Operating Systems
Overview

Chapter 2
Operating System

✓ A program that controls the execution of application programs
✓ An interface between the user and hardware
✓ Masks the details of the hardware
Layers and Views of a Computer System

End User

Application Programs

Utilities

Operating-System

Computer Hardware

Programmer

Operating-System Designer
Services Provided by the Operating System

✓ 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
    o memory error
    o device failure
  - software errors
    o arithmetic overflow
    o access forbidden memory locations
  - operating system cannot grant request issued by an application
Services Provided by the Operating System

✓ Accounting
  ▪ collect statistics
  ▪ monitor performance
  ▪ used to anticipate future enhancements
  ▪ used for billing users
Operating System

- It is actually a program
- Directs the processor in the use of system resources
- Directs the processor when executing other programs
- Processor stops executing the operating system in order to execute other programs
Operating System as a Resource Manager

Computer System

Memory

Operating System Software

Programs and Data

I/O Controller

I/O Controller

I/O Controller

Storage

O/S

Programs

Data

Stored data includes OS, Programs, Data
Evolution of an Operating System

- Hardware upgrades and new types of hardware
- New services
- Fixes
Monitors (an ancient type of OS)

- Software that controls the running programs
- Batch operating system
- Jobs are *batched* together
- Resident monitor is in main memory and available for execution
- Monitor utilities are *loaded* when needed
Memory Layout For a Resident Monitor

- Interrupt Processing
- Device Drivers
- Job Sequencing
- Control Language Interpreter

User Program Area

Monitor

Boundary
Job Control Language (JCL)

✓ Special type of programming language
✓ Provides instruction to the monitor about each job to be admitted into the system
  ▪ what compiler to use
  ▪ what data to use
  ▪ job priority
Hardware Features

✓ Memory protection
  - do not allow the memory area containing the monitor to be altered

✓ Timer
  - prevents a job from monopolizing the system
  - an interrupt occurs when time expires
Hardware Features

✓ Privileged instructions
  ▪ executed only by the monitor
  ▪ an interrupt occurs if a program tries these instructions

✓ Interrupts
  ▪ provide flexibility for controlling user programs
Multiprogramming or Multitasking

Allows the processor to execute another program while one program must wait for an I/O device.
## Example

<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>50K</td>
<td>100 K</td>
<td>80 K</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>
## Effects of Multiprogramming

<table>
<thead>
<tr>
<th></th>
<th>Uniprogramming</th>
<th>Multiprogramming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor use</td>
<td>17%</td>
<td>33%</td>
</tr>
<tr>
<td>Memory use</td>
<td>30%</td>
<td>67%</td>
</tr>
<tr>
<td>Disk use</td>
<td>33%</td>
<td>67%</td>
</tr>
<tr>
<td>Printer use</td>
<td>33%</td>
<td>67%</td>
</tr>
<tr>
<td>Elapsed time</td>
<td>30 min.</td>
<td>15 min.</td>
</tr>
<tr>
<td>Throughput rate</td>
<td>6 jobs/hr</td>
<td>12 jobs/hr</td>
</tr>
<tr>
<td>Mean response time</td>
<td>18 min.</td>
<td>10 min.</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
Process

✓ More general term than a job
✓ Consists of an executable program, associated data, and execution context (registers, program counter, etc.)
Difficulties with Designing OS Software

- Improper synchronization
  - ensure a process waiting for an I/O device receives the signal
- Failed mutual exclusion
- Non-deterministic program operation
  - program should only depend on its input, not on common memory areas
- Deadlocks
Memory Management

✓ Process isolation
✓ Automatic allocation and management
✓ Support for modular programming
✓ Protection and access control
✓ Long-term storage
Virtual Memory

✓ Allows programmers to address memory from a logical point of view
✓ While program is running portions of the program and data are kept in blocks on disk (transparently to the application).
File System

✓ Implements long-term storage
✓ Information stored in named objects, called *files*
Categories of Security and Protection

✓ Access control
  ▪ regulate user access to the system

✓ Information flow control
  ▪ regulate flow of data within the system and its delivery to users

✓ Certification
  ▪ proving that access and flow control perform according to specifications
Scheduling and Resource Management

✓ Fairness
  ▪ give equal and fair access to all processes

✓ Differential responsiveness
  ▪ discriminate between different classes of jobs

✓ Efficiency
  ▪ maximize throughput, minimize response time, and accommodate as many users as possible
Major Elements Of Operating System

Service Call from Process

Interrupt from Process
Interrupt from I/O

Service Call Handler

Interrupt Handler

Long-Term Queue
Short-Term Queue
I/O Queues

Short-Term Scheduler

Pass Control to Process
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 sub-problems
# Operating System Design Hierarchy

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Objects</th>
<th>Example Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Shell</td>
<td>User programming environment</td>
<td>Statements in shell language</td>
</tr>
<tr>
<td>12</td>
<td>User processes</td>
<td>User processes</td>
<td>Quit, kill, suspend, resume</td>
</tr>
<tr>
<td>11</td>
<td>Directories</td>
<td>Directories</td>
<td>Create, destroy, attach, detach, search, list</td>
</tr>
<tr>
<td>10</td>
<td>Devices</td>
<td>External devices, such as printer, displays and keyboards</td>
<td>Create, destroy, open, close, read, write</td>
</tr>
<tr>
<td>9</td>
<td>File system</td>
<td>Files</td>
<td>Create, destroy, open, close, read, write</td>
</tr>
<tr>
<td>8</td>
<td>Communications</td>
<td>Pipes</td>
<td>Create, destroy, open, close, read, write</td>
</tr>
</tbody>
</table>
## Operating System Design Hierarchy

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Objects</th>
<th>Example Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Virtual Memory</td>
<td>Segments, pages</td>
<td>Read, write, fetch</td>
</tr>
<tr>
<td>6</td>
<td>Local secondary store</td>
<td>Blocks of data, device channels</td>
<td>Read, write, allocate, free</td>
</tr>
<tr>
<td>5</td>
<td>Primitive processes</td>
<td>Primitive process, semaphores, ready list</td>
<td>Suspend, resume, wait, signal</td>
</tr>
</tbody>
</table>
## Operating System Design Hierarchy

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Objects</th>
<th>Example Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Interrupts</td>
<td>Interrupt-handling programs</td>
<td>Invoke, mask, unmask, retry programs</td>
</tr>
<tr>
<td>3</td>
<td>Procedures</td>
<td>Procedures, call stack</td>
<td>Mark stack, call, return</td>
</tr>
<tr>
<td>2</td>
<td>Instruction Set</td>
<td>Evaluation stack, micro-program interpreter, scalar and array data</td>
<td>Load, store, add, subtract, branch</td>
</tr>
<tr>
<td>1</td>
<td>Electronic circuits</td>
<td>Registers, gates, busses, etc.</td>
<td>Clear, transfer, activate, complement</td>
</tr>
</tbody>
</table>
Characteristics of Modern Operating Systems

✓ Microkernel architecture
  ▪ assigns only a few essential functions to the kernel
    o address space
    o interprocess communication (IPC)
    o basic scheduling
Characteristics of Modern Operating Systems

✓ **Multithreading**
  - process is divided into threads that can run simultaneously

✓ **Thread**
  - dispatchable unit of work
  - executes sequentially and is interruptible

✓ **Process** (sometimes also called *task*) is a collection of one or more threads
Characteristics of Modern Operating Systems

✓ **Symmetric multiprocessing (SMP)**
  - there are multiple processors
  - these processors share same main memory and I/O facilities
  - Each processor can perform the same functions (i.e., processors are interchangeable)
Characteristics of Modern Operating Systems

- Distributed operating systems
  - distinct computers organized into a single whole
  - provides the illusion of a single main memory
  - provides distributed file system that appears to the user as a single file space
Characteristics of 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 NT/.../XP/.../10/...

✓ Exploits the power of today’s microprocessors
✓ Provides full multitasking in a single-user environment
✓ Can execute on several hardware platforms
Windows NT/.../10/...

- Modified microkernel architecture
  - many of the system functions outside of the microkernel still run in kernel mode
- Hardware abstraction layer (HAL)
  - makes the hardware look the same to the kernel, regardless of the platform
  - provides support for symmetric multiprocessing
Windows NT/.../10/...

✓ The main parts of the OS are
  ▪ The Executive
    o Memory management, processes&threads (create, delete, track), I/O
  ▪ (micro) Kernel
    o Thread scheduling, process switching, interrupts, synchronization
  ▪ Hardware abstraction layer (HAL)
  ▪ Device drivers
  ▪ Window manager and graphics
✓ Executive is built using a client-server architecture
  ▪ Eg, memory management, process creation/deletion, scheduling are services that wait for requests
Advantages of Client/Server

✓ Simplifies the base operating system, the executive
  ▪ possible to construct a variety of APIs (application programming interface)

✓ Improves reliability
  ▪ each server runs as a separate process with its own partition of memory
  ▪ servers cannot directly access hardware

✓ Provides basis for distributed computing
Windows Features for SMP

- Operating system routines can run on any processor
- Different routines can execute simultaneously on different processors
- Multiple threads of execution within a single process
- Servers use multiple threads to process requests for more than one client simultaneously
- Flexible interprocess communication
Windows Objects

✓ Not all entities are objects
✓ Entities represented by objects are files, processes, threads, semaphores, times, and windows
✓ *Object manager* in the kernel is responsible for creating and destroying objects, for granting access to object’s services and data
UNIX Architecture
Linux, Solaris, Mac OS X

✓ Hardware is surrounded by the operating-system
✓ Operating system is called the *kernel*
✓ Comes with a number of user services and interfaces
  - shell
  - compilers
  - inter-process communication
✓ Symmetric multi-processing
✓ Some use microkernels (Solaris, Mac OS X); some are modular-monolithic (Linux)
UNIX Architecture