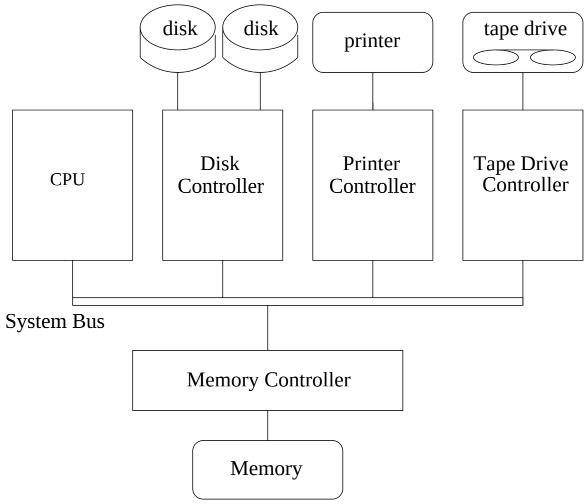
Last Class: OS and Computer Architecture



Last Class: OS and Computer Architecture

OS Service	Hardware Support
Protection	Kernel/User mode
	Protected Instructions
	Base and Limit Registers
Interrupts	Interrupt Vectors
System calls	Trap instructions and trap vectors
I/O	Interrupts or Memory-Mapping
Scheduling, error recovery, billing	Timer
Synchronization	Atomic instructions
Virtual memory	Translation look-aside buffers

Today: OS Structures & Services

- Introduce the organization and components in an OS.
- **OS Components**
  - Processes
  - Synchronization
  - Memory & Secondary Storage Management
  - File Systems
  - I/O Systems
  - Distributed Systems
- **Three example OS organizations**
  - Monolithic kernel
  - Layered architecture
  - Microkernel

From the Architecture to the OS to the User

From the Architecture to the OS to the User: Architectural resources, OS management, and User Abstractions.

Hardware Abstraction	Example OS Services	User Abstraction
Processor	Process management, Scheduling, Traps Protection, Billing, Synchronization	Process
Memory	Management, Protection, Virtual memory	Address space
I/O devices	Concurrency with CPU, Interrupt handling	Terminal, Mouse, Printer, System Calls
File system	Management, Persistence	Files
Distributed systems	Network security, Distributed file system	RPC system calls, file sharing

Processes

- The OS manages a variety of activities:
  - User programs
  - Batch jobs and command scripts
  - System programs: printers, spoolers, name servers, file servers, network listeners, etc.
- Each of these activities is encapsulated in a **process**.
- A process includes the execution context (PC, registers, VM, resources, etc.) and all the other information the activity needs to run.
- *A process is not a program.* A process is one instance of a program in execution. Many processes can be running the same program. Processes are independent entities.

OS and Processes

- The OS creates, deletes, suspends, and resumes processes.
- The OS schedules and manages processes.
- The OS manages inter-process communication and **synchronization**.
- The OS allocates resources to processes.

Synchronization Example:

Banking transactions

- Cooperating processes on a single account: ATM machine transaction, balance computation, Monthly interest computation and addition.
- All of the processes are trying to access the same account simultaneously. What can happen?

Memory & Secondary Storage Management

Main memory

- is the direct access storage for the CPU.
- Processes must be stored in main memory to execute.
- The OS must
  - allocate memory space for processes,
  - deallocate memory space,
  - maintain the mappings from virtual to physical memory (page tables),
  - decide how much memory to allocate to each process, and when a process should be removed from memory (policies).

File System

Secondary storage devices (disks) are too crude to use directly for long term storage.

- The file system provides logical objects and operations on these objects (files).
- A file is the long-term storage entity: a named collection of persistent information that can be read or written.
- File systems support directories which contain the names of files and other directories along with additional information about the files and directories (e.g., when they were created and last modified).

File System Management

- The File System provides *file management*, a standard interface to
  - create and delete files and directories
  - manipulate (read, write, extend, rename, copy, protect) files and directories
  - map files onto secondary storage
- The File System also provides general services such as backups, maintaining mapping information, accounting, and quotas.

Secondary Storage (disk)

- Secondary Storage = persistent memory (endures system failures)
- Low-level OS routines: responsible for low-level disk functions, such as scheduling of disk operations, head movement, and error handling.
  - These routines may also be responsible for managing the disk space (for example, keeping track of the free space).
  - The line between managing the disk space and the file system is very fuzzy, these routines are sometimes in the file system.
- **Example:** A program executable is stored in a file on disk. To execute a program, the OS must load the program from disk into memory.

I/O Systems

The I/O system supports communication with external devices: terminal, keyboard, printer, mouse, ...

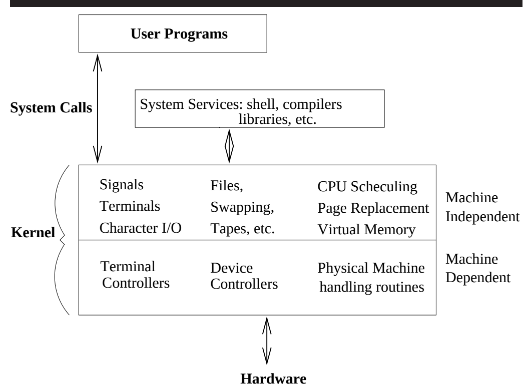
The I/O system

- Supports buffering and spooling of I/O
- Provides a general device driver interface, hiding the differences among devices, often mimicking the file system interface
- Provides device driver implementations specific to individual devices.

Distributed Systems

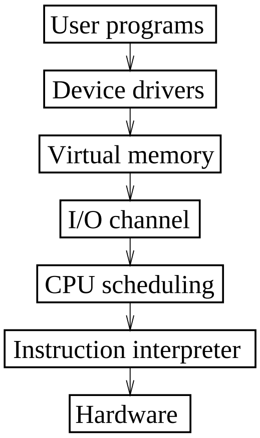
- A distributed system is a collection of processors that do not share memory or a clock.
  - To use non-local resources in a distributed system, processes must communicate over a network,
  - The OS must provide additional mechanisms for dealing with failures and deadlock that are not encountered in a centralized system.
- The OS can support a distributed file system on a distributed system.
  - Users, servers, and storage devices are all dispersed among the various sites.
  - The OS must carry out its file services across the network and manage multiple, independent storage devices.

One Basic OS Structure



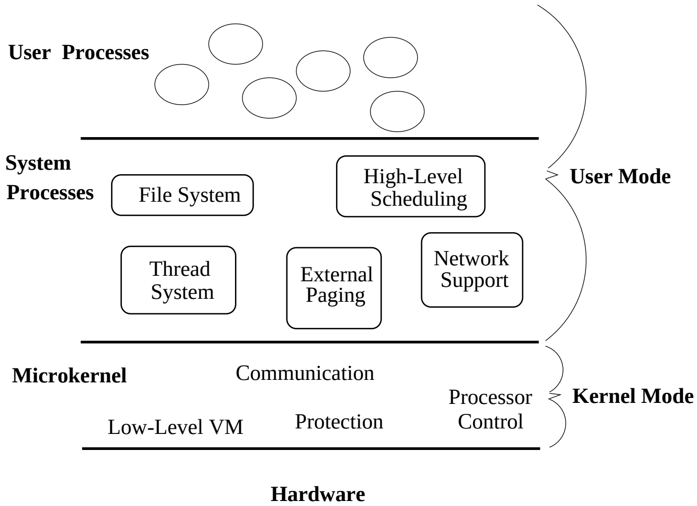
- The *kernel* is the protected part of the OS that runs in kernel mode, protecting the critical OS data structures and device registers from user programs.
- Debate about what functionality goes into the kernel (above figure: UNIX)

Layered OS design



- Advantages: modularity, simplicity, portability, ease of design/debugging
- Disadvantage - communication overhead between layers, extra copying, book-keeping

Microkernel



Microkernel Features

- **Goal:** to minimize what goes in the kernel (mechanism, no policy), implementing as much of the OS in User-Level processes as possible.
- Advantages
  - better reliability, easier extension and customization
  - mediocre performance (unfortunately)
- First Microkernel was Hydra (CMU '70). Current systems include Chorus (France) and Mach (CMU).

Summary

Big Design Issue: How do we make the OS efficient, reliable, and extensible?

General OS Philosophy: The design and implementation of an OS involves a constant tradeoff between *simplicity* and *performance*. As a general rule, strive for simplicity except when you have a strong reason to believe that you need to make a particular component complicated to achieve acceptable performance (strong reason = simulation or evaluation study).