Operating System

A program that controls the execution of application programs

An interface between applications and hardware

Objectives:
- Convenience: makes the computer more convenient to use
- Efficiency: allows computer system resources to be used in an efficient manner
- Ability to evolve: permits effective development, testing, and introduction of new system functions without interfering with service

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

<table>
<thead>
<tr>
<th>Program development</th>
<th>Error detection and response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editors and debuggers</td>
<td>Internal and external hardware errors</td>
</tr>
<tr>
<td>Program execution</td>
<td>Memory error</td>
</tr>
<tr>
<td>Access to I/O devices</td>
<td>Device failure</td>
</tr>
<tr>
<td>Controlled access to files</td>
<td>Software errors</td>
</tr>
<tr>
<td>System access</td>
<td>Arithmetic overflow</td>
</tr>
</tbody>
</table>

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

### OS as Resource Manager

![Diagram of Computer System and I/O Devices]

Figure 2.2 The Operating System as Resource Manager
OS Kernel

Portion of operating system that is in main memory

Contains most frequently used functions

Also called the nucleus

Evolution of an Operating System

Serial Processing
  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

Memory protection: do not allow the memory area containing the monitor to be altered
- User program executes in user mode
  - Certain instructions may not be executed
- Monitor executes in system (kernel) mode
  - Privileged instructions are executed
  - Protected areas of memory may be accessed

Timer
- Prevents a job from monopolizing the system

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

Interrupts
  - Early computer models did not have this capability

I/O Devices are Slow

<table>
<thead>
<tr>
<th>Operation</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>TOTAL</td>
<td>31 μs</td>
</tr>
</tbody>
</table>

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

Figure 2.4 System Utilization Example
**Uni-programming Environment**

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

![Diagram showing Uni-programming Environment](image)

**Multi-programming Environment**

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

![Diagram showing Multi-programming Environment](image)
### Utilization Histograms

#### Figure 2.6 Utilization Histograms

- **CPU Utilization**
- **Memory Utilization**
- **Disk Utilization**
- **Terminal Utilization**
- **Printer Utilization**

#### Job History

- **JOB1**
- **JOB2**
- **JOB3**

(a) Uniprogramming

(b) Multiprogramming

### 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>
Time Sharing

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

CTSS: first time-sharing system developed at MIT

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

A 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 Representation

Figure 2.8 Typical Process Implementation
### Difficulties with Designing System Software

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improper synchronization</td>
</tr>
<tr>
<td>Ensure a process waiting for an I/O device receives the signal</td>
</tr>
<tr>
<td>Failed mutual exclusion</td>
</tr>
<tr>
<td>Nondeterminate program operation</td>
</tr>
<tr>
<td>Program should only depend on input to it, not on the activities of other programs</td>
</tr>
<tr>
<td>Deadlocks</td>
</tr>
</tbody>
</table>

### Memory Management

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process isolation</td>
</tr>
<tr>
<td>Automatic allocation and management</td>
</tr>
<tr>
<td>Support of modular programming</td>
</tr>
<tr>
<td>Protection and access control</td>
</tr>
<tr>
<td>Long-term storage</td>
</tr>
<tr>
<td>Virtual Memory</td>
</tr>
<tr>
<td>- Allows programmers to address memory from a logical point of view</td>
</tr>
<tr>
<td>- No hiatus between the execution of successive processes while one process was written out to secondary store and the successor process was read in</td>
</tr>
<tr>
<td>- Interacts with file system</td>
</tr>
<tr>
<td>Paging</td>
</tr>
<tr>
<td>- Allows process to be comprised of a number of fixed-size blocks, called pages</td>
</tr>
<tr>
<td>- Virtual address is a page number and an offset within the page</td>
</tr>
<tr>
<td>- Each page may be located anywhere in main memory</td>
</tr>
<tr>
<td>- Real address or physical address in main memory</td>
</tr>
</tbody>
</table>
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.

Disk

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

Figure 2.9 Virtual Memory Concepts

Virtual Memory Addressing

Processor

Virtual Management Unit

Real Address

Main Memory

Disk Address

Secondary Memory

Figure 2.10 Virtual Memory Addressing
### Information Protection and Security

<table>
<thead>
<tr>
<th>Security Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>Concerned with protecting the system against interruption</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Assuring that users cannot read data for which access is unauthorized</td>
</tr>
<tr>
<td>Data integrity</td>
<td>Protection of data from unauthorized modification</td>
</tr>
<tr>
<td>Authenticity</td>
<td>Concerned with the proper verification of the identity of users and the validity of messages or data</td>
</tr>
</tbody>
</table>

### Scheduling and Resource Management

<table>
<thead>
<tr>
<th>Resource Management Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fairness</td>
<td>Give equal and fair access to resources</td>
</tr>
<tr>
<td>Differential responsiveness</td>
<td>Discriminate among different classes of jobs</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Maximize throughput, minimize response time, and accommodate as many uses as possible</td>
</tr>
</tbody>
</table>
Key Elements of Operating System

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

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | Electronic circuits  
Objects are registers, memory cells, and logic gates  
Operations are clearing a register or reading a memory location |
| 2     | Processor’s instruction set  
Operations such as add, subtract, load, and store |
| 3     | Adds the concept of a procedure or subroutine, plus call/return operations |
| 4     | Interrupts |
| 5     | Process as a program in execution  
Suspend and resume processes |
| 6     | Secondary storage devices  
Transfer of blocks of data |
| 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

**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

**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

**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

- **Time**
  - Process 1
  - Process 2
  - Process 3

  (a) Interleaving (multiprogramming, one processor)

- Process 1
- Process 2
- Process 3

  (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

**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

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

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)

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

Figure 2.14 General UNIX Architecture
OS Overview

UNIX Kernel

OS Overview

Modern UNIX Kernel

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

Figure 2.15 Traditional UNIX Kernel [BACH86]

Figure 2.16 Modern UNIX Kernel [VAHA96]