Operating Systems

Operating System Overview

Chapter 2
Outline

• Operating System Objectives and Functions
• Evolution of Operating Systems
• Major Achievements
• Characteristics of modern Operating Systems
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 of computer system layers]

**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 (shared system)
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 statistics
  – monitor performance
  – used to anticipate future enhancements
  – used for billing users
OS as a Resource Manager

• Functions same way as ordinary computer software
  – It is program that is executed

• OS directs the processor in the use of the other system resources and in the timing of its execution of other programs

• Operating system relinquishes control of the processor to execute other programs
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 and new types of hardware
- New services (new tools and new applications)
- Fixes (faults are discovered and fixes are made)
Evolution of Operating Systems

- **Serial Processing**
  - No operating system
  - Programmer interacted directly with hardware
  - Machines run from a console with display lights and toggle switches, input device (a card reader), and printer
  - **Scheduling** dilemma
  - **Setup** included loading the compiler, source program, saving compiled program, and loading and linking
Evolution of Operating Systems

• Simple Batch Systems
  – Monitors
    • Software that controls the running programs
    • Batch jobs together
    • Program constructed to branch back to monitor when finished
    • Resident monitor is in main memory and available for execution
    • *Handles the scheduling problem, and improves job setup time*
Job Control Language (JCL)

- Special type of programming language
- Provides instructions to the monitor
  - what compiler to use
  - Loading object programs into memory
  - what data to use
Desirable Hardware Features

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

• Timer
  – prevents a job from monopolizing the system
Problems with Simple Batch Systems

- Processor is often idle (I/O devices slow compared to processor)

Read one record from file: 0.0015 sec
Execute 100 instructions: 0.0001 sec
Write one record to file: 0.0015 sec
Total: 0.0031 sec

Percent CPU utilization: $\frac{0.0001}{0.0031} = 3.2\%$
Uniprogramming

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

(a) Uniprogramming
Multiprogramming

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

(b) Multiprogramming with two programs
(c) Multiprogramming with three programs
### Example

<table>
<thead>
<tr>
<th>JOB1</th>
<th>JOB2</th>
<th>JOB3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type of job</strong></td>
<td>Heavy compute</td>
<td>Heavy I/O</td>
</tr>
<tr>
<td><strong>Duration</strong></td>
<td>5 min.</td>
<td>15 min.</td>
</tr>
<tr>
<td><strong>Memory required</strong></td>
<td>50K</td>
<td>100 K</td>
</tr>
<tr>
<td><strong>Need disk?</strong></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Need terminal</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Need printer?</strong></td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Figure 2.6 Utilization Histograms
## Effects of Multiprogramming

<table>
<thead>
<tr>
<th></th>
<th>Uniprogramming</th>
<th>Multiprogramming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor use</td>
<td>22%</td>
<td>43%</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>

© Dr. Ayman Abdel-Hamid, OS
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
• OS interleaving execution of each user program in a short burst or *quantum* of computation
# Batch Multiprogramming versus Time Sharing

<table>
<thead>
<tr>
<th></th>
<th>Batch Multiprogramming</th>
<th>Time Sharing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal objective</td>
<td>Maximize processor use</td>
<td>Minimize response time</td>
</tr>
<tr>
<td>Source of directives</td>
<td>Job control language commands provided with the job</td>
<td>Commands entered at the terminal</td>
</tr>
<tr>
<td>to operating system</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Major Achievements

Theoretical advances in development of OS

- 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
- Factors contributing to development of process concept: multiprogramming batch operation, time sharing, and real-time transaction processing
Difficulties with Designing 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 relying on common memory areas

- Deadlocks
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 state)
  - Contents of various processor registers such as PC
  - Process priority
  - Whether the process is waiting for the completion of an I/O event
Process

Figure 2.8 Typical Process Implementation
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 one process is written out to secondary store and the successor process read in there in no hiatus
File System

- Implements long-term store
- Information stored in named objects called files
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 anywhere in main memory
- Real address or physical address in main memory
Main memory consists of a number of fixed-length frames, 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
© Dr. Ayman Abdel-Hamid, OS
Virtual Memory Addressing

Figure 2.10 Virtual Memory Addressing
Information Protection and Security

• **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 uses as possible
Major Elements of Operating System

Short-term scheduler: dispatcher

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
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>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 store</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>

**Resources of a single processor**
## 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 progs</td>
<td>Invoke, mask, unmask, retry</td>
</tr>
<tr>
<td>3</td>
<td>Procedures</td>
<td>Procedures, call stack, display</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, buses, etc.</td>
<td>Clear, transfer, activate, complement</td>
</tr>
</tbody>
</table>
An example: UNIX

Figure 2.15 General UNIX Architecture
© Dr. Ayman Abdel-Hamid, OS
Characteristics of Modern Operating Systems

- Microkernel architecture
  - assigns only a few essential functions to the kernel
    - address space
    - interprocess communication (IPC)
    - basic scheduling
  - Other OS services provided by processes (called servers) that run in user mode
  - Decouple kernel and server development
  - Servers may be customized to specific application or environment requirements
Characteristics of Modern Operating Systems

• Multithreading
  – process is divided into threads that can run simultaneously

• Thread
  – dispatchable unit of work
  – executes sequentially and is interruptable

• Process is a collection of one or more threads
Characteristics of Modern Operating Systems

• Symmetric multiprocessing
  – there are multiple processors
  – these processors share same main memory and I/O facilities
  – All processors can perform the same functions
Characteristics of Modern Operating Systems

• Distributed operating systems
  – provides the illusion of a single main memory and single secondary memory space
  – used for distributed file system
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
Traditional UNIX Kernel

![Diagram of Traditional UNIX Kernel]

**Figure 2.16** Traditional UNIX Kernel [BACH86]

© Dr. Ayman Abdel-Hamid, OS
Modern UNIX Kernel

Figure 2.17 Modern UNIX Kernel [VAHA96]
© Dr. Ayman Abdel-Hamid, OS