

# Computer System Organization

CS 256/456

Dept. of Computer Science, University of Rochester

1/22/2007

CSC 256/456 - Spring 2007

1

## Overview

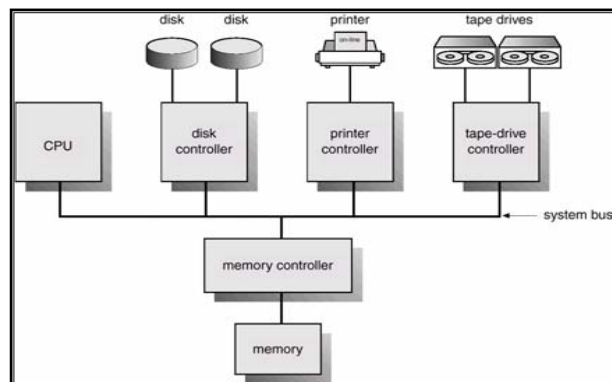
- Recap of last class
  - What is operating system (architecturally)?
  - Functionalities of operating system?
  - Types of operating systems.
- Computer hardware issues
  - Hardware interface to OS
  - Hardware support for OS mechanisms
- Operating system organization
  - OS components
  - OS architectures

1/22/2007

CSC 256/456 - Spring 2007

2

## Computer-System Architecture

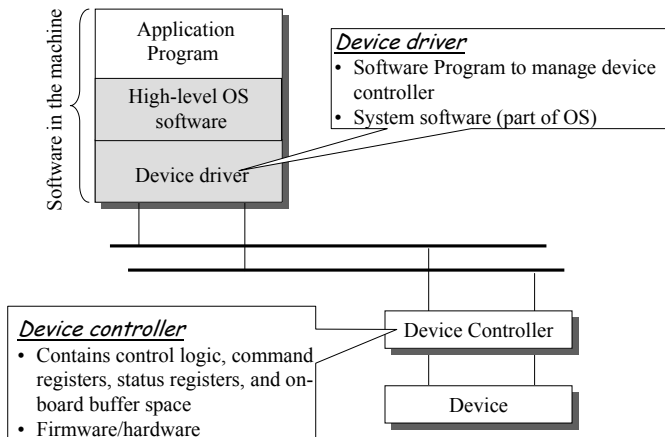


1/22/2007

CSC 256/456 - Spring 2007

3

## The Device-Controller-Software Relationship



1/22/2007

CSC 256/456 - Spring 2007

4

## I/O Operations

- How is I/O done?
  - I/O devices are much slower than CPU
- Synchronous (polling)
  - After I/O starts, busy polling the device status register until it shows the operation completes.
- Asynchronous (interrupt-driven)
  - After I/O starts, control returns to user program without waiting for I/O completion.
  - Device controller later informs CPU that it has finished its operation by causing an *interrupt*.
  - When an interrupt occurs, current execution is put on hold; the CPU jumps to a service routine called "interrupt handler".

1/22/2007

CSC 256/456 - Spring 2007

5

## System Protection

- User programs (programs not belonging to the OS) are generally not trusted
  - A user program may use unfair amount of resource
  - A user program may maliciously cause other programs or the OS to fail
- Need protection against untrusted user programs; the system must differentiate between at least two modes of operations
  1. *User mode* - execution of user programs
    - untrusted
    - not allowed to have complete/direct access to hardware resources
  2. *Kernel mode* (also *system mode* or *monitor mode*) - execution of the operating system
    - trusted
    - allowed to have complete/direct access to hardware resources
- Hardware support is needed in such protection

1/22/2007

CSC 256/456 - Spring 2007

6

## Memory Protection

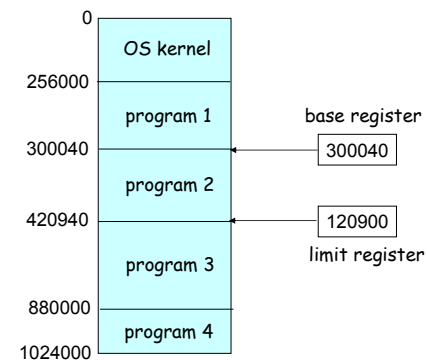
- Goal of memory protection?
  - A user program can't use arbitrary amount of memory
  - A user program can't access data belonging to the operating system or other user programs.
- How to achieve memory protection?
  - Indirect memory access
    - Memory access with a virtual address which needs to be translated into physical address
  - 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.

1/22/2007

CSC 256/456 - Spring 2007

7

## Hardware Address Protection



- Address of each memory address is checked against "base" and "base+limit"
- Trap to the OS kernel if it falls outside of the range (an exception)

1/22/2007

CSC 256/456 - Spring 2007

8

## Protection of I/O Devices

- User programs are not allowed to directly access I/O devices
  - Special I/O instructions can only be used in kernel mode
  - Controller registers can only be accessed in kernel mode
- So device drivers, I/O interrupt handlers must run in kernel mode.
- User programs perform I/O through requesting the OS (using system calls).

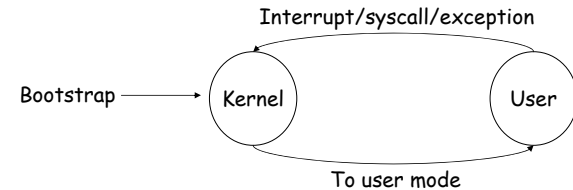
1/22/2007

CSC 256/456 - Spring 2007

9

## Transition between User/Kernel Mode

- When does the machine run in kernel mode?
  - after machine boot
  - interrupt handler
  - system call
  - exception

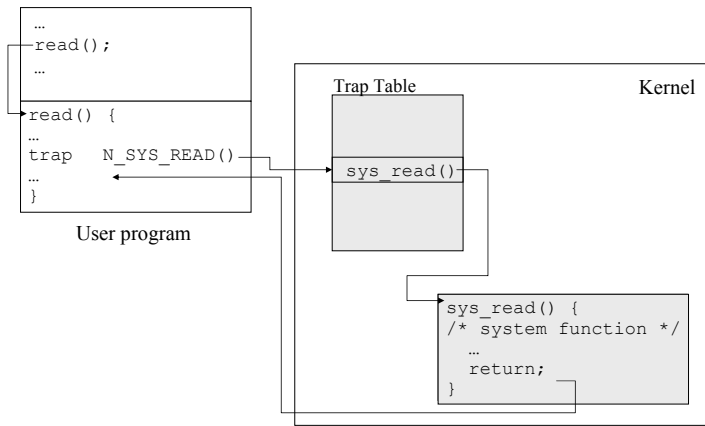


1/22/2007

CSC 256/456 - Spring 2007

10

## System Call Using the Trap Instruction



1/22/2007

CSC 256/456 - Spring 2007

11

## CPU Protection

- Goal of CPU protection
  - A user program can't hold the CPU for ever
- *Timer* - interrupts computer after specified period to ensure the OS kernel maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
  - CPU time sharing is implemented in the timer interrupt.

1/22/2007

CSC 256/456 - Spring 2007

12

## Operation System Organization

- System Components
  - process management
  - memory management
  - I/O system
  - file and storage
  - networking, ...
- Operating System Architectures
  - monolithic architecture
  - microkernel architecture
  - layered architecture
  - virtual machines

## Process Management

- A *process* is a program in execution.
  - Resource principal - A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
  - Protection domain.
- OS responsibilities for process management:
  - Process creation and deletion.
  - Process scheduling, suspension, and resumption.
  - Process synchronization, inter-process communication

## Memory Management

- **Memory**
  - A large array of addressable words or bytes.
  - A data repository shared by the CPU and I/O devices.
- OS responsibility for memory management:
  - Allocate and deallocate memory space as requested.
  - Efficient utilization when the memory resource is heavily competed.
  - Keep track of which parts of memory are currently being used and by whom.

## I/O System Management

- A computer needs I/O to interact with outside world:
  - Console/terminal
  - Non-volatile secondary storage - disks
  - Networking
- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

## File and Secondary Storage Management

- A file is a collection of information defined by its user. Commonly, both programs and data are stored as files.
- OS responsibility for file management:
  - Manipulation of files and directories.
  - Map files onto (nonvolatile) secondary storage - disks.
- OS responsibility for disk management:
  - Free space management and storage allocation.
  - Disk scheduling.
- They are not all always together
  - Not all files are mapped to secondary storage!
  - Not all disk space are used for the file system!

1/22/2007

CSC 256/456 - Spring 2007

17

## Networking (Distributed Systems)

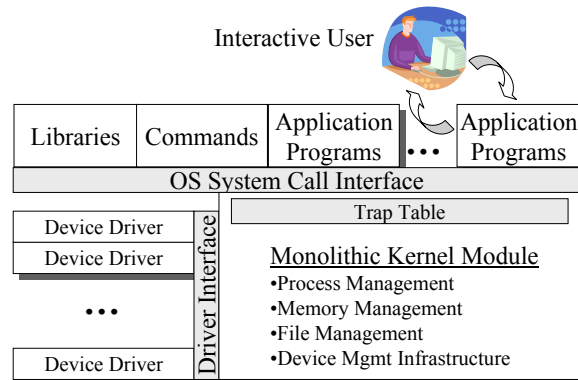
- A *distributed system*
  - A collection of processors that do not share memory.
  - Processors are connected through a communication network.
  - Communication takes place using a *protocol*.
- A distributed system provides user access to various shared system resources, which allows:
  - Computation speed-up
  - Enhanced reliability
- How much of it is in kernel space?

1/22/2007

CSC 256/456 - Spring 2007

18

## OS Architecture: Monolithic Structure



Most modern OSes fall into this category!

1/22/2007

CSC 256/456 - Spring 2007

19

## Microkernel System Architecture

- Microkernel architecture:
  - Moves as much from the kernel into "user" space (still protected from normal users).
  - Communication takes place between user modules using message passing.
- What must be in the kernel and what can be in user space?
  - Mechanisms determine how to do something.
  - Policies decide what will be done.
- Benefits:
  - More reliable (less code is running in kernel mode)
  - More secure (less code is running in kernel mode)
- Disadvantage?

1/22/2007

CSC 256/456 - Spring 2007

20

## Layered Structure

- Layered structure
  - The operating system is divided into a number of layers (levels), each built on top of lower layers.
  - The bottom layer (layer 0), is the hardware.
  - The highest (layer N) is the user interface.
  - Decreased privileges for higher layers.
- Benefits:
  - more reliable
  - more secure
  - more flexibility, easier to extend
- Disadvantage?
  - Weak integration results in performance penalty (similar to the microkernel structure).

## Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).