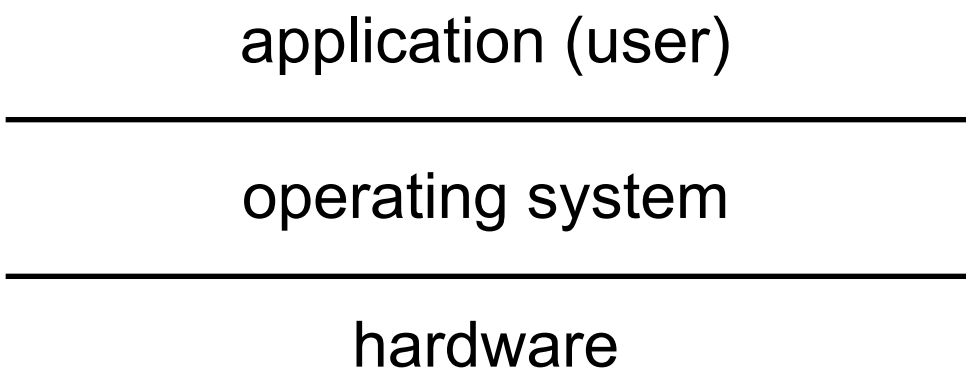
OS and Architecture Overview

What is an operating system?



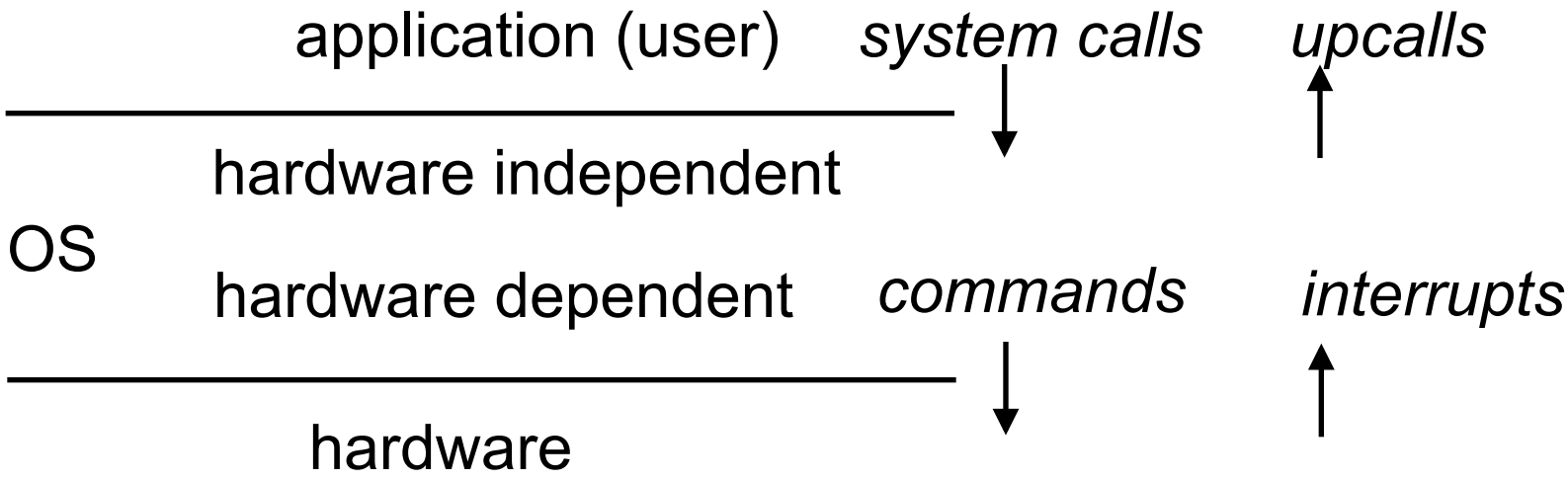
- **A software layer**
  - between hardware and application programs/users,
  - provides a *virtual machine* interface
    - easy to use (hides complexity)
    - safe (prevents and handles errors)
- **Acts as *resource manager***
  - allows programs/users to share hardware resources
  - in a protected way: fair and efficient

Operating System Definition

- A program that acts as an intermediary between a user of a computer and the computer hardware

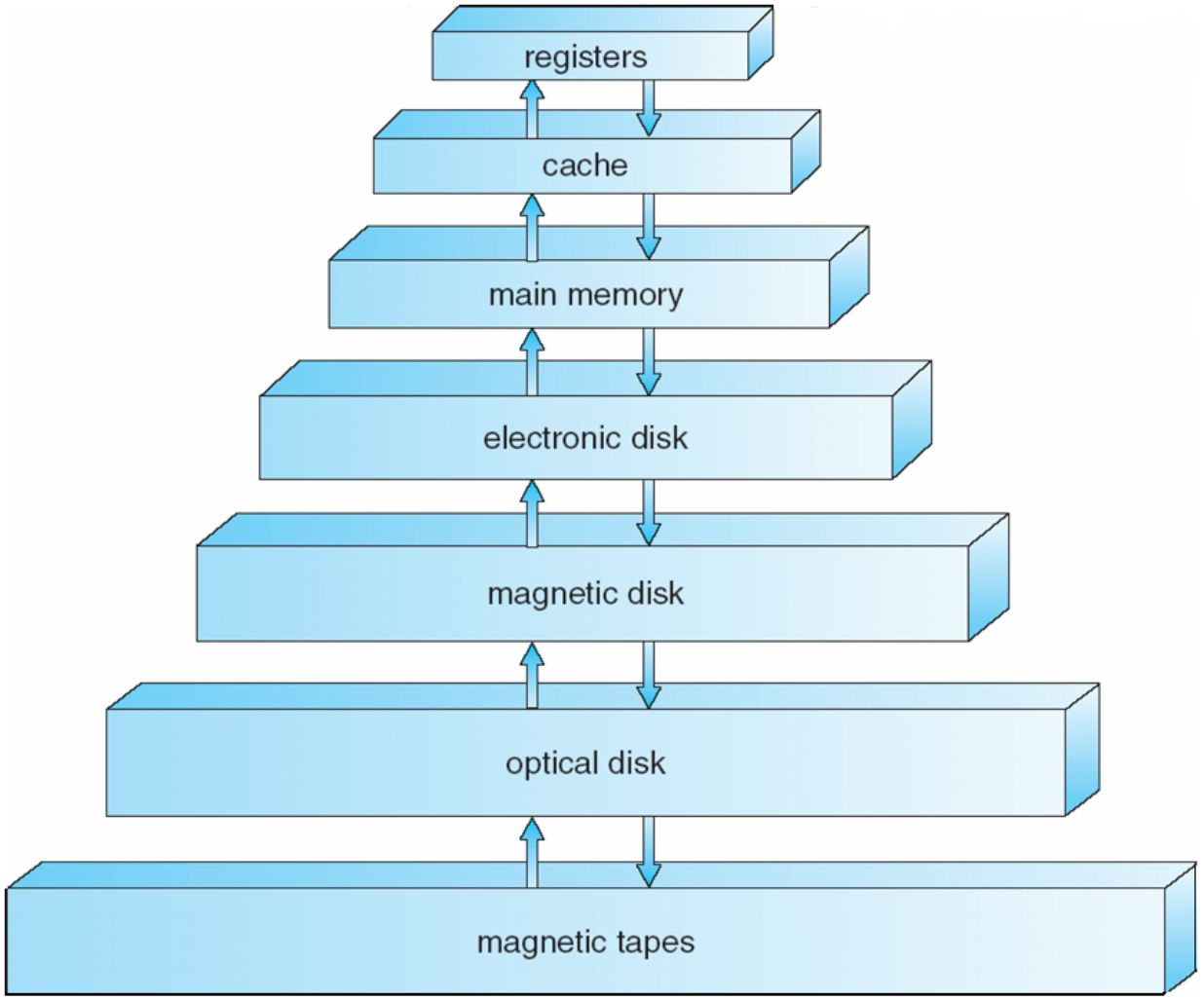
- Operating System Goals
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

How does an OS work?



- Receives requests from the application: system calls
- Satisfies the requests: may issue commands to hardware
- Handles hardware interrupts: may upcall the application
- OS complexity: synchronous calls + asynchronous events

Storage Device Hierarchy



Performance of various levels of storage

Level	1	2	3	4
Name	Registers	Cache	Main memory	Disk storage
Typical size	< 1KB	> 16MB	> 16 GB	> 100GB
Access time (ns)	0.25-0.5	0.5-25	80-250	5000
Bandwidth (MB/sec)	20,000-100,000	5000-10,000	1000-5000	20-150
Managed by	compiler	hardware	Operating system	Operating system

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
- Cache management important design problem
  - Cache size and replacement policy

Caching

- Why does cache work?
  - Temporal Locality: a program is likely to access data it has just recently accessed
  - Spatial Locality: a program is likely to access data that are close to what have just been accessed
- Requires a *cache management* policy
- Caching introduces another level in storage hierarchy.
  - This requires data that are simultaneously stored in more than one level to be *consistent*

Questions!

- How does the application use the OS services? Is it a synchronous or asynchronous?
- How does hardware interact with OS? Is it synchronous or asynchronous?

Hardware Protection

- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual-Mode Operation

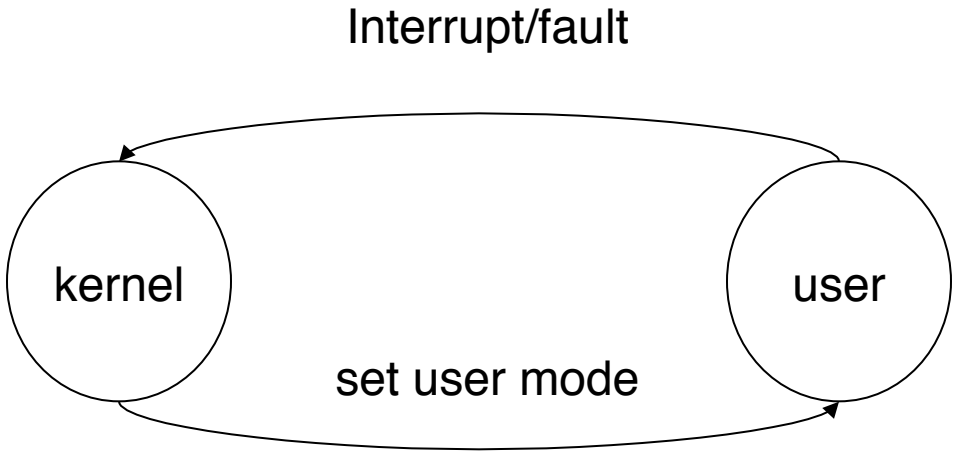
- OS requires hardware support to differentiate between at least two modes of operations
  1. *User mode* – execution done on behalf of a user
  2. *Kernel/monitor mode* – execution done on behalf of operating system

Dual-Mode Operation (Cont.)

- *Mode bit* added to computer hardware to indicate the current mode:

  kernel (0) or user (1)

- When an interrupt or fault occurs hardware switches to kernel mode

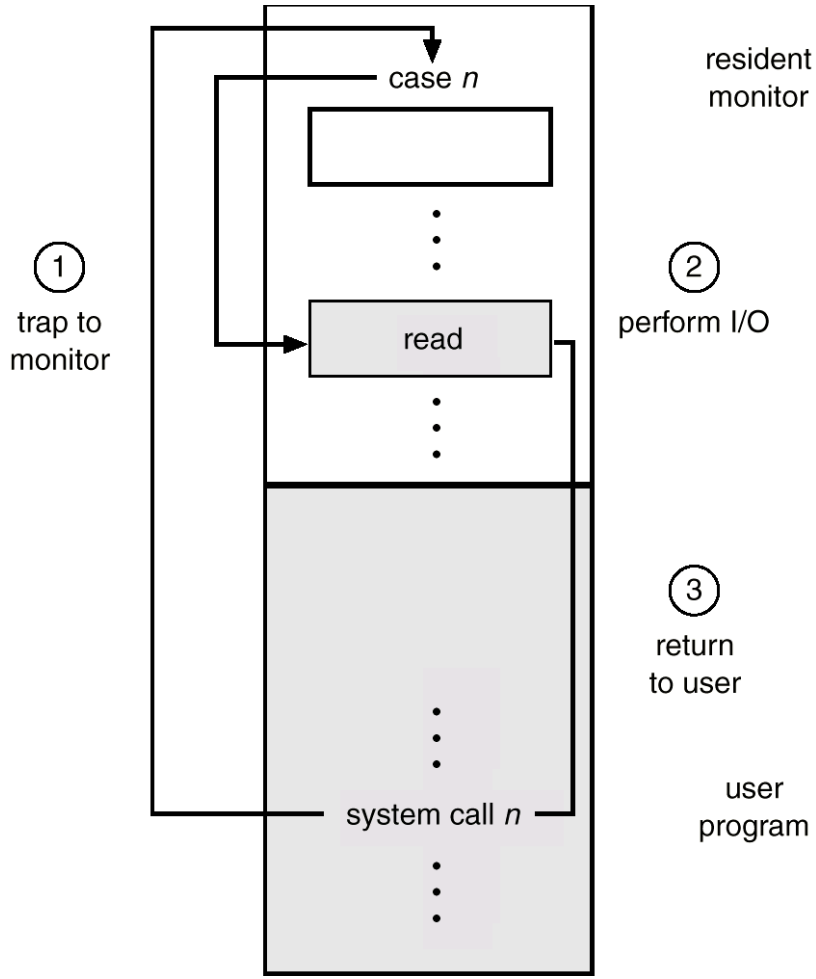


Privileged instructions can be issued only in kernel mode

I/O Protection

- All I/O instructions are privileged instructions
- Must ensure that a user program could never gain control of the computer in kernel mode
  - For example, a user program that, as part of its execution, stores a new address in the interrupt vector)

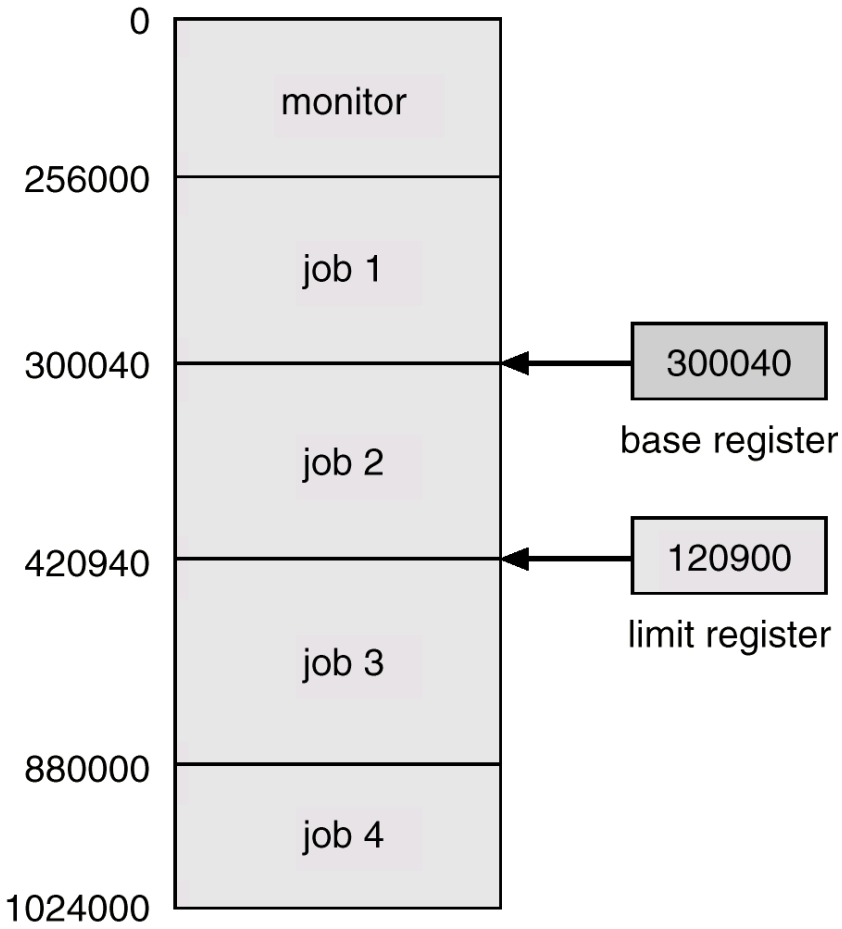
Use of A System Call to Perform I/O



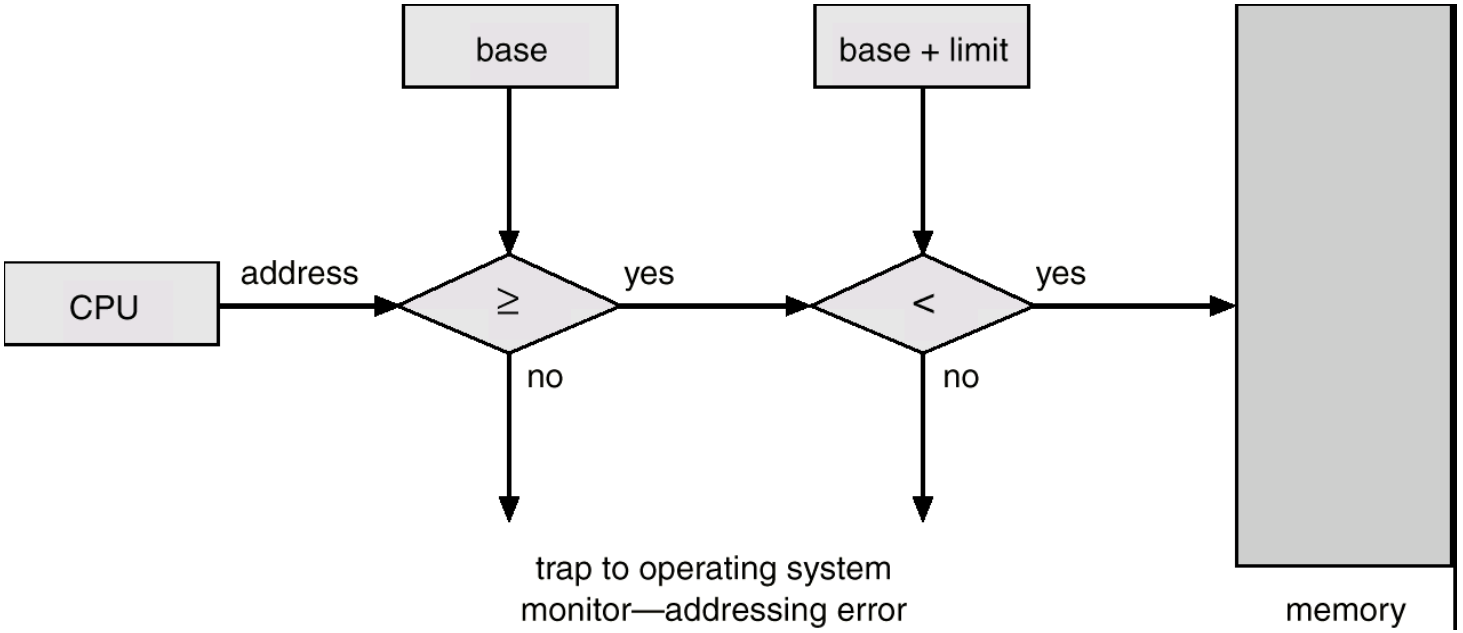
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines
- In order to have memory protection, at a minimum 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

Use of A Base and Limit Register



Hardware Address Protection



Hardware Protection

- When executing in the kernel mode, the operating system has unrestricted access to both kernel and user's memory
- The load instructions for the *base* and *limit* registers are privileged instructions

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control
  - Timer is decremented every clock tick
  - When timer reaches the value 0, an interrupt occurs
- Timer commonly used to implement time sharing
- Timer also used to compute the current time
- Load-timer is a privileged instruction

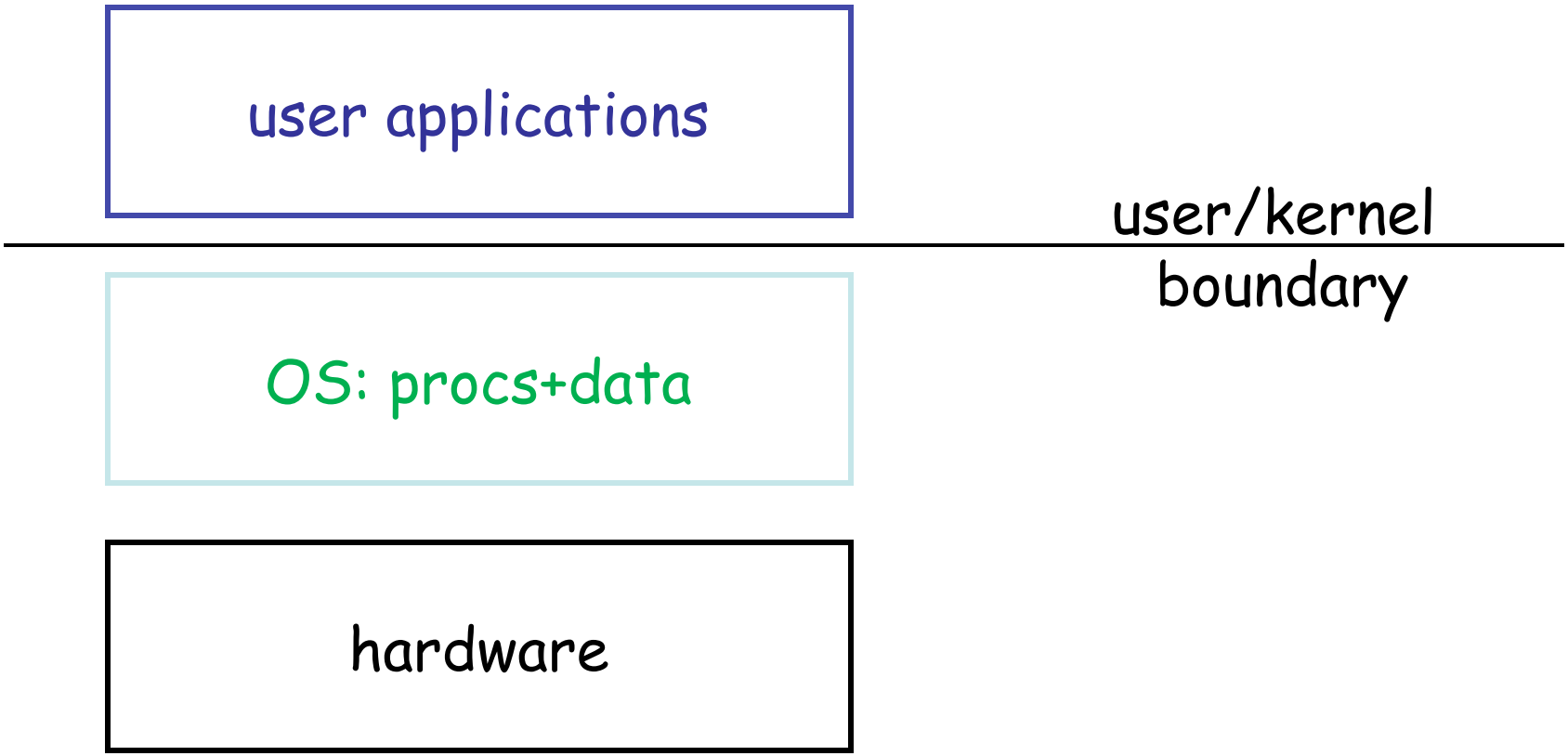
Question

- Can CPU directly access the following devices?
  - (A) Disk;
  - (B) Memory;
  - (C) Register;
  - (D) Network;
  - (E) CD-ROM;

OS Structure

- Monolithic
- Layered
- Microkernel

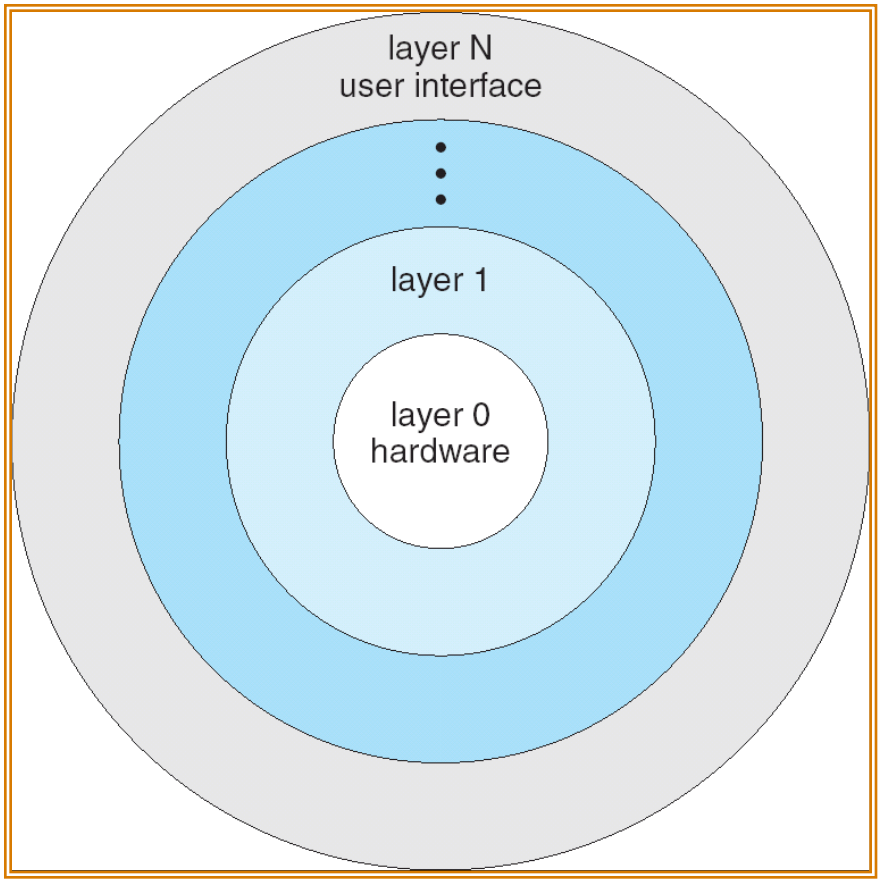
Monolithic


user applications
user/kernel
boundary
OS: procs+data
hardware

Layered Approach

- 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.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

Layered Operating System

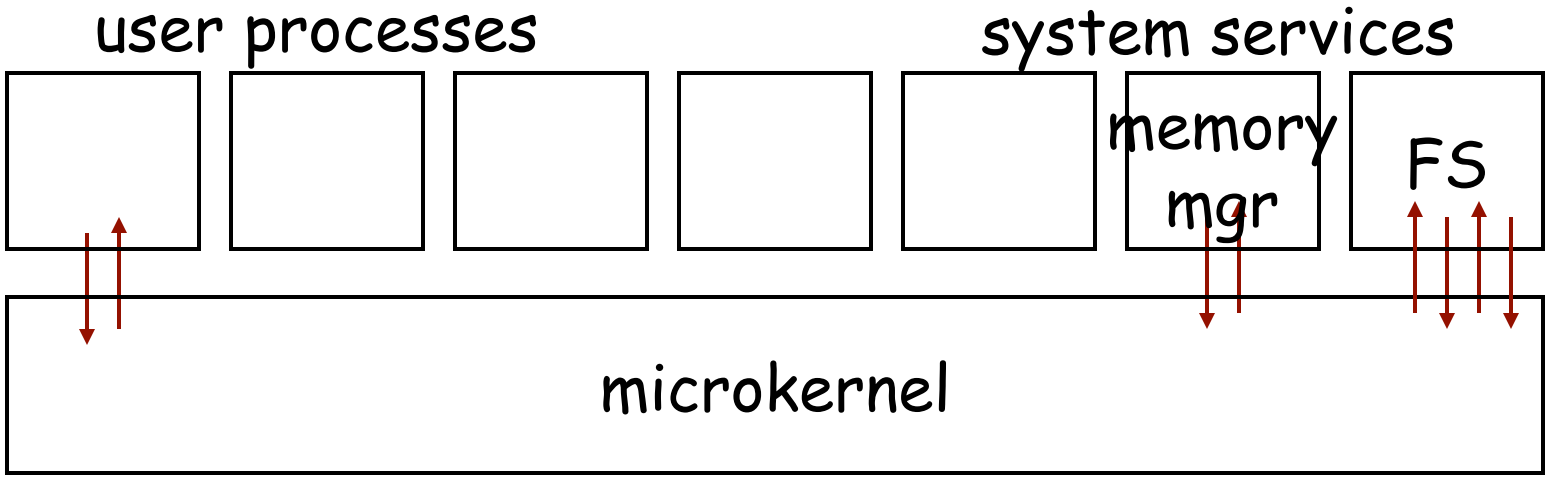


Microkernel System Structure

- Moves as much from the kernel into "*user*" space
- Communication takes place between user modules using message passing
- Advantages:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Disadvantages:
  - Performance overhead of user space to kernel space communication

Microkernel in action

- System services at the same level as user process
- System call crosses user/kernel boundary many times



Comparison

	Performance	Extensibility	Reliability
Monolithic	✓	✗	✗
Layered	✗	—	—
Microkernel	—	✓	✓


good

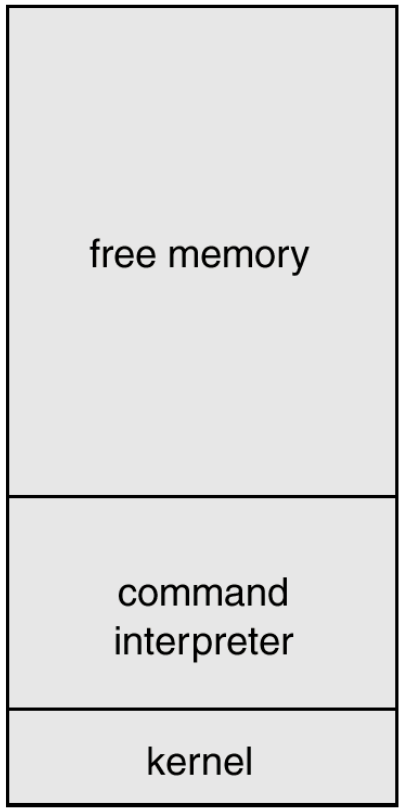
In between


bad

MS-DOS System Structure

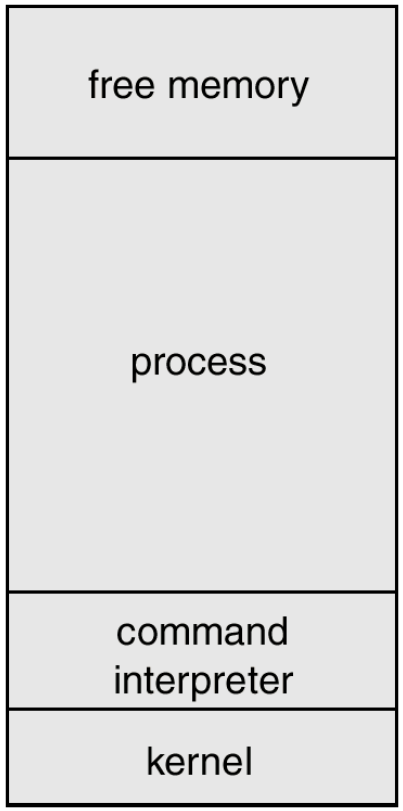
- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

MS-DOS Execution (Single Process OS)



(a)

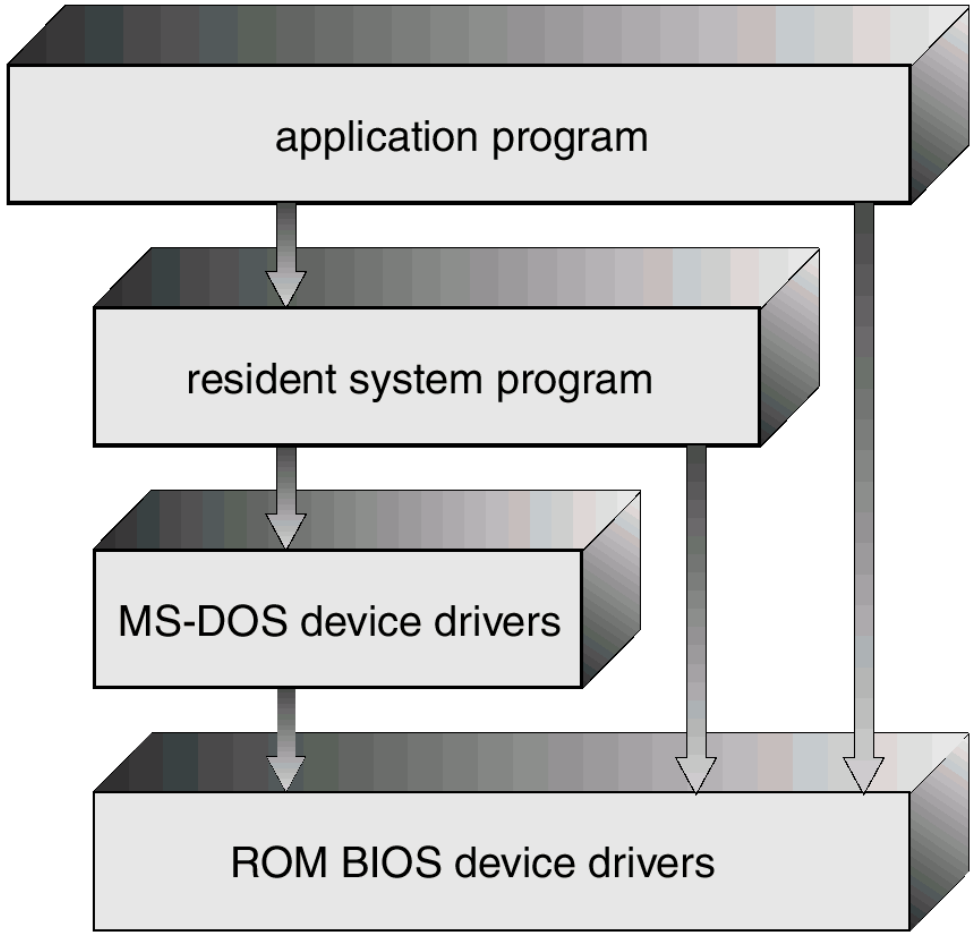
At System Start-up



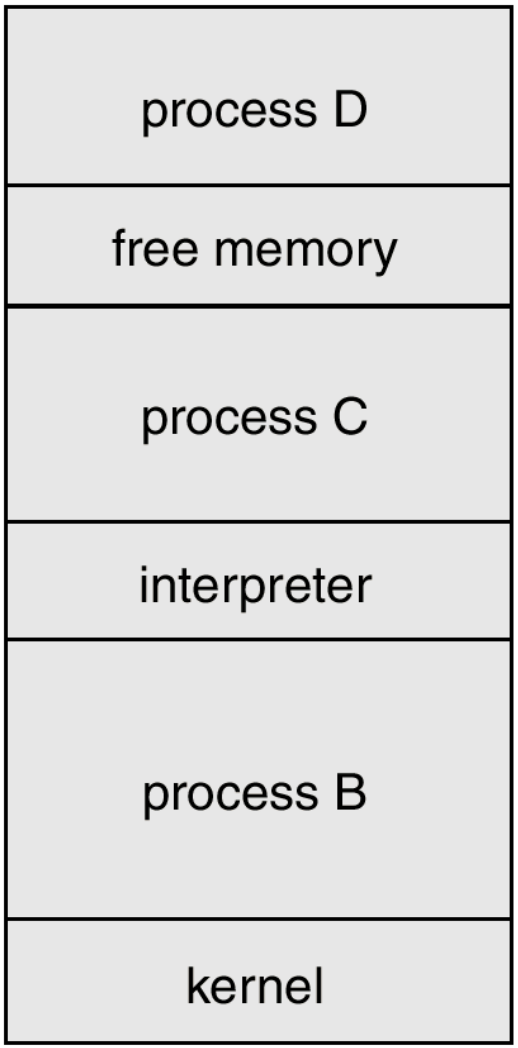
(b)

Running a Program

MS-DOS Layer Structure

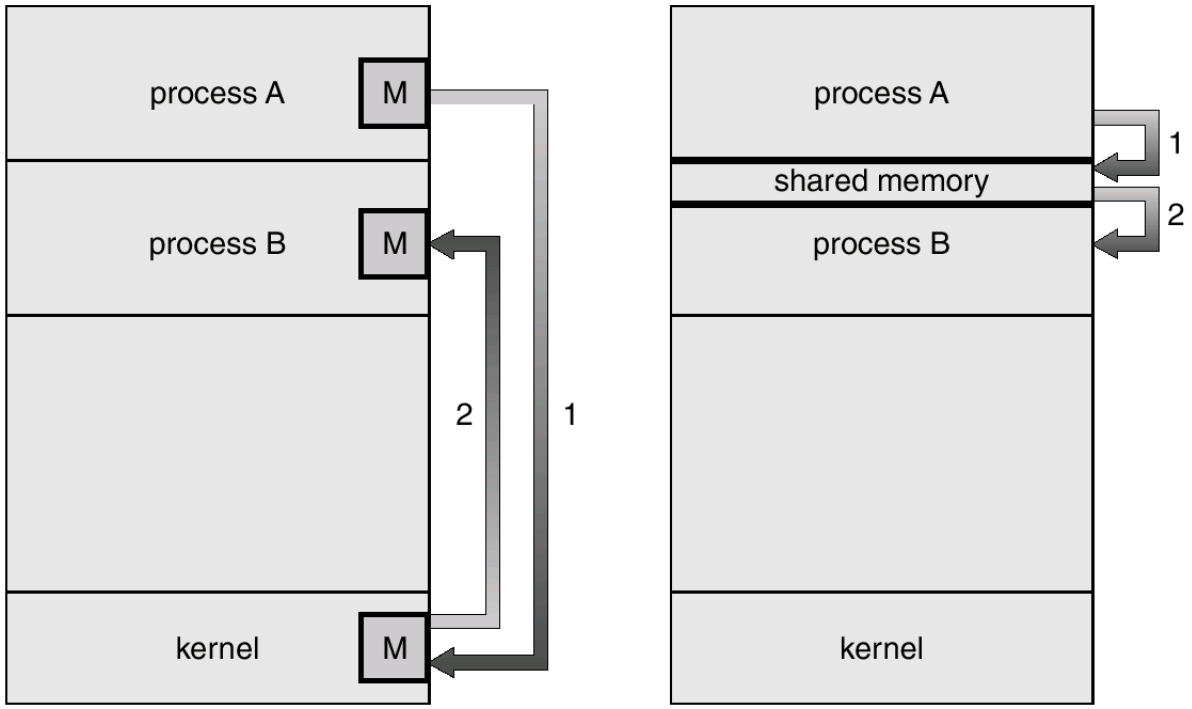


UNIX Running Multiple Programs



Communication Models

Communication may take place using either message passing or shared memory



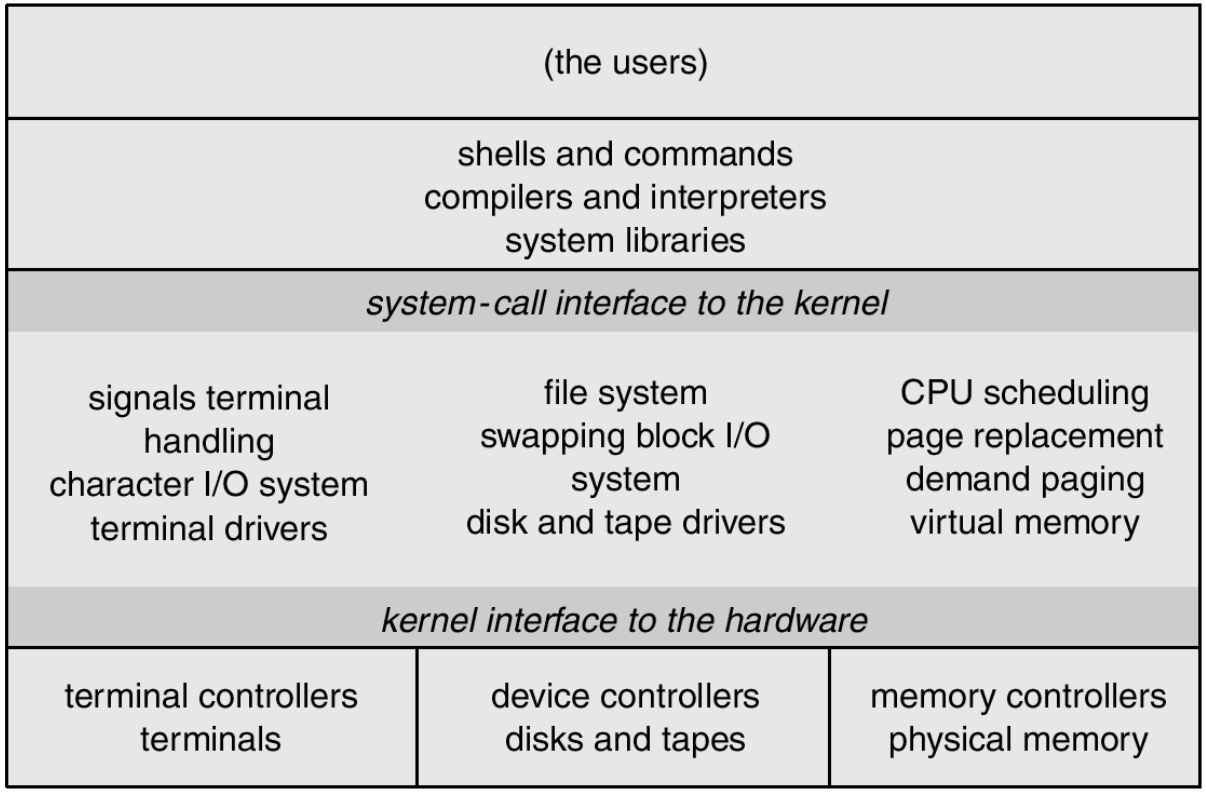
Message Passing

Shared Memory

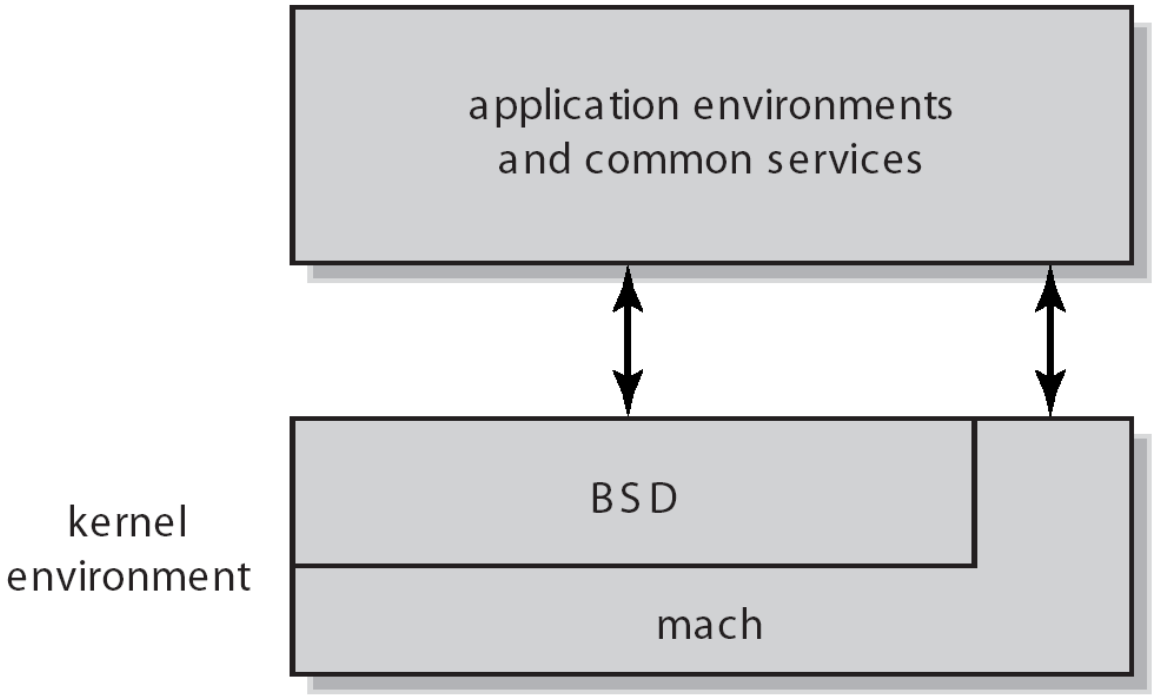
UNIX system structure

- UNIX
  - Limited hardware functionality -> the original UNIX operating system had limited structuring
  - UNIX OS consists of two separable parts
- Systems programs
- The kernel
  - Everything below the system-call interface and above the physical hardware
  - Provides
    - File system
    - CPU scheduling
    - Memory management
    - Etc.
  - A large number of functions for one level

UNIX System Structure



Mac OS X Structure



- Hybrid architecture
  - Layered structure + Mach microkernel