

I.-C. Lin, Assistant Professor. Textbook: Operating System
Principles 7ed

CHAPTER 2: SYSTEM STRUCTURES



Chapter 2: System Structures



- Operating System Services
- User Operating System Interface
- System Calls
- Types of System Calls
- System Programs
- Operating System Design and Implementation
- Operating System Structure
- Virtual Machines
- Operating System Generation
- System Boot

Objectives



- To describe the services an operating system provides to users, processes, and other systems
- To discuss the various ways of structuring an operating system
- To explain how operating systems are installed and customized and how they boot

Operating System Services

- Services for the convenience of the programmers (or users)
 - ▣ User interface
 - Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
 - ▣ Program execution
 - Load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
 - ▣ I/O operations
 - A running program may require I/O, which may involve a file or an I/O device.

CLI vs. GUI

The image illustrates the difference between a Command Line Interface (CLI) and a Graphical User Interface (GUI). It shows a Windows desktop environment with several windows open. The most prominent is a command prompt window titled 'C:\WINDOWS\system32\cmd.exe' which displays the following directory listing for C:\:

```
磁碟區序號: 3015-F344

C:\ 的目錄

2006/08/17 下午 06:32
2006/08/17 下午 06:32
2006/08/30 下午 11:39
2006/08/17 下午 07:20
2006/08/22 下午 05:14
2006/08/17 下午 06:48
2006/08/26 上午 12:07
2006/06/15 上午 08:23
2006/05/10 上午 02:53
2006/08/17 下午 06:25
2006/08/17 下午 06:25
2006/06/08 上午 05:10
2006/06/21 下午 08:01
2006/06/08 上午 05:10
2006/09/01 上午 02:58
2004/09/02 下午 06:01
2006/08/22 下午 05:24
          9 個檔案
          8 個目錄
```

The desktop also shows a Windows Media Player window, a file explorer window titled 'D:\Course\Teaching' showing a folder structure, and a taskbar at the bottom with various application icons. The system tray shows the date and time as '上午 12:49'.

Operating System Services (cont.)

- File-system manipulation

- Read and write, create and delete, search files and directories, list file information, permission management.

- Communications

- Processes may exchange information, on the same computer or over a network
- May be via shared memory or through message passing

- Error detection

- Error may occur in the CPU and memory hardware, in I/O devices, in user program
- Should take the appropriate action to ensure correct and consistent computing
- Debugging facilities for users to efficiently use the system

Operating System Services (Cont.)

- For ensuring efficient system operations
 - ▣ **Resource allocation**
 - allocating resources to multiple users or multiple jobs running at the same time
 - Many types of resources.
 - ▣ **Accounting**
 - To keep track of which users use how much and what kinds of computer resources
 - Statistics for improving computing services
 - ▣ **Protection and security**
 - **Protection** involves ensuring that all access to system resources is controlled
 - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

User Operating System Interface - CLI

Command-line interpreter allows direct command entry

- ▣ Sometimes implemented in kernel, sometimes by systems program

- ▣ Sometimes multiple flavors implemented – **shells**

- ▣ Primarily fetches a command from user and executes it
 - Sometimes commands built-in, sometimes just names of programs
 - If the latter, adding new features doesn't require shell modification

User Operating System Interface - GUI

- User-friendly **desktop** metaphor interface
 - ▣ Usually mouse, keyboard, and monitor
 - ▣ **Icons** represent files, programs, actions, etc
 - ▣ Various mouse buttons over objects in the interface cause various actions

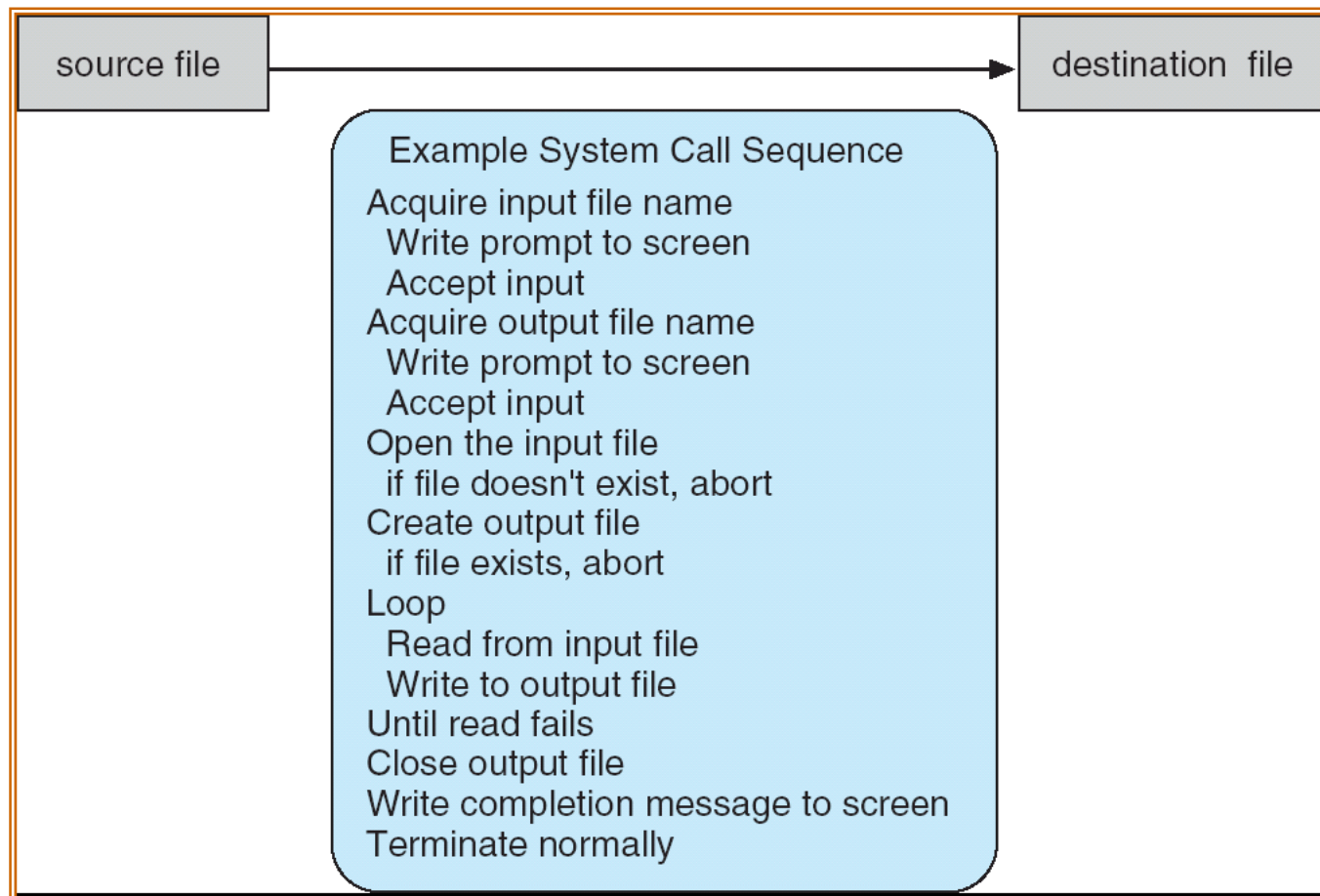
- Many systems now include both CLI and GUI interfaces
 - ▣ Microsoft Windows is GUI with CLI “command” shell
 - ▣ Apple Mac OS X as “Aqua” GUI interface with UNIX kernel underneath and shells available
 - ▣ Solaris is CLI with optional GUI interfaces

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
 - Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why do we use APIs rather than system calls?

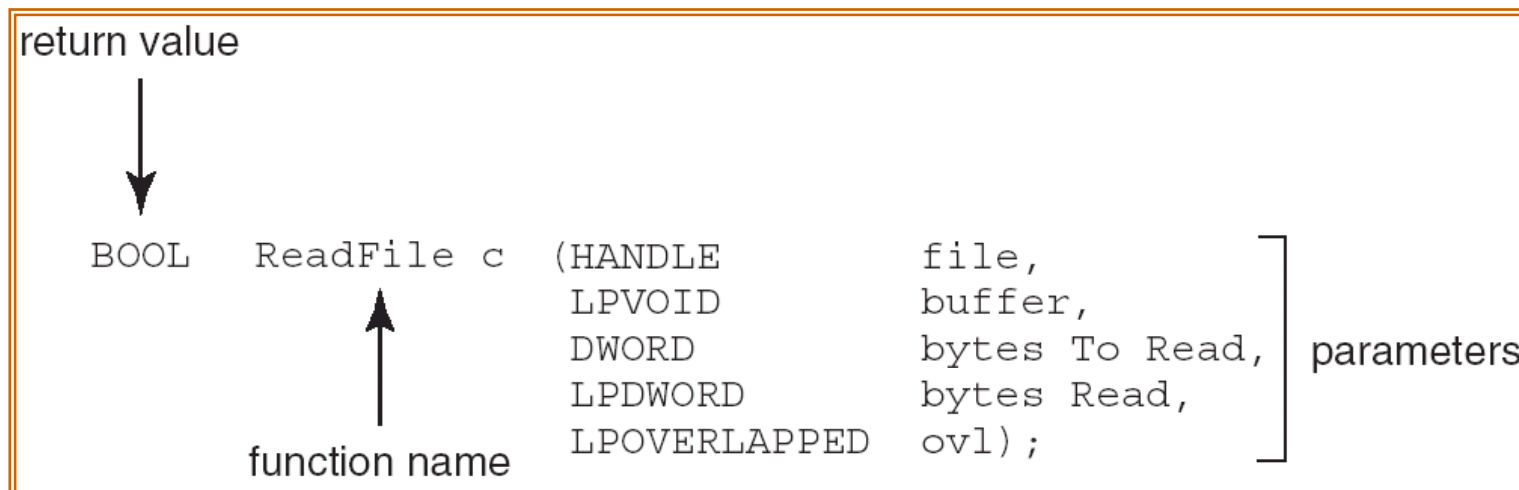
Example of System Calls

- System call sequence to copy the contents of one file to another file



Example of Standard API

- Consider the ReadFile() function in the Win32 API—a function for reading from a file
- A description of the parameters passed to ReadFile()
 - HANDLE file—the file to be read
 - LPVOID buffer—a buffer where the data will be read into and written from
 - DWORD bytesToRead—the number of bytes to be read into the buffer
 - LPDWORD bytesRead—the number of bytes read during the last read
 - LPOVERLAPPED ovl—indicates if overlapped I/O is being used



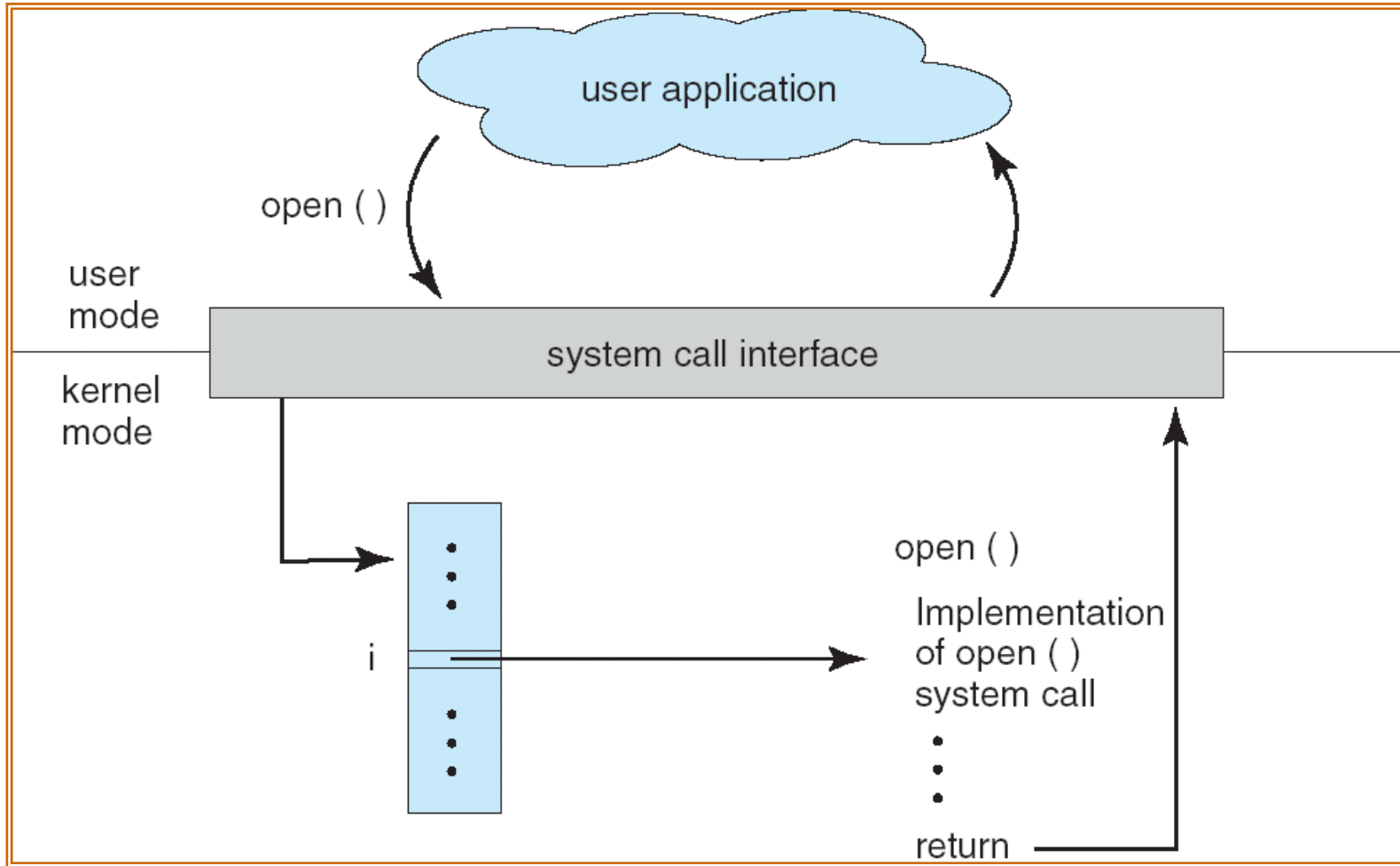
System Call Implementation



- The system call interface
 - ▣ invokes intended system call in OS kernel and returns status of the system call and any return values

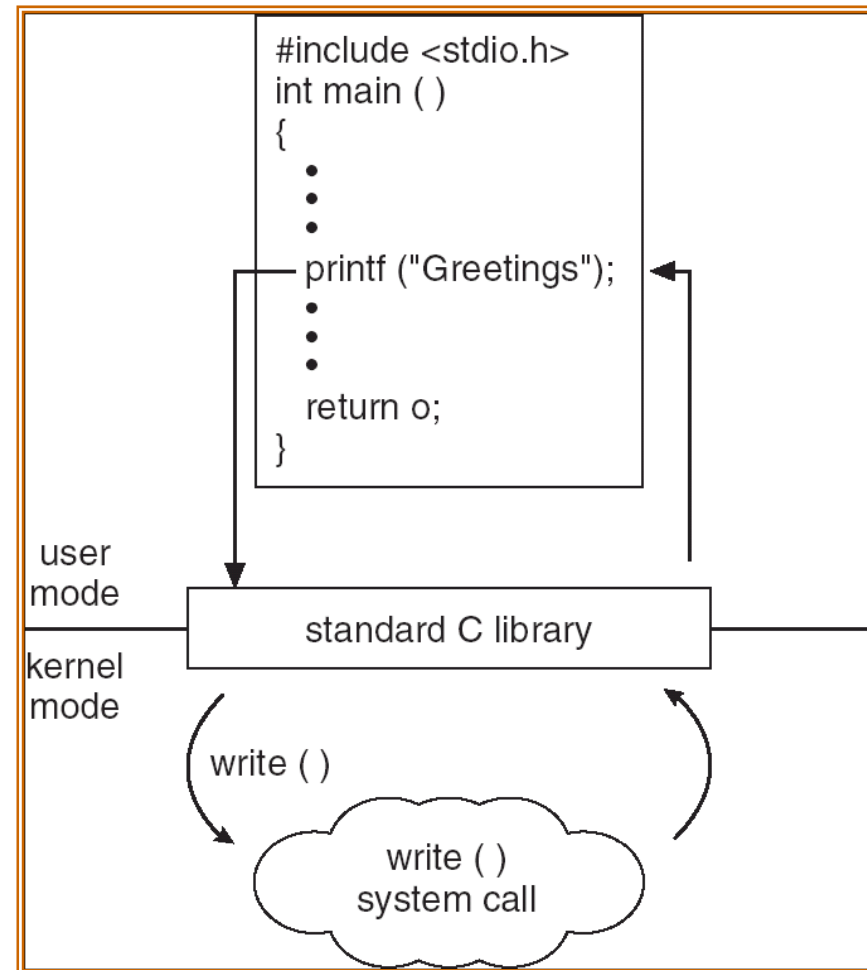
- The caller need know nothing about how the system call is implemented
 - ▣ Just needs to obey API and understand what OS will do as a result call
 - ▣ Most details of OS interface hidden from programmer by API
 - Managed by run-time support library (set of functions built into libraries included with compiler)

API – System Call – OS Relationship



Standard C Library Example

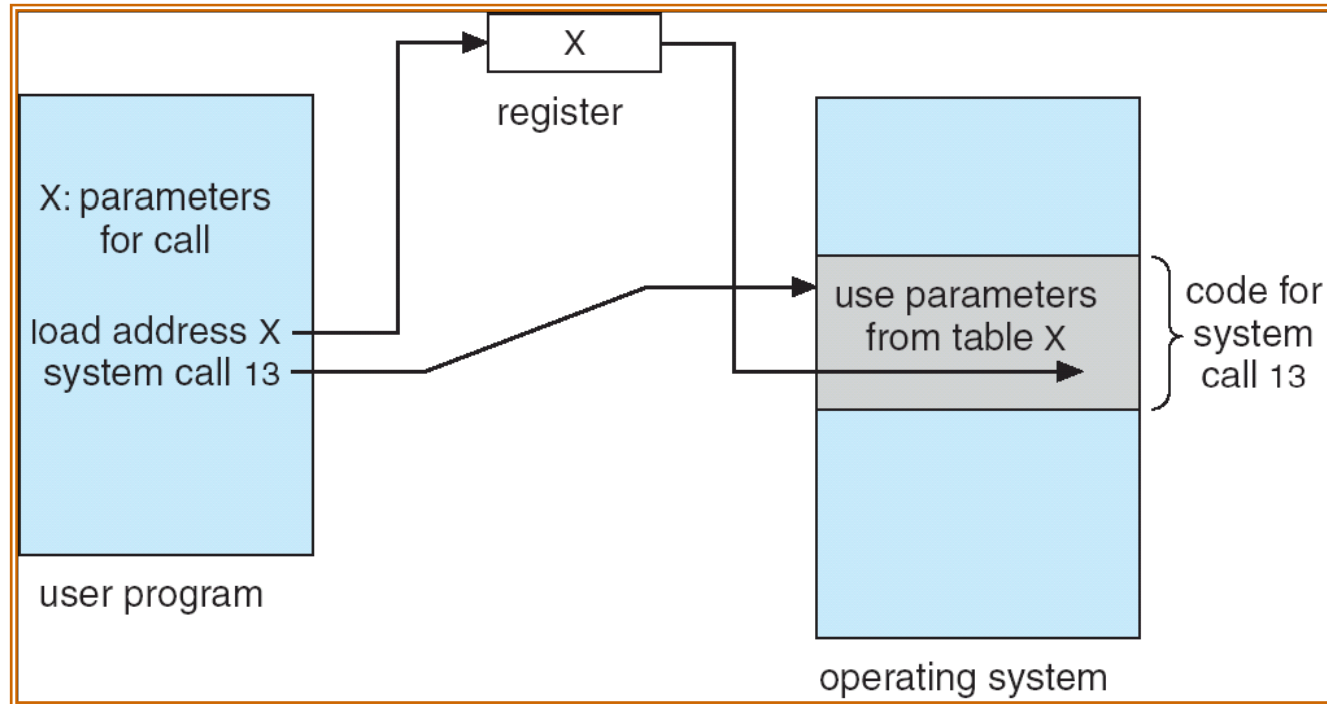
- C program invoking printf() library call, which calls write() system call



System Call Parameter Passing

- Three general methods used to pass parameters to the OS
 - ▣ Simplest: pass the parameters in *registers*
 - In some cases, may be more parameters than registers
 - ▣ Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
 - This approach taken by Linux and Solaris
 - ▣ Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system
 - ▣ Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table

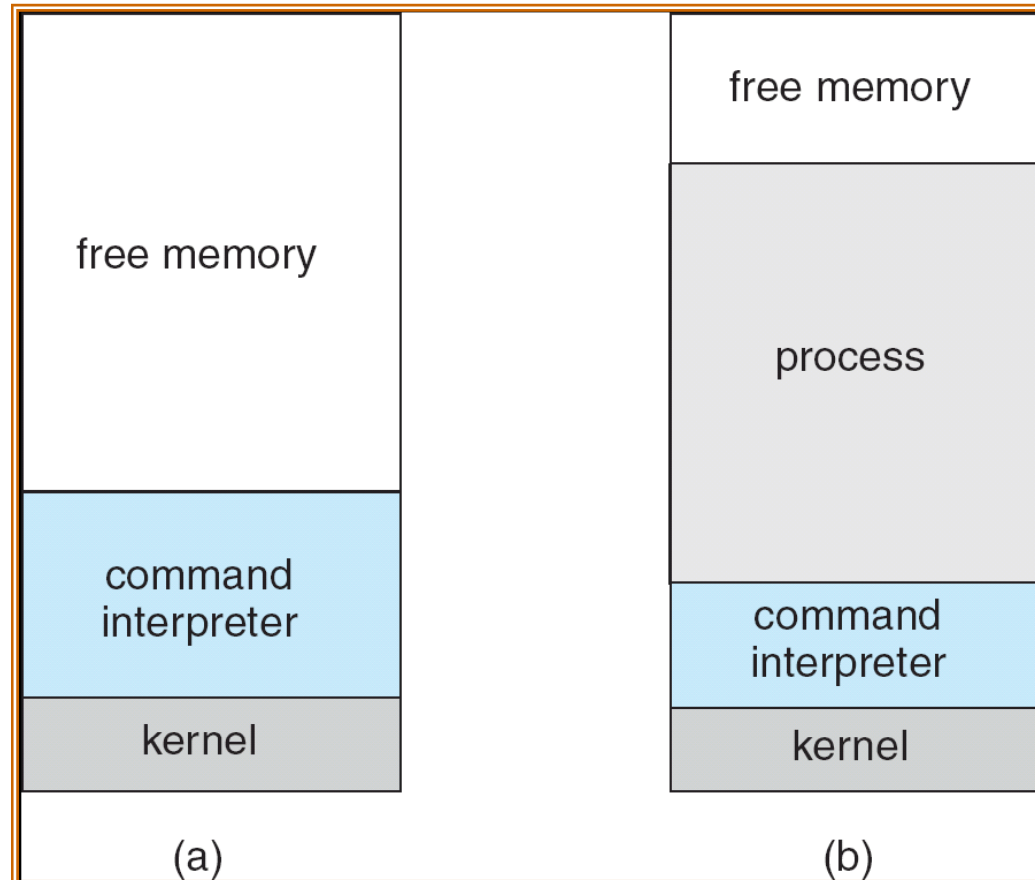


Types of System Calls



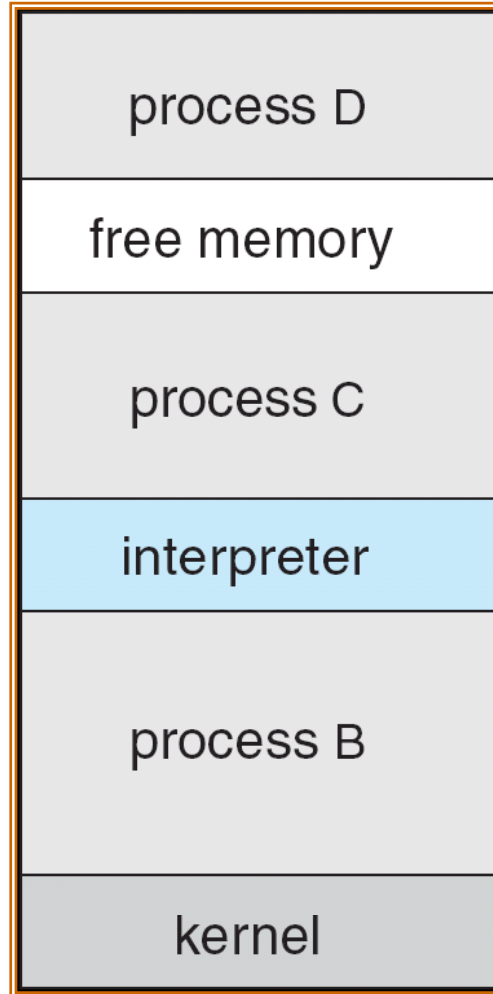
- Process control
- File management
- Device management
- Information maintenance
- Communications

MS-DOS execution



(a) At system startup (b) running a program

FreeBSD Running Multiple Programs



System Programs

- A convenient environment for program development and execution. They can be divided into:
 - File manipulation
 - Status information
 - File modification
 - Programming language support
 - Program loading and execution
 - Communications
 - Application programs

- Most users' view of the operation system is defined by system programs, not the actual system calls

System Programs



- Provide a convenient environment for program development and execution
 - ▣ Some of them are simply user interfaces to system calls; others are considerably more complex
- File management
 - ▣ Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

System Programs (cont.)

- Status information

- Some ask the system for info - date, time, amount of available memory, disk space, number of users
- Others provide detailed performance, logging, and debugging information
- Typically, these programs format and print the output to the terminal or other output devices
- Some systems implement a registry - used to store and retrieve configuration information

System Programs (cont.)



- File modification
 - ▣ Text editors to create and modify files
 - ▣ Special commands to search contents of files or perform transformations of the text

- Programming-language support
 - ▣ Compilers, assemblers, debuggers and interpreters sometimes provided

- Program loading and execution
 - ▣ Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language


System Programs (cont.)



- Communications

- Provide the mechanism for creating virtual connections among processes, users, and computer systems
- Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

Operating System Design and Implementation



- Design and Implementation of OS not “solvable”, but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start by defining goals and specifications
- Affected by choice of hardware, type of system

Operating System Design and Implementation (Cont.)

- *User goals and System goals*

- ▣ User goals

- operating system should be convenient to use, easy to learn, reliable, safe, and fast

- ▣ System goals

- operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

- Important principles to separate

- ▣ **Policy:** What will be done?

- ▣ **Mechanism:** How to do it?

Operating System Design and Implementation (Cont.)



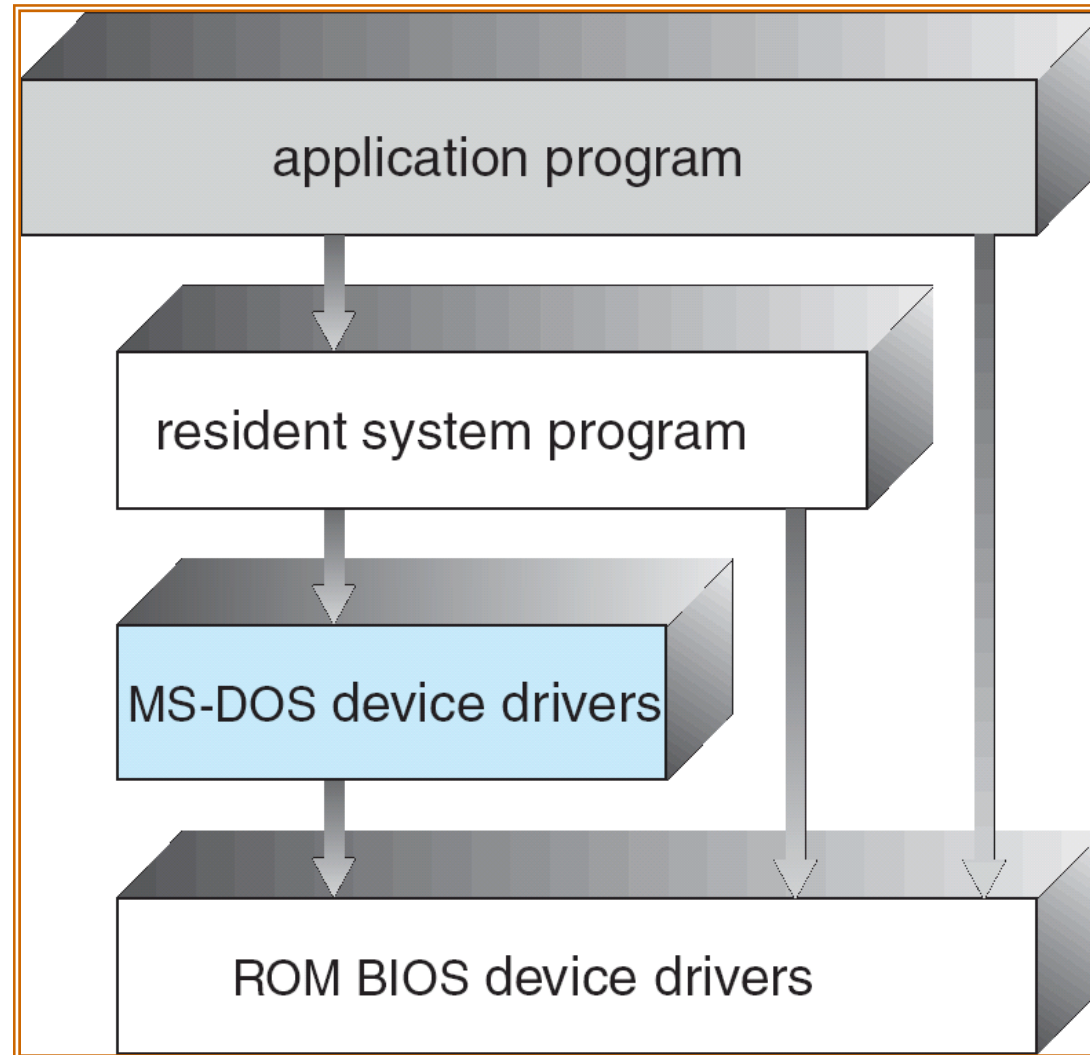
- Mechanisms determine how to do something, policies decide what will be done
 - ▣ The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

Simple 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 Layer Structure

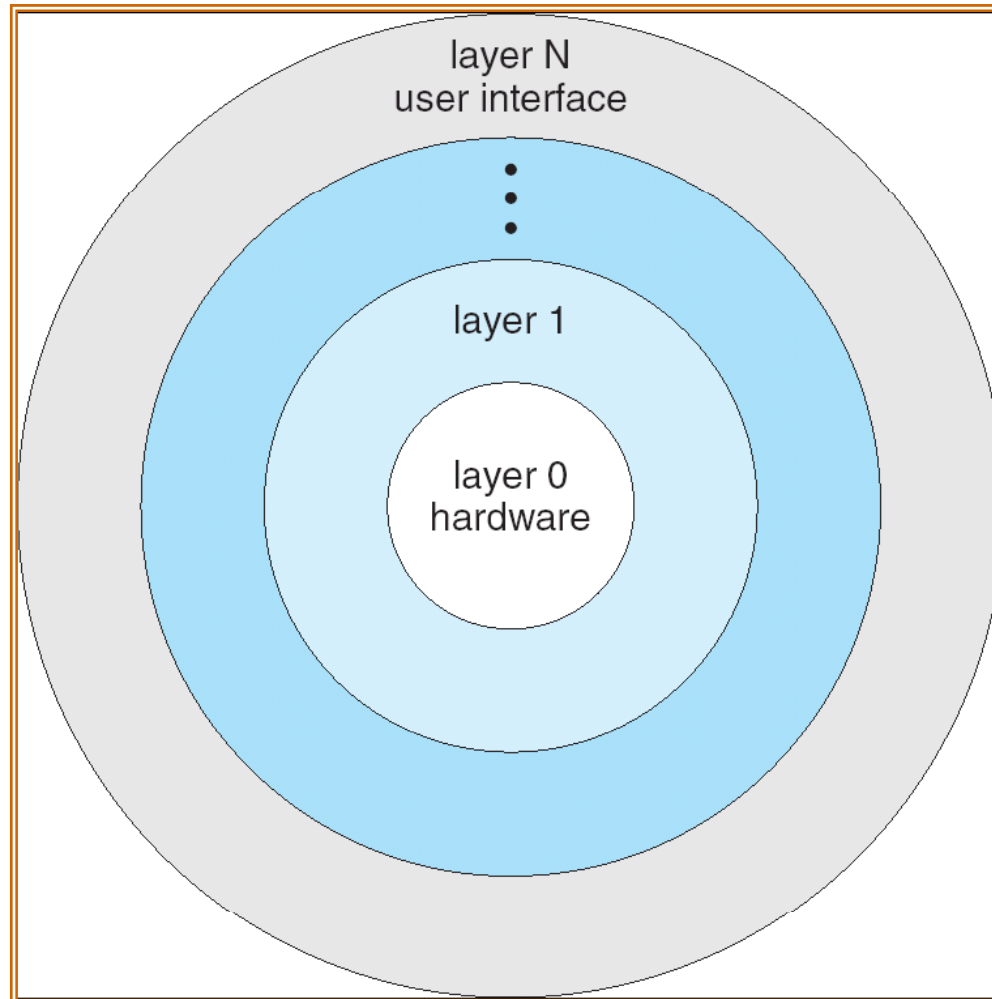


Layered Approach

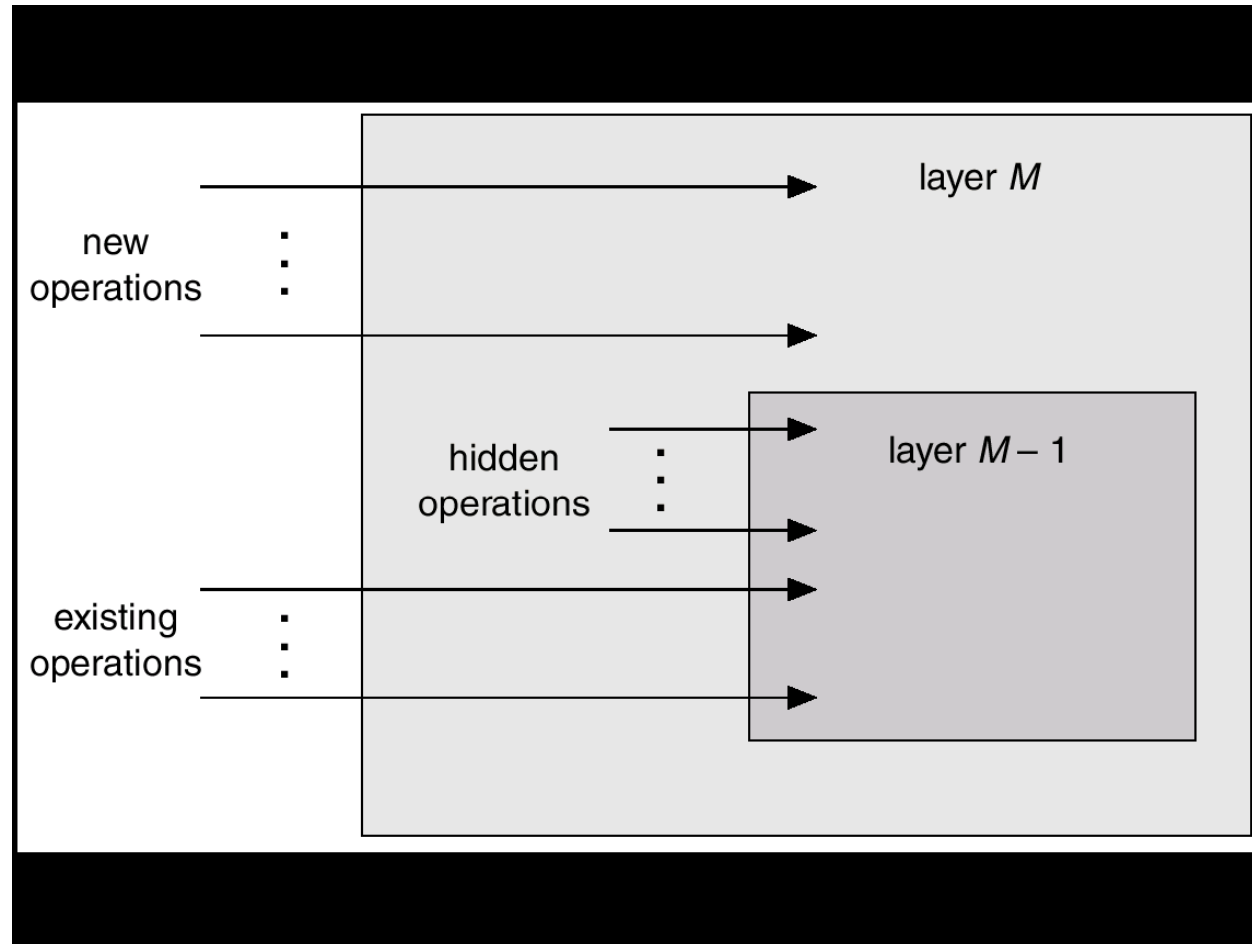


- The operating system is divided into a number of layers (levels)
 - ▣ The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

Layered Operating System



Layered Operating System



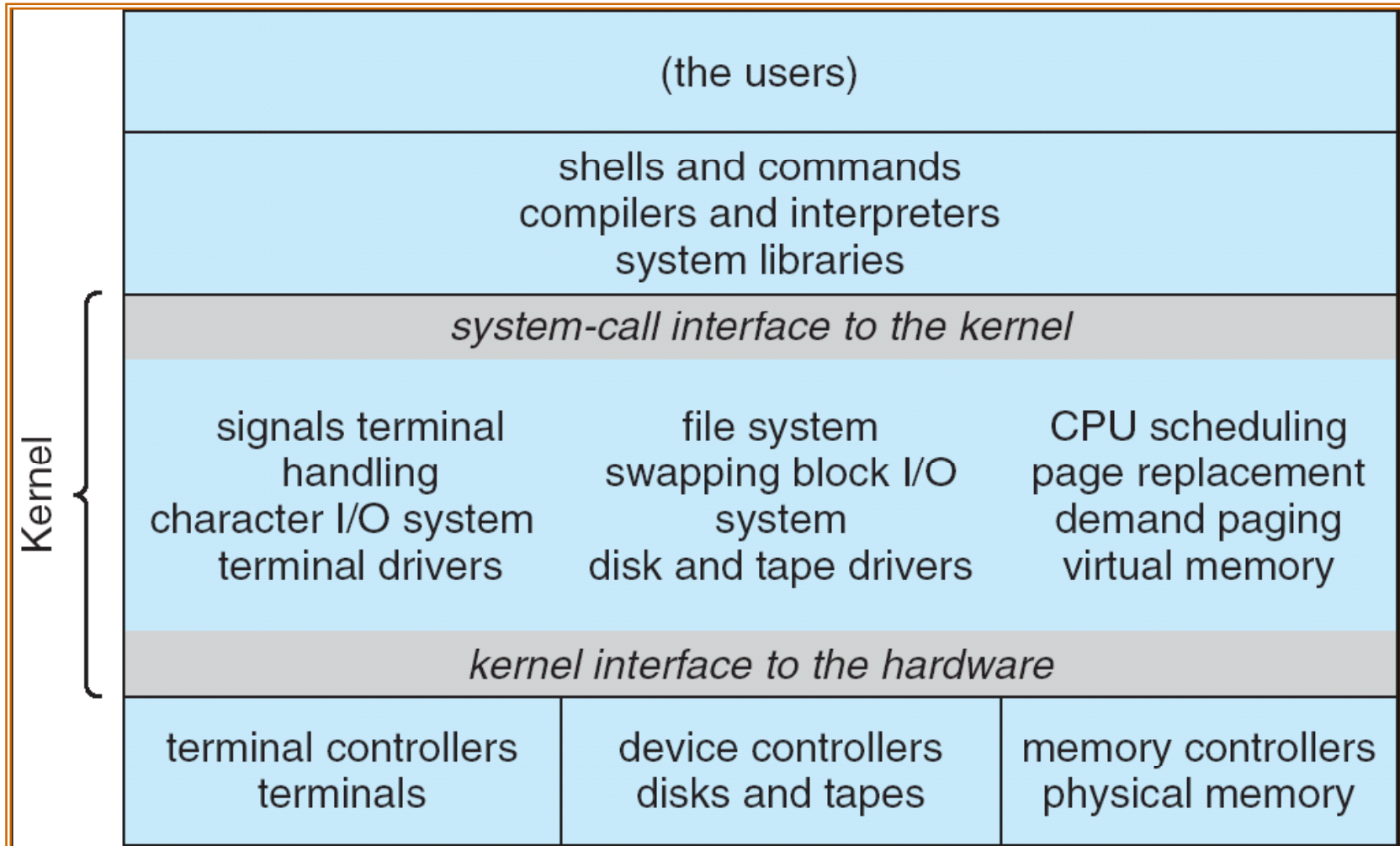
UNIX

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.

- The UNIX OS consists of two separable parts
 - ▣ Systems programs
 - ▣ The kernel
 - Consists of everything below the system-call interface and above the physical hardware

 - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

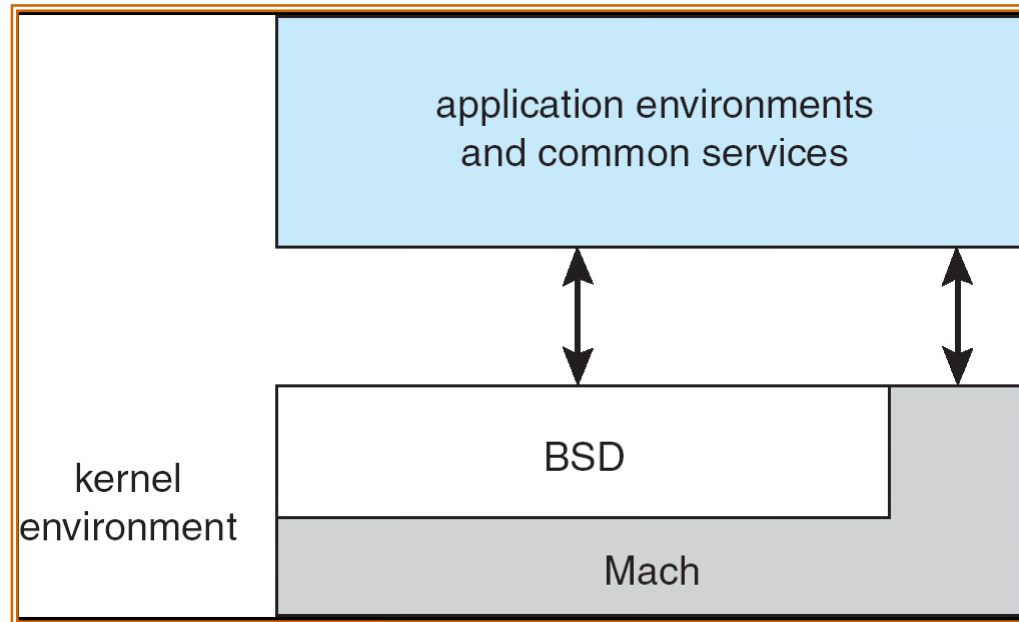
UNIX System Structure



Microkernel System Structure

- Moves as much from the kernel into “*user*” space
- Communication takes place between user modules using message passing
- Benefits:
 - ▣ 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
- Detriments:
 - ▣ Performance overhead of user space to kernel space communication

Mac OS X Structure

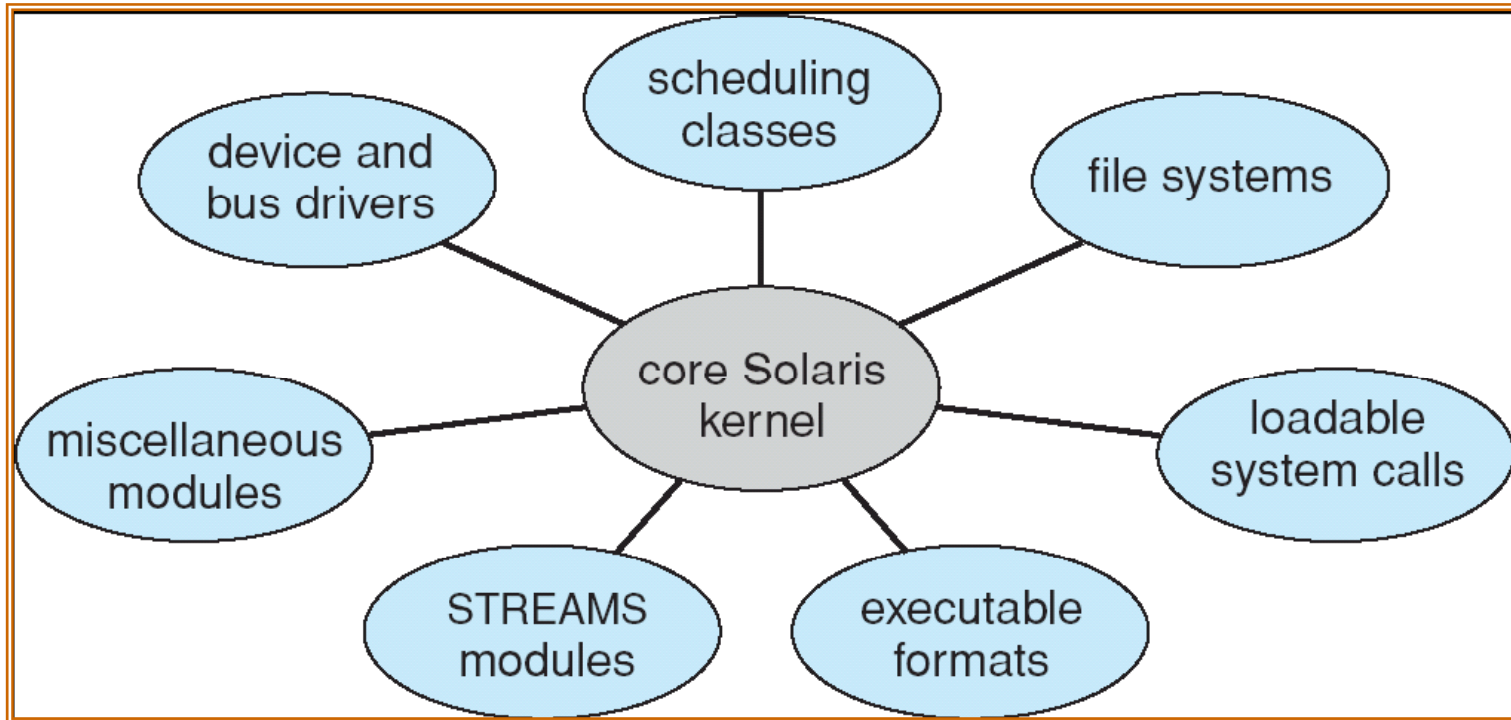


Modules

- Most modern operating systems implement kernel modules
 - ▣ Uses object-oriented approach
 - ▣ Each core component is separate
 - ▣ Each talks to the others over known interfaces
 - ▣ Each is loadable as needed within the kernel

- Overall, similar to layers but with more flexible

Solaris Modular Approach



Virtual Machines

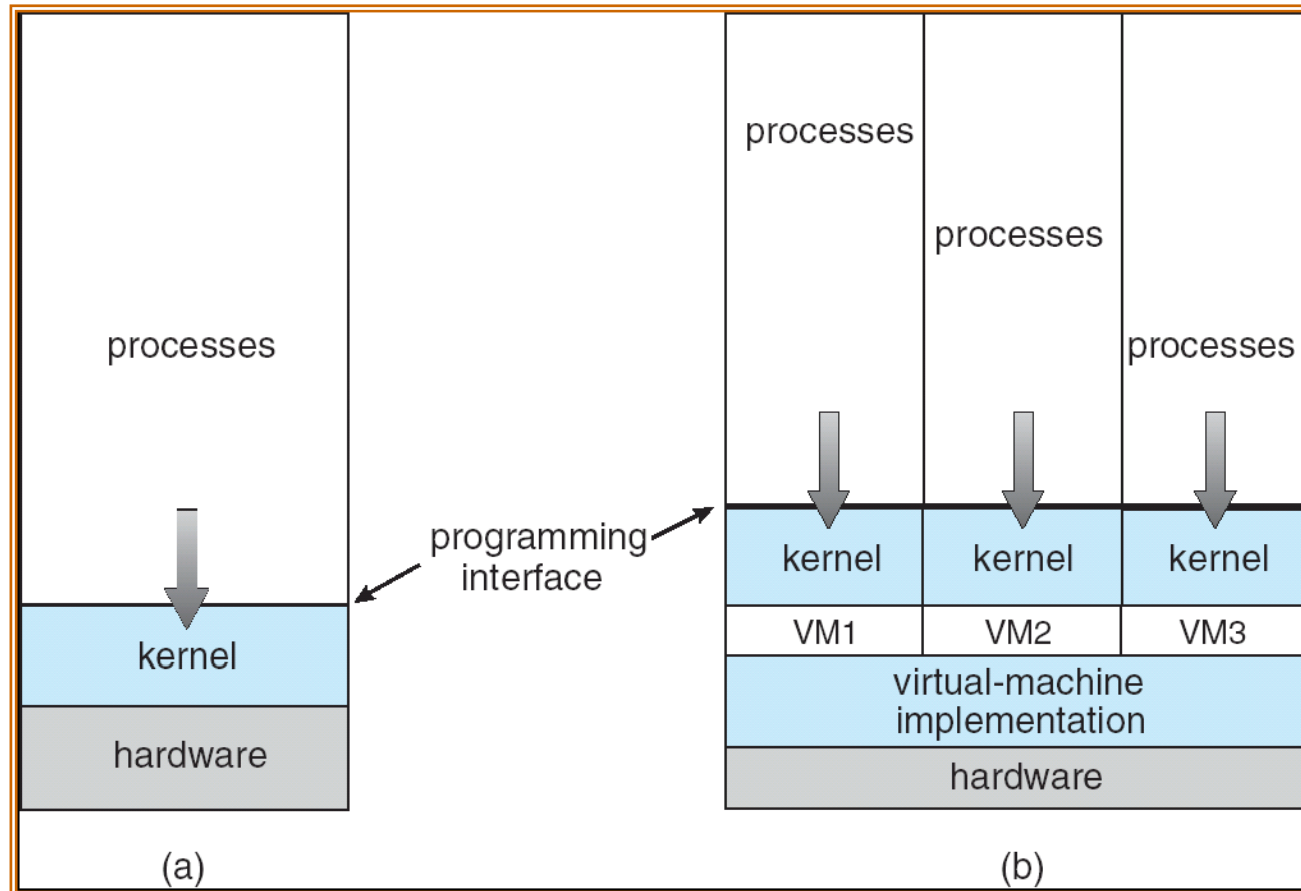


- Takes the layered approach to its logical conclusion.
 - ▣ It treats hardware and the operating system kernel as though they were all hardware
- A virtual machine provides an interface *identical* to the underlying bare hardware
- The operating system creates the illusion of multiple processes.
 - ▣ Each executing on its own processor with its own (virtual) memory

Virtual Machines (Cont.)

- The resources of the physical computer are shared to create the virtual machines
 - ▣ CPU scheduling can create the appearance that users have their own processor
 - ▣ Spooling and a file system can provide virtual card readers and virtual line printers
 - ▣ A normal user time-sharing terminal serves as the virtual machine operator's console
- Difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine
 - ▣ Virtual user mode, virtual monitor mode: in the physical user mode.

Virtual Machines (Cont.)



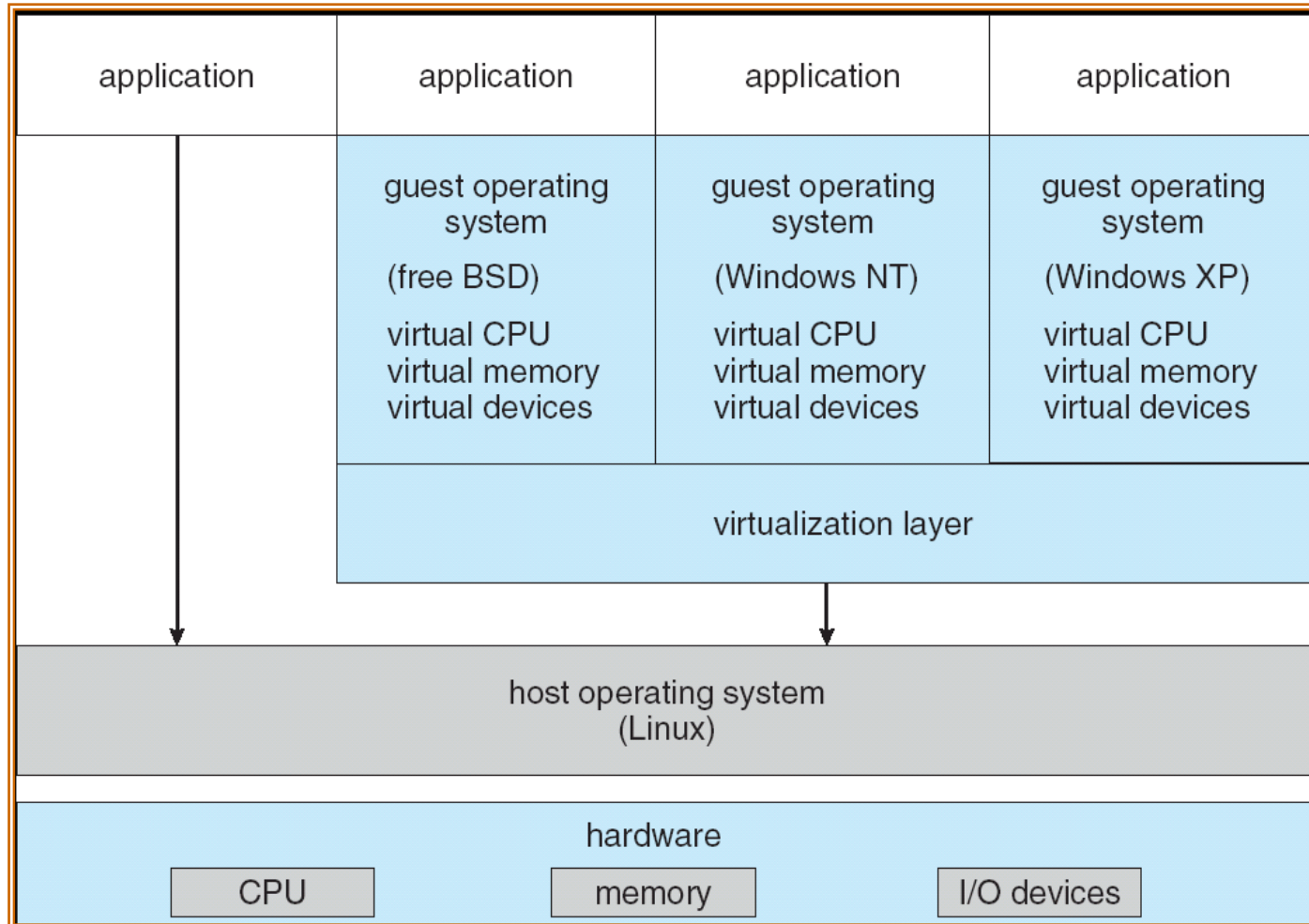
Benefits of Virtual Machines

- Provides complete protection of system resources
 - Each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.

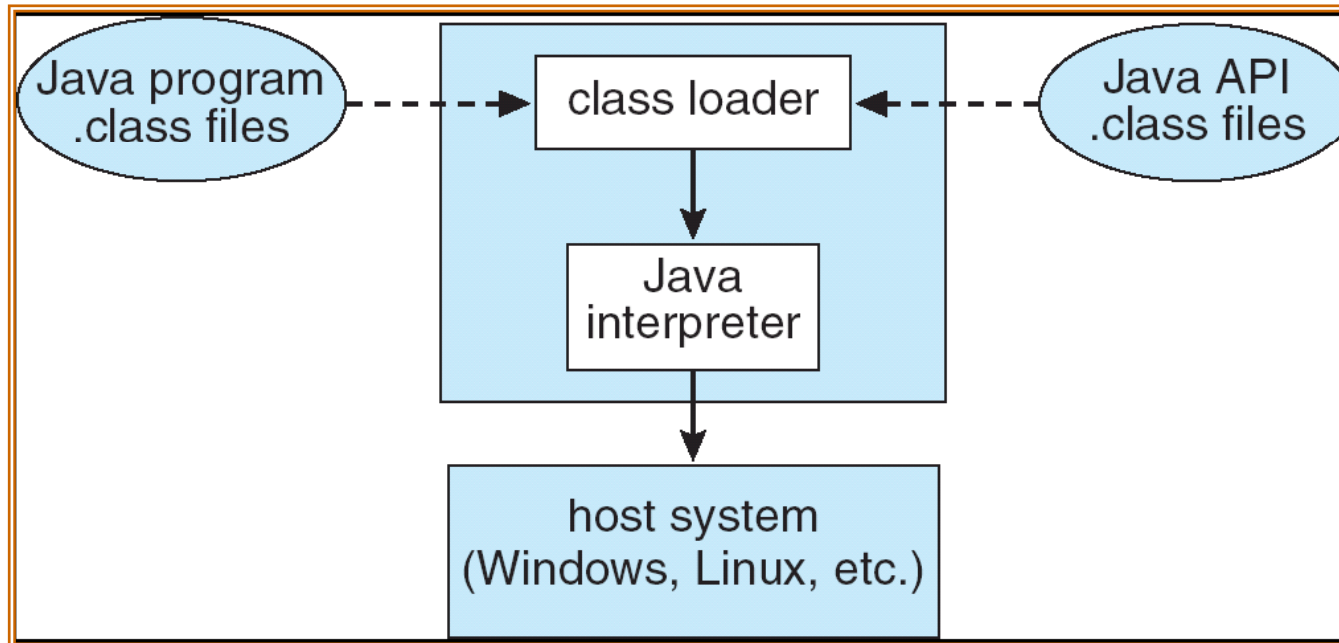
- A perfect vehicle for operating-systems research and development.
 - Instead of on a physical machine; does not disrupt normal system operation.

- A means of solving system compatibility problems

VMware Architecture



The Java Virtual Machine



System Implementation

- Traditionally written in assembly language, operating systems can now be written in higher-level languages. (Unix, WinNT, OS/2)
 - ▣ Unix: parts of the scheduler and device drivers are implemented in assembly language.
- Code written in a high-level language:
 - ▣ can be written faster.
 - ▣ is more compact.
 - ▣ is easier to understand and debug.
- An operating system is far easier to port (move to some other hardware) if it is written in a high-level language.

Operating System Generation



- Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site
- *Booting* – starting a computer by loading the kernel
- *Bootstrap program* – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution

System Boot

- Operating system must be made available to hardware so hardware can start it
 - ▣ Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
 - ▣ Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
 - ▣ When power initialized on system, execution starts at a fixed memory location
 - Firmware used to hold initial boot code
- MBR (Master Boot Record): typically, the 1st sector of the hard disk

END OF CHAPTER 2

