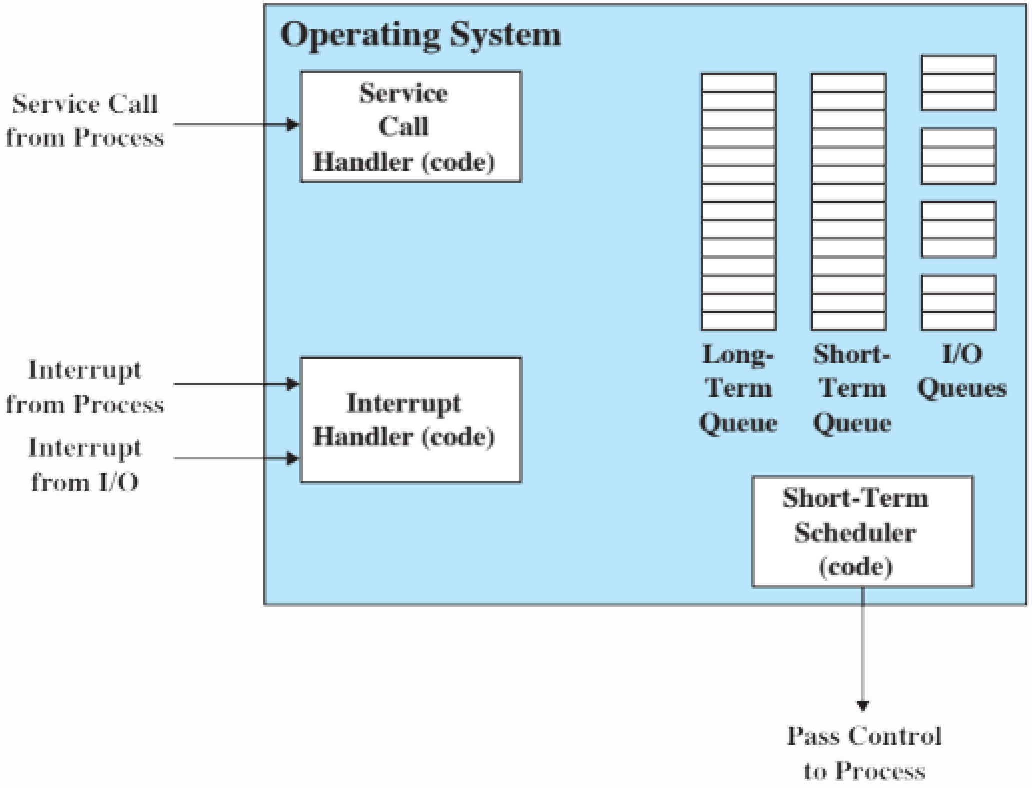


Figure 2.11 Key Elements of an Operating System for Multiprogramming

Different Architectural Approaches

- Demands on operating systems require new ways of organizing the OS

Different approaches and design elements have been tried:

- multithreading
- symmetric multiprocessing
- distributed operating systems
- object-oriented design

Microkernel Architecture

- Assigns only a few essential functions to the kernel:

address spaces	interprocess communication (IPC)	basic scheduling

- The approach:

address spaces	interprocess communication (IPC)	basic scheduling

Multithreading

Thread

- dispatchable unit of work
- includes a processor context and its own data area to enable subroutine branching
- executes sequentially and is interruptible

Process

- a collection of one or more threads and associated system resources
- programmer has greater control over the modularity of the application and the timing of application related events

Symmetric Multiprocessing (SMP)

- Term that refers to a computer hardware architecture and also to the OS behavior that exploits that architecture
- Several processes can run in parallel
- Multiple processors are transparent to the user
  - these processors share same main memory and I/O facilities
  - all processors can perform the same functions
- The OS takes care of scheduling of threads or processes on individual processors and of synchronization among processors

SMP Advantages

Performance

more than one process can be running simultaneously, each on a different processor

Availability

failure of a single process does not halt the system

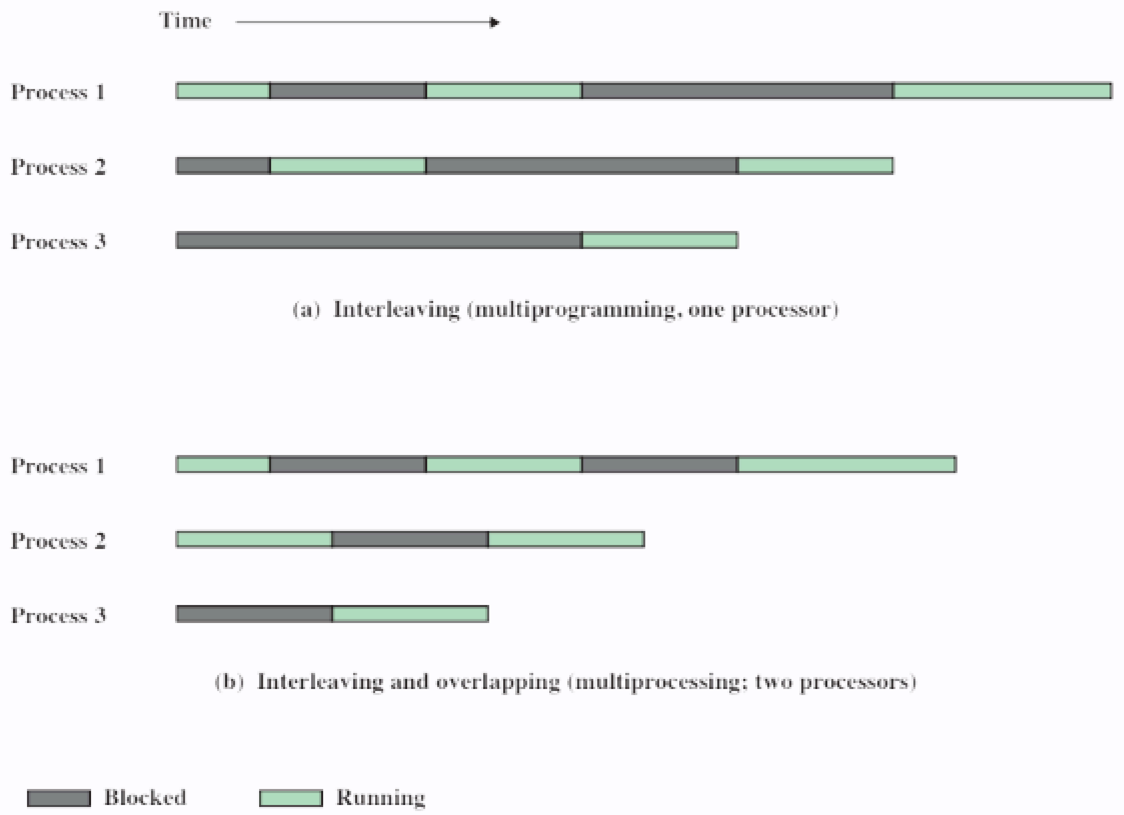
Incremental Growth

performance of a system can be enhanced by adding an additional processor

Scaling

vendors can offer a range of products based on the number

Multiprogramming versus Multiprocessing



OS Design

Distributed Operating System

- Provides the illusion of
  - a single main memory space
  - single secondary memory space
  - unified access facilities
- State of the art for distributed operating systems lags that of uniprocessor and SMP 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
- Eases the development of distributed tools and full-blown distributed operating systems

Summary

- Operating system objectives and functions
  - User/computer interface
  - Resource manager
- Evolution of operating systems
  - Serial processing
  - Simple/multiprogrammed/time-sharing batch systems
    - Major achievements
    - Developments leading to modern operating systems