Operating System Overview

Dept. of Computer Science

Hanyang University

Operating System Structure

- Multiprogramming needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so that CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running. Creates interactive computing
  - Response time should be sufficiently small
  - Each user has at least one program executing in memory ⇨process
  - If several jobs ready to run at the same time ⇨ CPU scheduling
  - If processes don't fit in memory, swapping moves them in and out to run
  - Virtual memory allows execution of processes not completely in memory

Operating System Operations

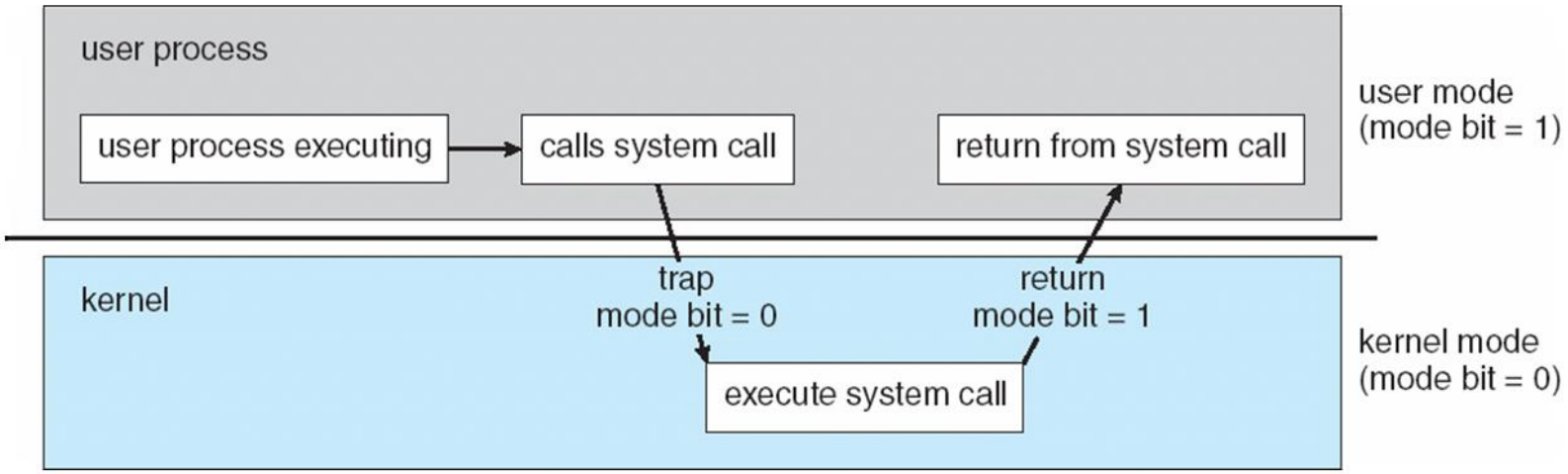
- Interrupt driven by hardware
- Software error or request creates exception or trap
  - Division by zero, request for operating system service (system call)
- Both interrupt and exception intercept normal program execution

- (Optional) Event handling in operating systems
  - Asynchronous event: interrupt
    - We do not know when and what kind of event will occur
    - Interrupt informs us that a SPECIFIC event occurred NOW
  - Synchronous event: exception (trap, fault, abort)
    - Trap: intentional exception (e.g., system call)
    - Fault: recoverable error (e.g., page fault, divide by zero) → (may) continue the program execution after handling the event
    - Abort: unrecoverable error (e.g., memory bit corruption) → stop the program

Operating System Operations

- Dual-mode operation allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

< Transition from User to Kernel Mode >



Process Management

- Process
  - A program in execution
  - It is a unit of work within the system
  - Program is a *passive entity*, process is an *active entity*
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
- Multi-threaded process has one program counter per thread

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and response time to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

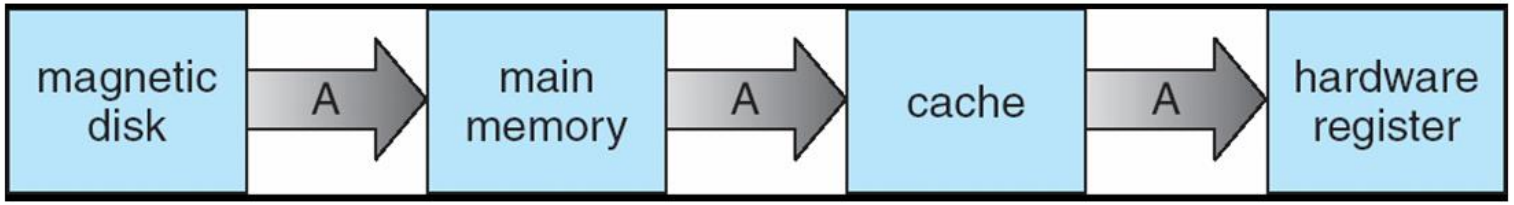
Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store
  - data that does not fit in main memory or
  - data that must be kept for a “long” period of time
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Migration of Integer A from Disk to Register



- Multitasking environments must be careful so as to use most recent value, no matter where it is stored in the storage hierarchy
- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including
    - buffering (storing data temporarily while it is being transferred)
    - caching (storing parts of data in faster storage for performance)
    - spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- Protection
  - Any mechanism for controlling access of processes or users to resources defined by the OS
- Security
  - Defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft
- Systems generally first distinguish among users, to determine who can do what
  - User identities (user IDs, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, file
  - Privilege escalation allows user to change to effective ID with more rights (ex. setuid)

Operating System Services

- A set of operating-system services provides functions that are helpful to the user:
  - User interface
    - Almost all operating systems have a user interface (UI)
    - Varies between Command-Line Interface(CLI), Graphics User Interface (GUI), Batch Interface
  - Program execution
    - The system must be able to 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
  - File-system manipulation
    - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

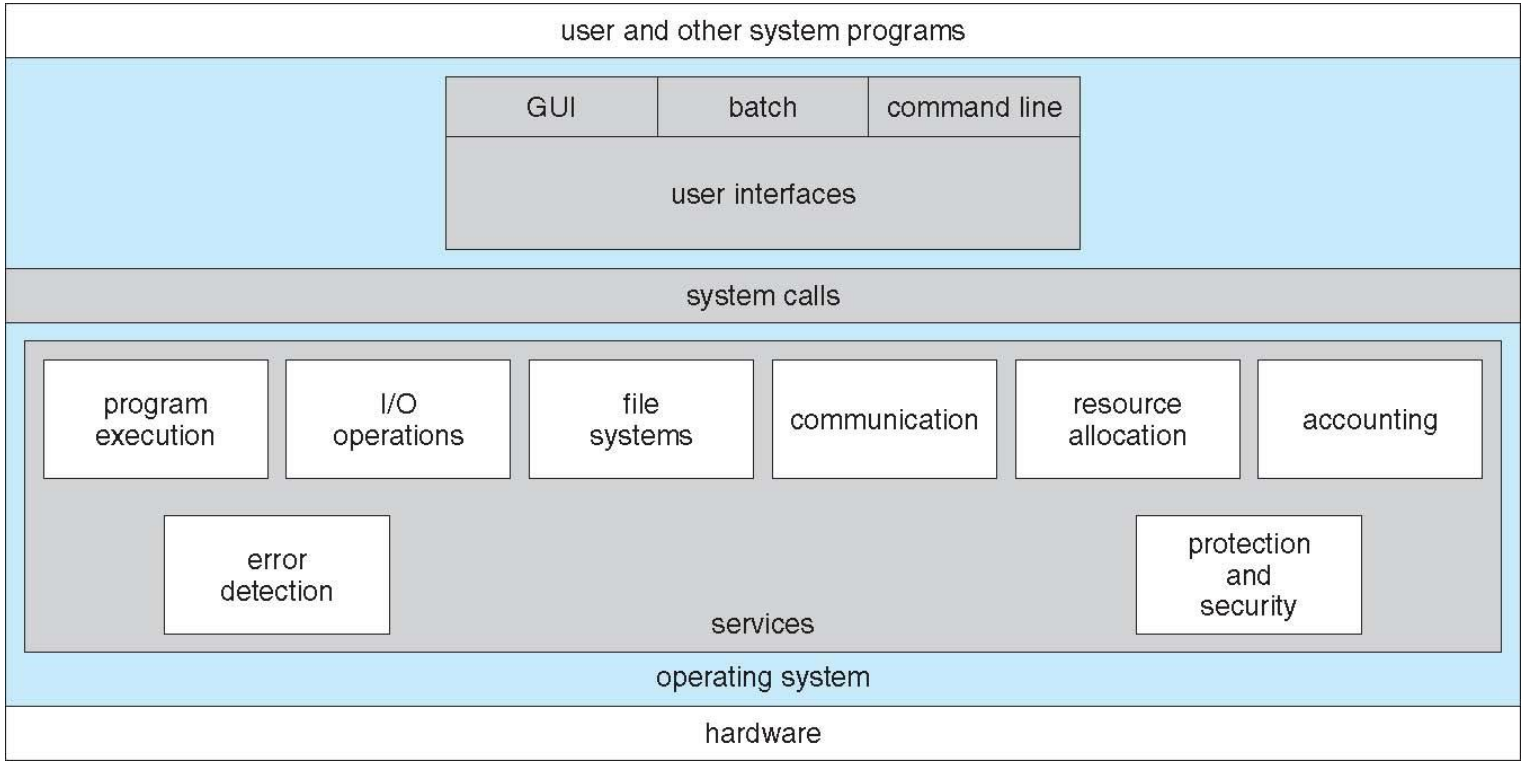
Operating System Services (Cont)

- One set of operating system services provides functions that are helpful to the user (Cont):
  - Communications
    - Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - Error detection
    - OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer' abilities to efficiently use the system

Operating System Services (Cont)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - Resource allocation
    - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
  - Accounting
    - To keep track of which users use how much and what kinds of computer resources
  - Protection and security
    - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **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
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

A View of Operating System Services

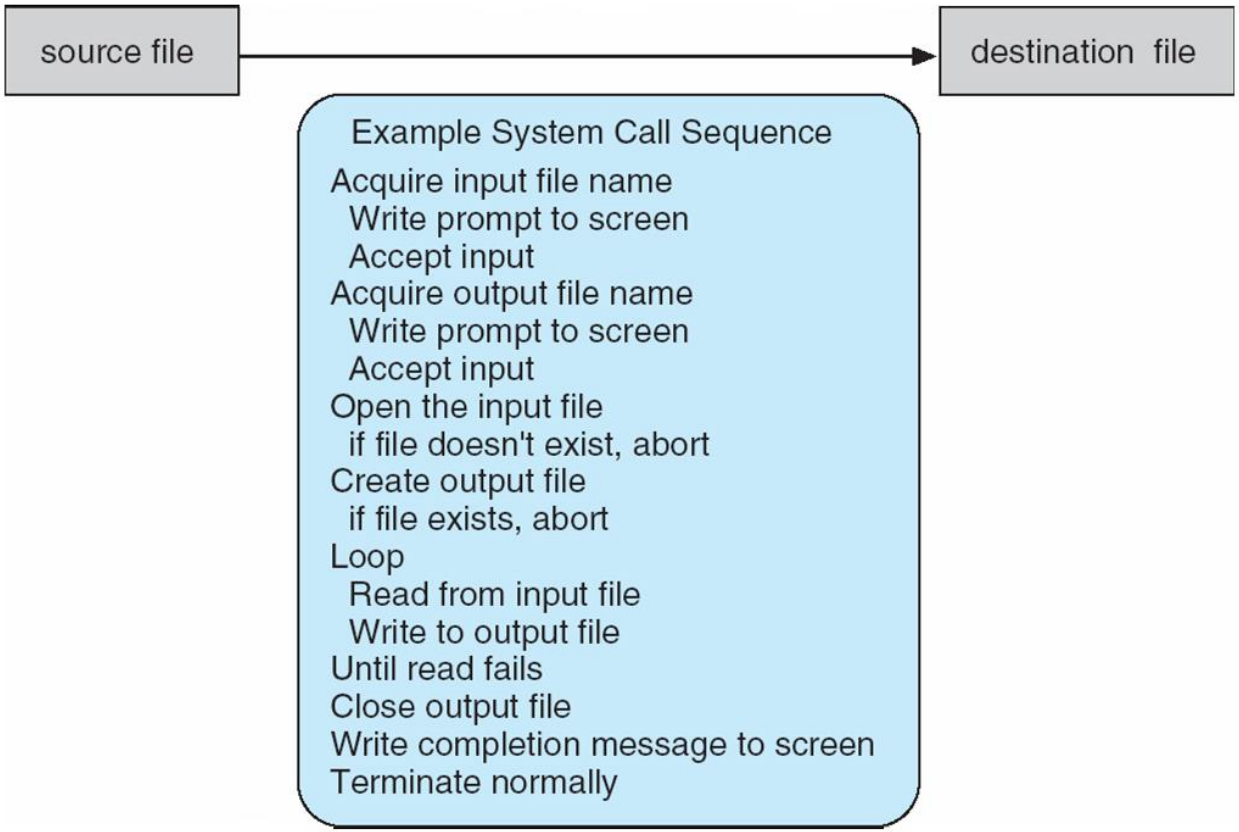


System Calls

- 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:
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (UNIX, Linux, and Mac OS X)
  - Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?
  - Portability
  - Ease of use

Example of System Calls

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

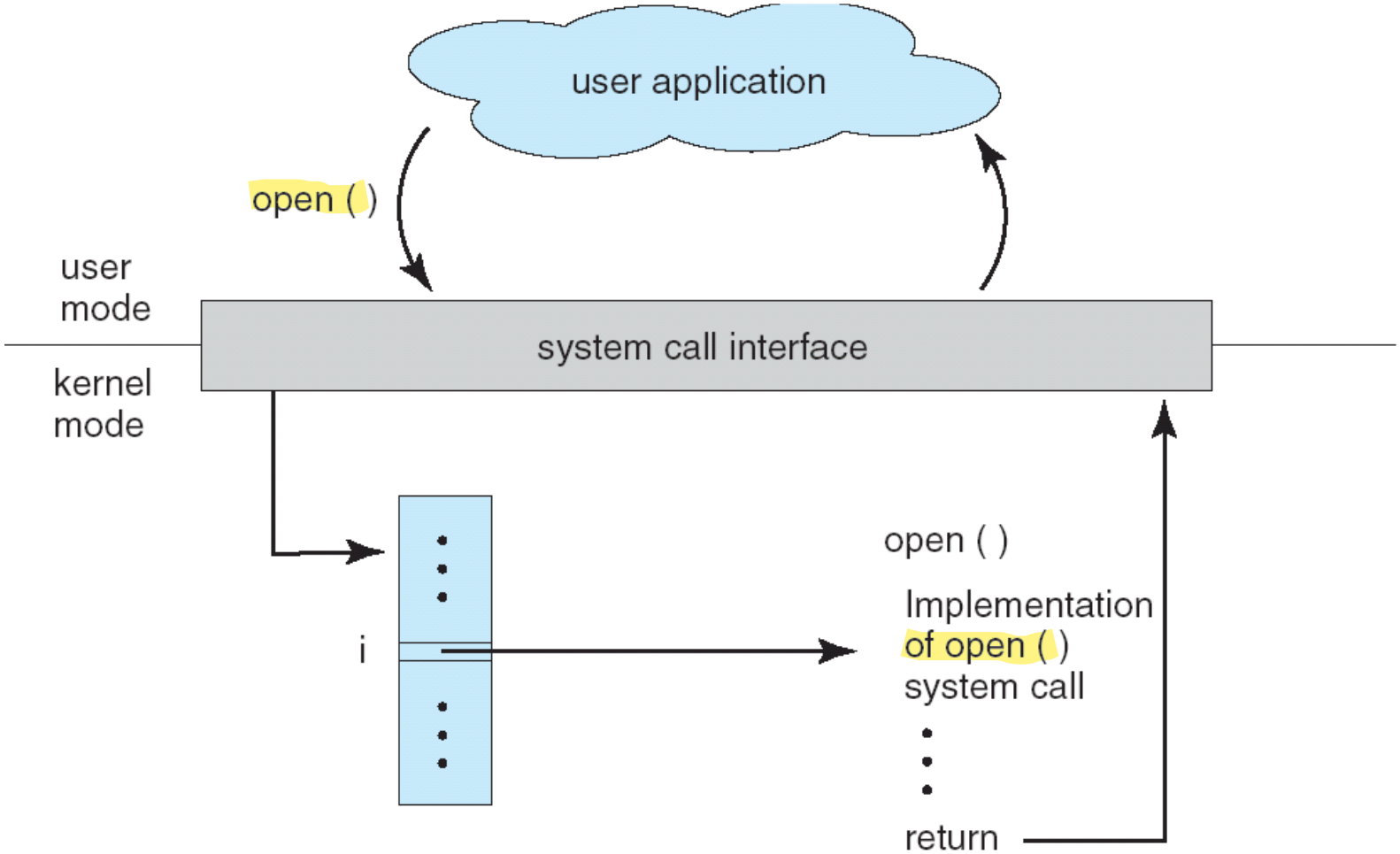


System Call Implementation

- Typically, a number associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- 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 to know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result
  - Most details of OS interface hidden from programmer by API

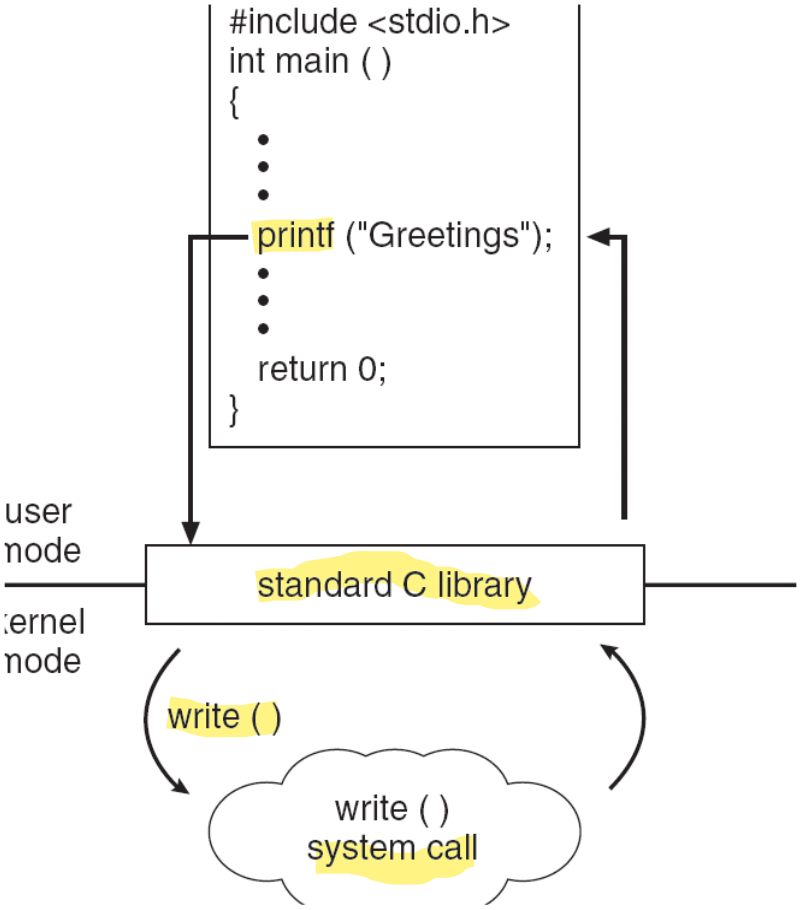
API – System Call – OS Relationship

- Use system call, directly



Standard C Library Example

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



Types of System Calls

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

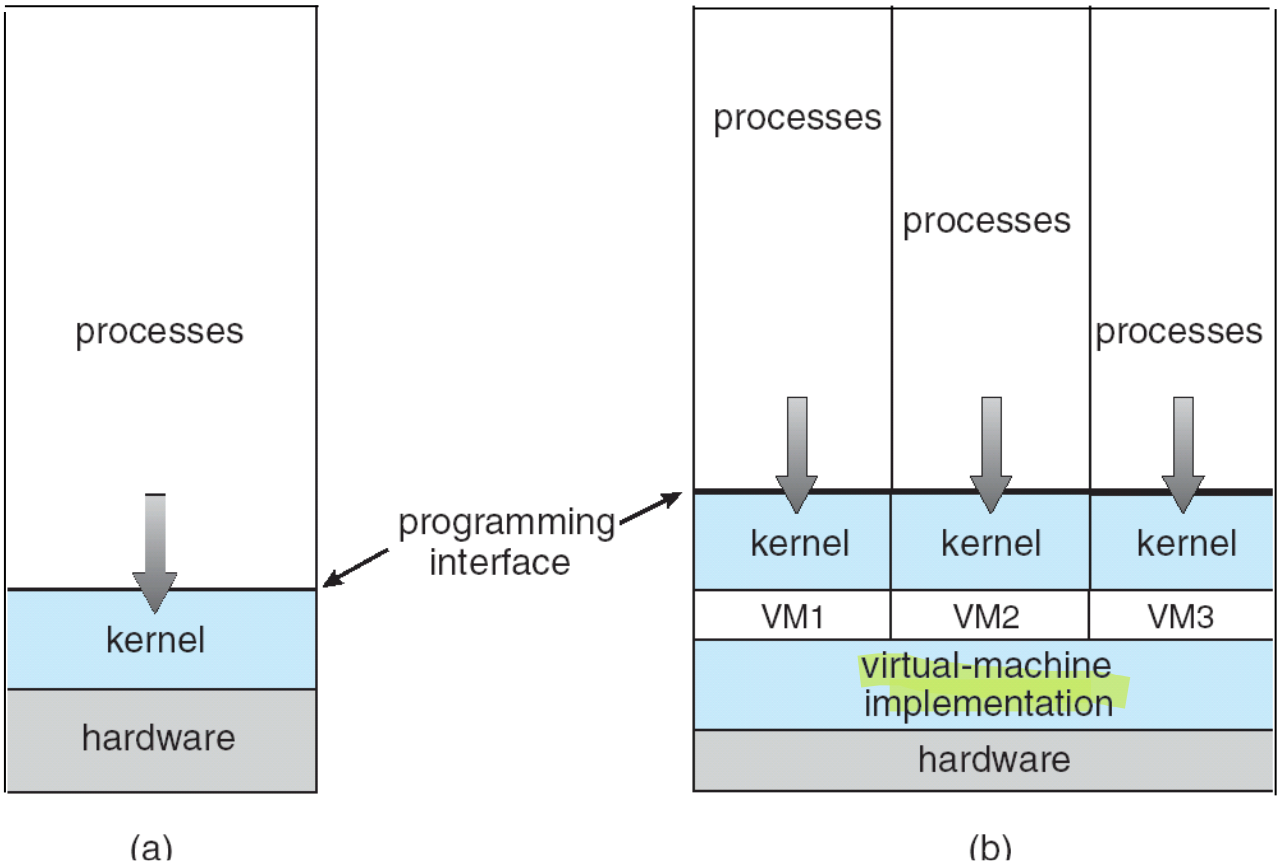
Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Virtual Machines

- Virtual machine
  - Takes the layered approach to its logical conclusion
  - Treats hardware and the operating system kernel as though they were all hardware
  - Provides an interface *identical* to the underlying bare hardware
- The operating system host creates the illusion that a process has its own processor and (virtual memory)
- Each guest provided with a (virtual) copy of underlying computer

Virtual Machines



(a) Nonvirtual machine

(b) virtual machine

The Java Virtual Machine

"Write (Compile) once, run anywhere"