Operating System Overview

Chapter 2
Operating System

• A program that controls the execution of application programs

• An interface between applications and hardware
Operating System Objectives

• Convenience
  – Makes the computer more convenient to use

• Efficiency
  – Allows computer system resources to be used in an efficient manner

• Ability to evolve
  – Permit effective development, testing, and introduction of new system functions without interfering with service
Layers of Computer System

![Diagram showing the layers of a computer system]

**Figure 2.1** Layers and Views of a Computer System
Services Provided by the Operating System

- Program development
  - Editors and debuggers
- Program execution
- Access to I/O devices
- Controlled access to files
- System access
Services Provided by the Operating System

• Error detection and response
  – Internal and external hardware errors
    • Memory error
    • Device failure
  – Software errors
    • Arithmetic overflow
    • Access forbidden memory locations
  – Operating system cannot grant request of application
Services Provided by the Operating System

• 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 program that is executed
• Operating system relinquishes control of the processor
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 an Operating System

- Hardware upgrades plus new types of hardware
- New services
- Fixes
Evolution of Operating Systems

• Serial Processing
  – No operating system

• Simple Batch Systems
  – Monitor

• Multiprogrammed Batch Systems
  – Multi-programming

• Time Sharing Systems
  – Multi-User
Serial Processing Systems

- No operating system
- Machines run from a console with display lights, toggle switches, input device, and printer
- Schedule time
- Setup included loading the compiler, source program, saving compiled program, and loading and linking
Simple Batch Systems

• Monitors
  – Software that controls the sequence of events
  – Batch jobs together
  – Program branches back to monitor when finished

• Job Control Language (JCL)
  – Special type of programming language
  – Provides instruction to the monitor
    • What compiler to use
    • What data to use
Hardware Features
(Batch Systems)

• Memory protection
  – Do not allow the memory area containing the monitor to be altered

• Timer
  – Prevents a job from monopolizing the system

• Interrupts
  – Early computer models did not have this capability
Hardware Features
(Batch Systems)

• Privileged instructions
  – Certain machine level instructions can only be executed by the monitor

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

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

- Processor must wait for I/O instruction to complete before preceding
I/O Devices Slow

<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read one record from file</td>
<td>15 µs</td>
</tr>
<tr>
<td>Execute 100 instructions</td>
<td>1 µs</td>
</tr>
<tr>
<td>Write one record to file</td>
<td>15 µs</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>31 µs</strong></td>
</tr>
</tbody>
</table>

Percent CPU Utilization = \(\frac{1}{31} = 0.032 = 3.2\%\)

Figure 2.4 System Utilization Example
Multiprogrammed Batch Systems

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

(c) Multiprogramming with three programs
Example

Table 2.1 Sample Program Execution Attributes

<table>
<thead>
<tr>
<th></th>
<th>JOB1</th>
<th>JOB2</th>
<th>JOB3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of job</td>
<td>Heavy compute</td>
<td>Heavy I/O</td>
<td>Heavy I/O</td>
</tr>
<tr>
<td>Duration</td>
<td>5 min</td>
<td>15 min</td>
<td>10 min</td>
</tr>
<tr>
<td>Memory required</td>
<td>50 M</td>
<td>100 M</td>
<td>75 M</td>
</tr>
<tr>
<td>Need disk?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Need terminal?</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Need printer?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Utilization Histograms

Uniprogramming

- CPU: 20%
- Memory: 33%
- Disk: 33%
- Terminal: 14%
- Printer: 14%

Elapsed Time: 30 minutes
Throughput: 6 jobs/hr
Mean Response Time: 18 min

Mulitprogramming

- CPU: 40%
- Memory: 67%
- Disk: 67%
- Terminal: 14%
- Printer: 14%

Elapsed Time: 15 minutes
Throughput: 12 jobs/hr
Mean Response Time: 10 min
Time Sharing

• Using multiprogramming to handle multiple *interactive* jobs
• Multiple users simultaneously access the system through terminals
• Processor’s time is shared among multiple users
Compatible Time-Sharing System (CTSS)

- First time-sharing system developed at MIT

Job Execution Sequence:

Job 1
Job 2
Job 3
Job 1
Job 4
Job 2

Figure 2.7 CTSS Operation
Major Achievements in Operating Systems

• Processes
• Memory Management
• Information protection and security
• Scheduling and resource management
• System structure
Processes

• 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
• A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources
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
Difficulties with Designing “Process-Based” System Software

• Improper synchronization
  – Ensure a process waiting for an I/O device receives the signal

• Failed mutual exclusion

• Nondeterminate program operation
  – Program should only depend on input to it, not on the activities of other programs

• Deadlocks
Memory Management

• Process isolation
  – Memory, data, instructions

• Automatic memory allocation and management
  – Transparent to users

• Support of modular programming
  – Define program modules: dynamic creation and destruction

• Protection and access control
  – Isolated and shared memory

• Long-term storage
  – Non-volatile, persistent storage
Virtual Memory

• Allows programmers to address memory from a logical point of view
• No hiatus between the execution of successive processes while one process was written out to secondary store and the successor process was read in
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

Main Memory

Main memory consists of a number of fixed-length frames.

Disk

Secondary memory (disk) can hold many fixed-length pages.
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

• Efficiency
  – Maximize throughput, minimize response time, and accommodate as many uses as possible
Key Elements of Operating System

![Key Elements of Operating System diagram](image)

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
Process Hardware 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
Process Hardware 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

• 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 operating system 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

Interleaving: Multiprogramming, Uniprocessor

Interleaving and Overlapping: Multiprogramming, 2 processors

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
Windows Architecture

- Modular structure for flexibility
- Executes on a variety of hardware platforms
- Supports application written for other operating system
Figure 2.13 Windows 2000 Architecture [SOLUTION]
Operating System Organization

• Modified microkernel architecture
  – Not a pure microkernel
  – Many system functions outside of the microkernel run in kernel mode

• Any module can be removed, upgraded, or replaced without rewriting the entire system
Kernel-Mode Components

- **Executive**
  - Contains base operating system services
    - Memory management
    - Process and thread management
    - Security
    - I/O
    - Interprocess communication

- **Kernel**
  - Consists of the most used components
Kernel-Mode Components

• Hardware abstraction layer (HAL)
  – Isolates the operating system from platform-specific hardware differences

• Device drivers
  – Translate user I/O function calls into specific hardware device I/O requests

• Windowing and graphics systems
  – Implements the graphical user interface (GUI)
Windows Executive

• I/O manager
• Cache manager
• Object manager
• Plug and play manager
• Power manager
• Security reference monitor
• Virtual memory manager
• Process/thread manager
• Configuration manager
• Local procedure call (LPC) facility
User-Mode Processes

• Special system support processes
  – Ex: logon process and the session manager
• Service processes
• Environment subsystems
• User applications
Client/Server Model

• Simplifies the Executive
  – Possible to construct a variety of APIs

• Improves reliability
  – Each service runs on a separate process with its own partition of memory
  – Clients cannot directly access hardware

• Provides a uniform means for applications to communicate via LPC

• Provides base for distributed computing
Threads and SMP

- Operating system routines can run on any available processor
- Different routines can execute simultaneously on different processors
- Multiple threads of execution within a single process may execute on different processors simultaneously
- Server processes may use multiple threads
- Share data and resources between process
Windows Objects

• Encapsulation
  – Object consists of one or more data items and one or more procedures

• Object class or instance
  – Create specified instances of an object

• Inheritance
  – Support to some extent in the Executive

• Polymorphism
UNIX

- Hardware is surrounded by the operating system software
- Operating system is called the system kernel
- Comes with a number of user services and interfaces
  - Shell
  - Components of the C compiler
UNIX

Figure 2.14 General UNIX Architecture
Figure 2.15 Traditional UNIX Kernel [BACH86]
Modern UNIX Kernel

Figure 2.16  Modern UNIX Kernel [VAHA96]
Modern UNIX Systems

- System V Release 4 (SVR4)
- Solaris 9
- 4.4BSD
- Linux