Chapter 1

Terms

- CPU - hardware that executes instructions
- Processor - physical chip that contains one or more CPU's
- Core - basic computation unit of the CPU
- Multicore - includes multiple computing cores on the same CPU
- Multiprocessor - includes multiple processors

General Organization of Computer System

- One or more CPU's and device controllers connected through a bus
- Each controller is in charge of a specific type of device - more than one device can be attached depending on the controller
    - A controller moves data between peripheral devices it controls and its local buffer storage
    - eg. disk drive, audio device
- Typically, every device controller has a device driver
    - Driver provides rest of OS with with an interface to the device

Role of Interrupts

- Controllers use interrupts to inform the device driver that an operation has finished
- When an interrupt is generated (travels through the system bus), the CPU stops what it is doing and transfers execution to a fixed location (location contains the starting address where the interrupt service routine is located)
- Once the interrupt is dealt with, CPU resumes what it was doing before
- Vector table is used to determine which ISR to run

<u>Implementation</u>

- CPU hardware has an interrupt-request line wire
- When it detects that a controller placed a signal on the wire, it reads the interrupt number and jumps to the interrupt-handler routine
- We say that a drive raises an interrupt by placing a signal on the wire, the CPU catches the interrupt and dispatches it to interrupt handler, and the handler clears the interrupt by servicing the device
- Modern OS's need:
    - The ability to defer interrupt handling during critical processing
    - An efficient way to dispatch to the proper interrupt handler for a device
    - multilevel interrupts, so that the operating system can distinguish between high- and low-priority interrupts and can respond with the appropriate degree of urgency
    - These features are provided by the CPU and interrupt-controller hardware

Components in a Modern Multiprocessor Computer System

- Two or more processors, each with a single core CPU

-

Transition from User Mode to Kernel Mode

- Aka user mode and processor mode (using a state/mode bit)
- Hardware boots in kernel mode, then loads OS and user programs in user mode

Chapter 2

Terms

Services Provided by OS

- User interface - usually a GUI with a mouse pointer for selecting shit
    - On mobile devices, the UI is in the form of a touch-screen interface
    - Some provide a command-line interface
- Program execution - system must be able to load a program into memory and be able to run that program
    - Program must be able to end
- I/O operations - a running program may require I/O
    - For protection and efficiency, the user usually cannot control I/O devices directly, so the OS must provide the means to do so
- File-system manipulation
- Communications - many circumstances where one process needs to communicate with another
    - Can be done through shared memory or message passing
- Error detection

The above exist to help the user, the rest is for efficient operation

- Resource allocation
- Logging - keeping track of which programs use how much and what kinds of resources
- Protection and security

System Calls

- Can be grouped into 6 categories:
    - Process control
        - create/terminate process
        - Load, execute
        - get/set process attributes
        - Wait event, signal event
        - allocate/free memory
    - File management
        - create/delete file
        - open/close file
        - Read, write, reposition

        - get/set file attributes
    - Device management
        - request/release device
        - Read, write, reposition
        - get/set device attributes
        - Logically attach or detach devices
    - Information maintenance
        - get/set time or date
        - get/set system data
        - Get process, file, or device attributes
        - Set process, file, or device attributes
    - Communications
        - create/delete communication connection
        - send/receive messages
        - Transfer status info
        - attach/detach remote devices
    - Protection
        - set/set file permissions

Operating System Structure

- Monolithic Structure (“tightly coupled” - meaning changes to one part can have large effects on the rest)
    - No structure at all
    - Place all functionality into a single, static binary file that runs in a single address space
- Layered Approach (“loosely coupled” - changing one part only changes that part and has no effect on other parts)
    - OS is broken into a number of layers
    - Bottom layer (layer 0) is the hardware and the top layer (layer N) is the UI
- Microkernels
    - Removes all nonessential components from the kernel and implements them as user-level programs that reside in a separate address space
    - Results in a smaller kernel
- Modules
    - Use loadable kernel modules (LKMs)
    - Kernel has set of core components and can link additional services via modules (either at boot time or during run time)
- Hybrid systems
    - Very few OS’s adopt only one of the above structures
    - Usually a combination

Chapter 4 - Threads

Overview

- Threads consist of: thread ID, PC, register set, and a stack

Motivation

- Instead of creating a new process to service a request (which runs the same shit as the original process but just has added overhead) → it's more efficient to just create another thread

Types of Parallelism

- Data parallelism - focuses on distributing subsets of the same data across multiple computing cores and performing the same operation on each core
- Task parallelism - involves distributing not data but tasks (threads) across multiple computing cores

Threading Issues

- fork() and exec() system calls change a little with multi-threaded programs
    - eg. for fork() does the new process duplicate all threads or stay single-threaded?
- Signal handling - when a process has multiple threads, where should the signal go? (4 options)
    - Deliver signal to thread to which signal applies
    - Deliver signal to every thread
    - Deliver signal to certain threads
    - Assign one thread to receive all signals

Thread Cancellation

- Involves terminating a thread before it has completed
- A thread to be cancelled is often referred to has a **target thread**
- Threads can be cancelled in two ways:
    - Asynchronous cancellation - one thread immediately terminates the target thread
    - Deferred cancellation - target thread periodically checks whether it should terminate