Operating System Introduction

Chapter 1

Computer System Structure

The major players

- Hardware
- Operating system
  - Allocates system resources
- Application programs
- Users
  - People
  - Machines
  - Other computers

Operating System Definition

No universally accepted definition

- OS is a resource allocator
  - Manages all resources
  - Decides between conflicting requests
- OS is a control program
  - Controls execution of programs to prevent errors and improper use of the computer
- “Everything a vendor ships when you order an operating system”
- “The one program running at all times on the computer”

Kinds Of Computers

Which computers need an OS?

- “Normal” big computers
- “Normal” little computers (PCs)
- Computer components
- Embedded
- Multimedia
- Handheld (including phones, etc.)

In The Beginning

Before operating systems

- There were computers and you “rolled your own” programs
- Libraries of common, repeated code-fragments grew
- The libraries represented universal services, which evolved into operating systems

What Is An Operating System?

A (master control) program (MCP)

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
- Use the computer hardware in an efficient manner

In The Beginning

Before operating systems

- There were computers and you “rolled your own” programs
- Libraries of common, repeated code-fragments grew
- The libraries represented universal services, which evolved into operating systems
Operating System Services

- Process management
- Memory management
- Storage management
  - Disk
  - File systems
- Security (internal and external)
- User interface
- Generalized device interfaces

A Brief History

- Batch processing systems through early 60’s
- Time-sharing systems
  - IBM System/360
  - MULTICS (1964-1970)
    - Multiplexed Information and Computing Service
  - UNIX (released outside of Bell Labs 1975)
- Microcomputers
  - monitors & disk-based OSes through 80’s
    - CP/M, DOS, Mac OS
  - “Real” OS mid to late 90’s

Operating System Role

- Hardware accessed through the OS
  - User interacts with the system through an interface
  - Provides a consistent environment despite hardware variations

Typical Microcomputer OS

- User may access at many different levels
  - Compare with Figure 1.1 and with Figure 2.11

Computer System Organization

- One or more CPUs, device controllers connect through common bus providing access to shared memory

Computer-System Operation

- I/O devices and CPU execute concurrently
  - Each device controller is in charge of a particular device type
    - Each device controller has a local buffer
    - CPU moves data from/to main memory to/from local buffers
    - I/O is from the device to local buffer of controller
    - Device controller informs CPU that it has finished its operation by causing an interrupt.
Interrupt Table

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt service routines</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1c</td>
<td>void power_fail() { . . . }</td>
</tr>
<tr>
<td>0x18</td>
<td>void read_done() { . . . }</td>
</tr>
<tr>
<td>0x14</td>
<td>void get_char() { . . . }</td>
</tr>
<tr>
<td>0x10</td>
<td>void clock_tick() { . . . }</td>
</tr>
<tr>
<td>0x0c</td>
<td>void sys_call(int c) { . . . }</td>
</tr>
<tr>
<td>0x08</td>
<td>word</td>
</tr>
<tr>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

Interrupt Handlers

- Small and fast
- Reentrant functions
  - Cannot access static/global data
  - Cannot return a pointer to static/global data
  - Operate only on local data (including parameters)
  - Cannot call non-reentrant functions
- Disable interrupts while processing
- A trap or system call is a software interrupt
- Operating systems are interrupt driven

Dual Mode Operation

- User mode
  - Some instructions not available
- Supervisor, kernel, system, or privileged mode
  - Able to execute privileged instructions
- Triggered by hardware or software interrupt

I/O

- Synchronous
- Asynchronous

Direct Memory Access

- High-speed I/O
- Controller transfers blocks of data w/o CPU intervention
- Only one interrupt per block

Memory Hierarchy

- Multi-level to satisfy multiple needs
- Primary: volatile, random access
- Secondary: non-volatile, random access
- Tertiary: non-volatile, sequential access

Cost & Speed
## Caching

Based on "locality of reference"

- Performed at many levels
- Data in use temporarily copied from slower to faster storage
- Faster storage (cache) checked
  - Cache hit: data used from cache (fast)
  - Cache miss: data copied to cache then used
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

## Multiprogramming

Needed for efficiency

- Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
    - When it has to wait (for I/O for example), OS switches to another job
- Keep the computer (CPU) busy!!!!

## Multitasking (Time Sharing)

Similar to concurrent programming

- Extension of multiprogramming
  - CPU switches jobs quickly and so frequently
    - Users can interact with each job while it is running
    - Illusion of a dedicated system
  - Response time (context switch) < 1 second
  - Each task allotted a time-slice
  - Each user has at least one runnable process
  - Several jobs ready to run, OS schedules CPU
  - OS swaps processes in and out of memory
  - Virtual memory allows execution of processes not completely in memory (locality of reference)

## People vs Computer Cost

Comparing processing styles

- Multiprogramming, batch operated was efficient
- Multitasking, multiuser (time sharing) is efficient

## Computer Organization

### Flynn’s Taxonomy

<table>
<thead>
<tr>
<th></th>
<th>Single Instruction</th>
<th>Multiple Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Data</td>
<td>SISD</td>
<td>MISD</td>
</tr>
<tr>
<td>Multiple Data</td>
<td>SIMD</td>
<td>MIMD</td>
</tr>
</tbody>
</table>

- SPMD
  - Single program multiple data
  - Similar to SIMD but not synchronous

- Michael J. Flynn (1966)

## SISD

- “Ordinary” computers (until recently)
- Although still considered SISD
  - CPUs have pipelined architectures
  - Graphics consoles and game consoles include array-processing (SIMD)
  - “Household” computers are hyper-threaded and have multiple cores
What does it mean to execute more than one instruction on one chunk of data?

- Uncommon
  - Redundant, high-risk controllers
    - Aircraft
    - Nuclear power
    - Chemical

### MISD
Multiple instruction, single data

- One program
  - Synchronous operation
  - Array processors: super computers (60's - 80's)
  - Scientific, engineering, and modeling
  - E.g., weather forecasting

### SIMD
Single instruction, multiple data

- Multiple instruction, single data

- True parallel computer
  - Multiple processing elements
  - Different programs different data

### MIMD
Multiple instruction, multiple data

- True parallel computer
  - Multiple processing elements
  - Different programs different data

### Interconnection Networks
Speed vs Cost

- Shared Bus
  - Slow but cheap

- Interconnection Network
  - Fast but expensive
  - Medium speed and expense