CMSC 412
Fall 2005
Computer and Operating System Structures

Announcements

- Project #0
  - Due Friday
  - Submission program not yet working (tomorrow hopefully)
- Project #1
  - Much harder than Project #0!
  - Posted Friday
- Reading
  - Chapter 1 and 2 (today)
  - Chapter 2 and 3 (Monday)
Computer Systems

I/O Subsystem Structure

- Many different types of devices
  - disks
  - networks
  - displays
  - mouse
  - keyboard
- Each has different peak performance
  - bandwidth
    - rate at which data can be moved
  - latency
    - time from request to first data back
Performing I/O

• Synchronous
  - OS issues I/O request, and waits for it to complete before continuing.

• Asynchronous
  - OS issues request, and then does something else. Device generates interrupt to notify OS of completion.

• Direct Memory Access (DMA)
  - A kind of async I/O, but for larger blocks rather than single characters/bytes

Storage Structure

• Main memory - only large storage media that the CPU can access directly
  - DRAM, SRAM (Caches)

• Secondary storage - extension of main memory that provides large nonvolatile storage capacity
  - Disks (floppy disks, hard disk, optical disk)
  - Tape drives, Zip drives
  - Flash RAM (somewhere in between?)
Storage Hierarchy

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility

- Caching - copying information into faster storage system; main memory can be viewed as a last cache for secondary storage.

Storage-Device Hierarchy
Caching

- Use of high-speed memory to hold recently-accessed data.
- Requires a *cache management* policy.
- Caching introduces another level in storage hierarchy. This requires data that is simultaneously stored in more than one level to be *consistent*.

Migration of A From Disk to Register

![Diagram showing migration of A from magnetic disk to main memory, then to cache, and finally to hardware register.](image)
Operating System Services

- Program execution
  - Load a program into memory and to run it.
- I/O operations
- File-system manipulation
- Communications
  - Between local or distributed processes
  - Either *shared memory* or *message passing*.
- Error detection and recovery
  - In the CPU and memory hardware, in I/O devices, or in user programs.

Additional OS Functions

- Resource allocation
  - Allocating resources to multiple users or multiple jobs running at the same time.
- Accounting
  - Keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection
  - Controlling all access to system resources.
Common OS Components

- Process Management
- Main Memory Management
- File Management
- I/O System Management
- Secondary Storage Management
- Networking
- Protection System
- Command-Interpreter System

Process Management

- A process is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - Process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication
Main-Memory Management

- **Memory** is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is *volatile* storage.
  - It loses its contents on power-loss.
- OS memory-related activities:
  - Track which parts of memory are being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space.

File Management

- A *file* is a collection of related information defined by its creator. Commonly, files represent programs and data.
- OS file-related activities:
  - Provide primitives to create, delete, and manipulate files (and directories).
  - Map files onto secondary storage.
I/O System Management

- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

Secondary-Storage Management

- Secondary is storage is a *nonvolatile* backup to main memory.
- Most modern computer systems use disks.
- Secondary-storage-related OS functions:
  - Free space management
  - Storage allocation
  - Disk scheduling
Networking

• A distributed system is a collection processors that do not share memory or a clock, connected through a communication network.
  - Communication takes place using a protocol.

• Networking-related OS activities
  - Managing protocol state
  - Managing states of distributed resources (e.g. remote filesystem)

Protection System

• Protection refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.

• The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.
Key Mechanism: System Calls

- A software-generated interrupt
  - a.k.a. *trap*
- Provide the interface between application programs and the OS kernel
- Are like procedure calls
  - take parameters
  - calling routine waits for response

System Call Mechanism

- Use numbers to indicate what call is made
- Parameters stored in registers or the stack
- Why do we use system call numbers rather than directly calling a kernel subroutine?
  - permits changing the size and location of system call implementations without having to re-link application programs
GeekOS and x86

- Intel system call instruction
  - int \( n \) where \( n \) is the interrupt vector #
  - “call kernel routine \( n \)”
  - vectors 0-31 reserved
    - Page fault, segmentation violation, etc.
- GeekOS
  - All system calls set \( n \) as 90
  - System call number stored in eax register

Types of System Calls

- File Related
  - open, create, read, write, close, delete
  - get or set file attributes
- Informational
  - get time
  - set system data (OS parameters)
  - get process information (id, time used)
- Communication Related
  - establish a connection; terminate a connection
  - send, receive messages
- Process control
  - create/terminate a process (including self)
Use of a System Call for I/O

Why use system calls at all?

- Why not “link” application programs against the kernel and call kernel routines directly?
Protection

• Need to protect OS from user programs and user programs from each other
  - Don’t want a bug in a user program to crash the whole machine (as in earlier OSs, like MS-DOS, MacOS, Windows 3.1, and others)

• Hardware resources of interest
  - Memory, I/O devices, CPU

Dual-Mode Operation

• Provide hardware support to differentiate between at least two modes of operations.
  1. *User mode* - execution for a user.
  2. *Monitor mode* (also *kernel mode* or *system mode*) - execution done on behalf of operating system.

• Operations in user mode a subset of those allowed in monitor mode
  - Privileged instructions only in monitor
Dual-Mode Operation

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1). X86 actually has 4 modes (2 bits).
- When an interrupt or fault occurs hardware switches to monitor mode.

```
monitor
```

interrupt/fault

```
user
```

set user mode

I/O Protection

- System call mechanism prevents user mode programs from accessing devices directly.
  - All I/O instructions are privileged
- But what if user program can overwrite interrupt handler with its own code?
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - Base register - holds the smallest legal physical memory address.
  - Limit register - contains the size of the range
- Memory outside the defined range is protected.

Base and Limit Registers
Hardware Address Protection

Changing the base and limit registers are privileged operations

CPU Protection

- **Timer** - interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Commonly used to implement time sharing.
- Load-timer is a privileged instruction.