CMSC 412
Fall 2004

Operating Systems Structures

Announcements

- Project #1
  - Posted. Due Friday.
- Reading
  - Chapter 3, 4

Common System 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

4

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.

Command-Interpreter System

- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking

Command-Interpreter System

- The program that reads and interprets control statements is called variously:
  - command-line interpreter
  - shell (in UNIX)

Its function is to get and execute the next command statement.

Operating System Services

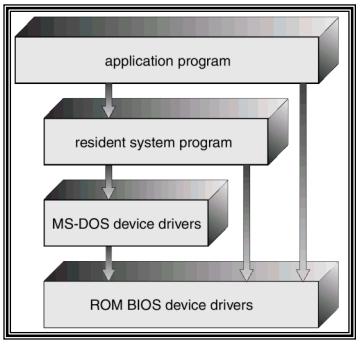
- Program execution
  - Load a program into memory and to run it.
- I/O operations
- File-system manipulation
- Communications
  - Between local or distributed processes
  - Either *shared memory* or *message passing*.
- Error detection and recovery
  - In the CPU and memory hardware, in I/O devices,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

OS Structure: Simple

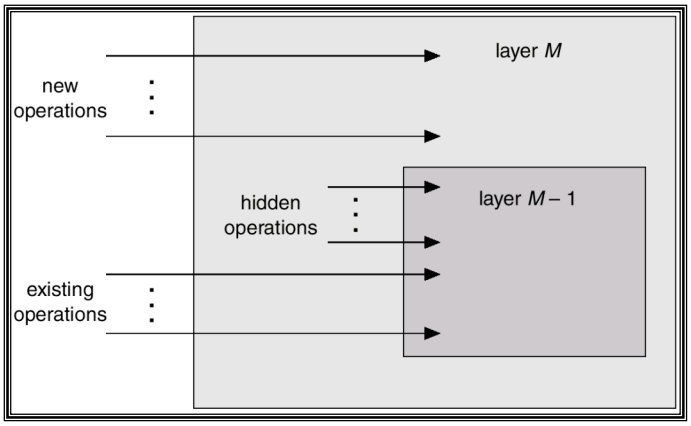
- Any part of the system may use the functionality of the rest of the system
  - MS-DOS (user programs can *call* low level I/O routines)



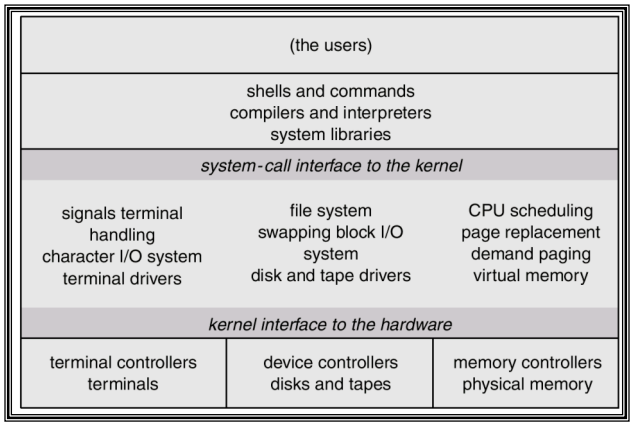
OS Structure: Layered

- Layer *n* can only see the functionality that layer *n-1* exports
  - *abstracts* the lower level details
    - new hardware can be added if it provides the interface required of a particular layer
  - system call interface is an example of layering

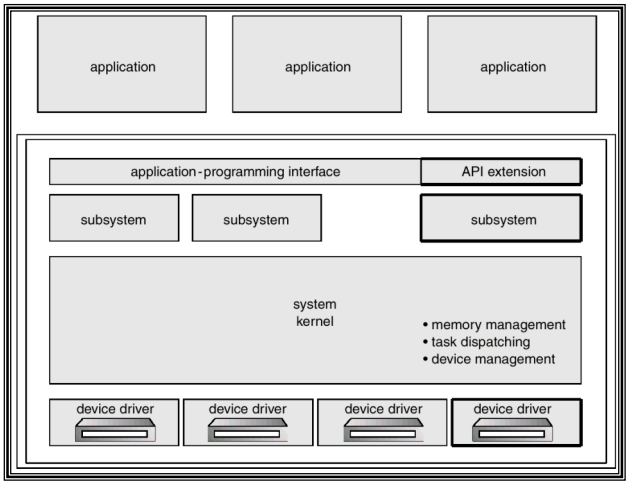
OS Structure: Layered



Example: UNIX



Example: OS/2



OS Structure: Hybrid

- Layering has problems on its own
  - May be difficult to communicate information easily
  - May be inefficient
- Hybrid
  - Use layered, modular approach, but permit "back doors" to improve information flow at the cost of abstraction

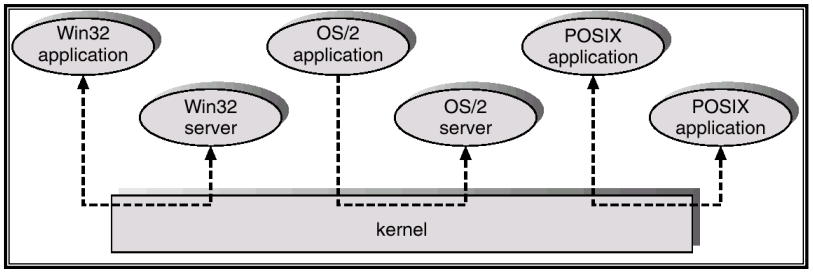
Policy vs. Mechanism

- Policy - what to do
  - users should not be able to read other users files
- Mechanism- how to accomplish the goal
  - file protection properties are checked on open system call
- Want to be able to change policy without having to change mechanism
  - change default file protection
- Extreme examples of each:
  - micro-kernel OS - all mechanism, no policy
  - MacOS - policy and mechanism bound together

Microkernel System Structure

- Goal: make the kernel as small as possible.
- Communication takes place between user modules using message passing via the kernel.
- Benefits:
  - easier to extend a microkernel
  - easier to port the OS to new architectures
  - more reliable and secure (less code running in kernel mode)
- Drawback:
  - More overhead for operation

Windows NT Client-Server Structure



12