Chapter 1: Introduction

Saturday, November 03, 2007
5:09 PM

- Operating system (OS) = program that manages computer hardware.

- Computer system
  - Hardware
    - CPU
    - Memory
    - I/O devices
  - Operating system
  - Application programs = oriented to solve users' problems
  - Users

- OS' roles
  - Resource allocator
  - Control program
    - Prevent errors and improper use of the computer.

- **Kernel** = the one program that runs all the time. (This definition might not be correct. If the computer is in user mode, then the kernel doesn't run at that time.)
  - Provides the lowest level of abstraction for resources.

- Computer system structure.
  - Computers are generally devices connected through a **bus**.
  - We use **von Neumann** architecture.
    - Programs are stored as data in memory.
    - To execute program, the CPU loads instruction as data, and interpret it.
    - The CPU keeps track of the current instruction using the **instruction register** or **program counter** or **instruction pointer**.
  - Storage structure
    - CPU have fast **registers**.
    - Main memory is the memory that CPU can access directly.
      - Programs has to be in main memory to be executable.
      - Typically implemented using **semiconductor memory**.
      - Is accessed randomly. (Hence, Random-access Memory (RAM).)
      - Fast, but not as fast as registers, and is very expensive.
      - Is typically *volatile.*
    - Secondary storage.
      - Usually **magnetic disk**.
      - Non-volatile.
      - Slow, but inexpensive.
      - Offers more space.
      - Programs are stored in disk until it is to be executed. In that case, it is loaded to main memory.
    - Tertiary storage
      - CD-ROM, DVD-ROM, and tapes.
    - Can be arranged into a hierarchy such that, the closer to the CPU a memory is, the fast, the smaller, and the more expensive it is.
  - I/O
    - Connected to the CPU via a bus.
    - A **device driver** talks to the I/O device, and provides an interface of the device to the user.
    - Device drivers sends interrupt to CPU if something happens.

- Operating system abilities

    - **Multiprogramming** = increasing CPU utilization by organizing jobs so that CPU always has one to execute.
        - Example: Most program will wait for interrupts. While the program waits, the OS can run another task.
    - **Time sharing** = execute multiple jobs by rapidly switching among them. Gives the illusion that all of them are running concurrently. This allows user to interact with programs at interactive rate.
        - Example: The OS needs to switch context very often to prevent starvation.
    - **Job scheduling** = select one job from the *disk* to load to the memory and execute. The jobs that lie in the disk are said to be in a **job pool**.
    - **CPU scheduling** = select one job from the jobs in main memory to execute.
        - A program that is loaded into the memory and is executing is called a **process**.
    - **Virtual memory** = gives each process an illusion that it has a contiguous piece of memory all to itself.
    - **Fault isolation** = one process' failure should not affect others.

- Operating system operation
    - The operation is **interrupt-driven** in nature.
        - When an event occur, hardware fires an interrupt to the CPU.
        - Software can also fires interrupt via **system calls**.
        - A **trap** is an interrupt generated by (1) errors or (2) software.
        - When the CPU receives an interrupt:
            - CPU's execution is stopped.
            - The CPU's state is saved.
            - The control transfers to a fixed location.
            - The code at this fixed location dispatches the control to the interrupt service routine.
            - Once the routine finished, the control transfers to the user program as if the interrupt never occurred.
        - Table of pointers that store addresses to interrupt service routines is called **interrupt vectors.**
    - Dual mode operation
        - Modern hardware support two modes of CPU execution
            - User mode
            - Kernel mode (or supervisor mode)
        - There's a bit in the CPU register that indicates this.
        - Some instructions are only executable in kernel mode. These are called **privileged instructions.** For example, setting various flags, controlling the counter, or output to ports.
        - User programs are executed while the CPU is in user mode. So it cannot manage hardware by itself. On the other hand, kernel is executed in kernel mode, so it can do everything.
        - If the user program wants to do something not in its power, it requests the kernel to do so via a **system call**.
        - A system call is implemented as a software interrupt in Intel 80x86. Some chips have built-in syscall instruction.
        - After the system call is invoked, the CPU switches into kernel mode. The interrupt is handled by the interrupt service routine for that particular interrupt. This interrupt service routine parses additional information that comes with the system call to determine what the user program want to get served, and gives appropriate service. It then transfers the control back to the user program.
    - Timer
        - To prevent a process from hogging CPU cycles, the OS makes uses of the timer that comes with the CPU.
        - The OS would set the timer to fires an interrupt every now and then. Typically every 10ms.
        - The timer interrupt would cause the OS to run the scheduler, and switches to another process if appropriate.
        - This is used to achieve time sharing. Also called **preemptive scheduling.**

- What do operating system do?
    - Process management
        - Gives the illusion that each process has the CPU to itself.
        - Creating and deleting processes
        - Suspending and resuming processes
        - Providing mechanisms for process synchronization
        - Providing mechanisms for process communications
        - Providing mechanisms for deadlock handling
    - Memory management
        - Gives the illusion that each process has a contiguous array of memory to itself.
        - Keeping track of which parts of memory are currently being used and by whom.
        - Deciding which processes and data to move into and out of memory (job scheduling and virtual memory)
        - Allocating and deallocating memory
    - File system management
        - File system is the abstraction of the storage.
        - File = collection of related information defined by the creator.
        - Create and delete files
        - Create and delete directories
        - Provide primitives for manipulating files and directories
        - Mapping files onto secondary storage
        - Backing up files on nonvolatile storage media
    - Caching
        - Cache = a fast memory that is used to mirror data from slower memory (or computation process) to improve efficiency.
        - Compiler manages register (cache of L1 and L2 cache)
        - CPU manages L1 and L2 cache (cache of main memory)
        - OS manages the disk cache (cache of secondary storage)
        - In multi-processor environment, have to maintain **cache coherency** = the caches at different processors have to contain the same value.
    - I/O System Management
        - Buffering, caching, and spooling for I/O devices.
        - General driver interface.
        - Drivers for specific hardware devices.
    - Protection and security
        - **Protection** = controlling access to system resources
            - Example: preventing processes from accessing other processes memory or files.
        - **Security** = defend system from internal and external attacks
        - Both require system to be able to distinguish users from non-users. Most implement some kind of **user ID**, and **group ID.**

- Network operating system
    - **Network** = communication path between two systems. Varies by protocols used, distance between nodes, and the transport media.
    - **Network operating system** = operating system that provide network and distributed system functions such as distributed file system, and a communication protocol.
    - Network OS might provide an illusion that all the systems across the network are running a single OS.

- Real-time operating system
    - Must finish some tasks before a deadline. Otherwise, the system will fail.
    - This is different from OS in time-sharing system, which must only ensure that the users get quick responses.

- Handheld operating system
    - Must manage scarce memory.
    - Must cope with slow processor.

- Must take power into account.