Nanyang Polytechnic

IT2164/IT2561 Operating Systems and Administration

Chapter 2

Operating System Organization

Operating System Organization

At the end of this chapter, you should be able to

- List and describe the common services offered by an OS
- Describe the functions of an operating system
- Understand the operating system requirements
- Describe the basic operating system organization
- Explain the general implementation issues and implementation mechanisms
- Understand modern OS architecture

Operating System Services

- An operating system provides an environment for the execution of programs
- Different operating systems are organized along different lines
- As such, specific services may differ, but common classes exist
  - Those that provide convenience for user/programmer
  - Those that ensure efficient operations of the system

Operating System Services

- Services that provides convenience to user/programmer
  - User interface
    - Batch interface, command line interface, GUI
  - Program execution
    - Able to load, run and terminate a program
  - Input/Output operations
    - File, CD/DVD drive, Display device, SmartCard
  - File-system manipulation
    - Create/Read/Write/Delete files and directories
    - Permission management
  - Communications
    - Local/remote inter-process communication
  - Error detection
    - Able to detect error and take proper action to ensure consistent computing

Operating System Services

- Services that ensures efficient operation of the system
  - Resource allocation
    - Resources to be allocated among multiple users fairly
    - Different algorithms for different resources
  - Accounting
    - Keep track of usage statistics eg CPU, Printer, Harddisk quota
    - Reconfigure system to improve computing services
  - Protection and security
    - Security of system from outsiders
    - Ensure access to all system resources is controlled
    - Audit trail of access

Functions of an Operating System

- Regardless of services offered, there are four major group of basic functions common to all OS
  - Device management
  - Process, thread and resource management
  - Memory management
  - File management

Basic Operating System Organization

- Based on the four functions, the modern OS implements 4 major managers, which provides abstractions to the respective resource that they manage.
  - Process, Thread and Resource Manager
  - Memory Manager
  - Device Manager
  - File Manager
- Close interactions between the four managers

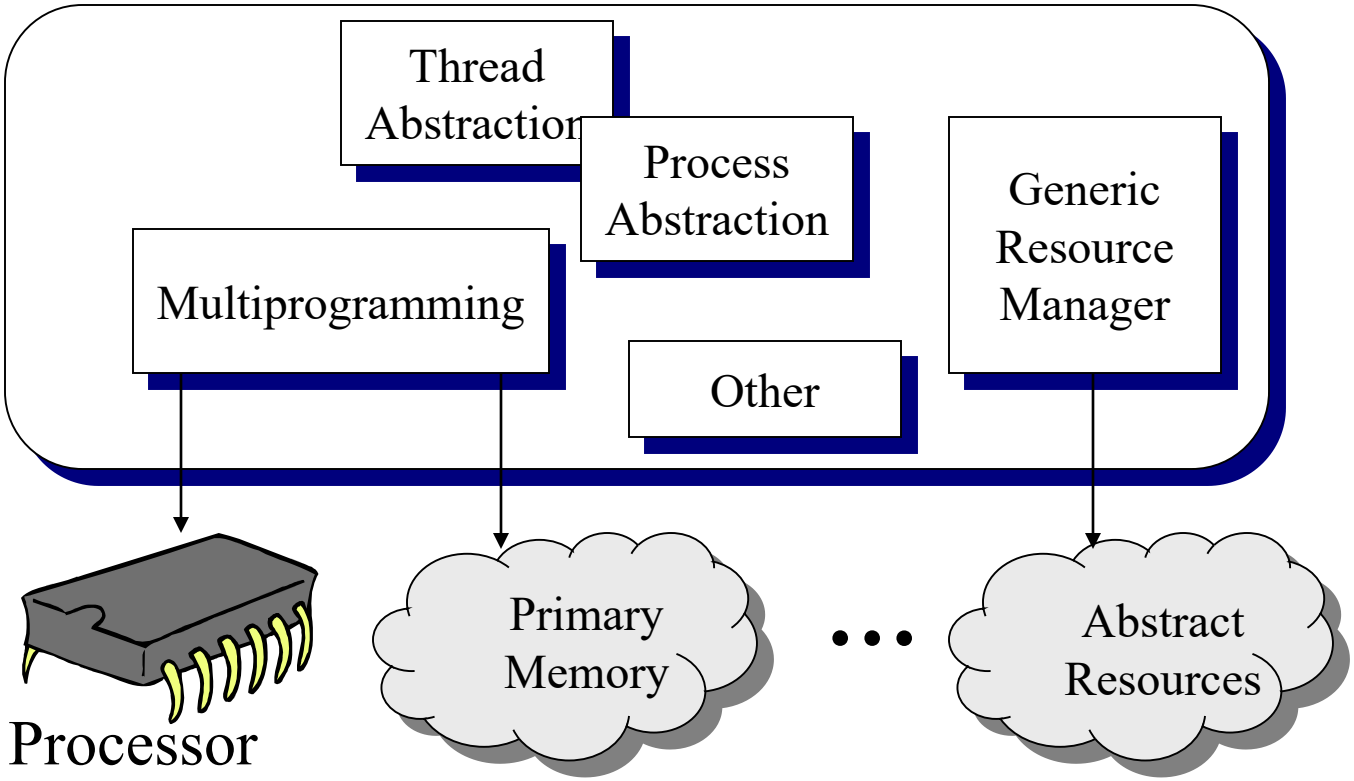
Device Management

- Refers to the way generic devices are handled. Includes disk, tapes, terminals, printers etc
- Special management approaches for processor and memory
- Partitioning design simplifies adding and upgrading of devices

Process, Thread and Resource Management

- Creates abstractions of processes, threads, resources
- Allocates processor resource equitably
- Allocates and tracks abstract resource such as queues, semaphores, messages
- Cooperates with memory manager to administer the primary memory

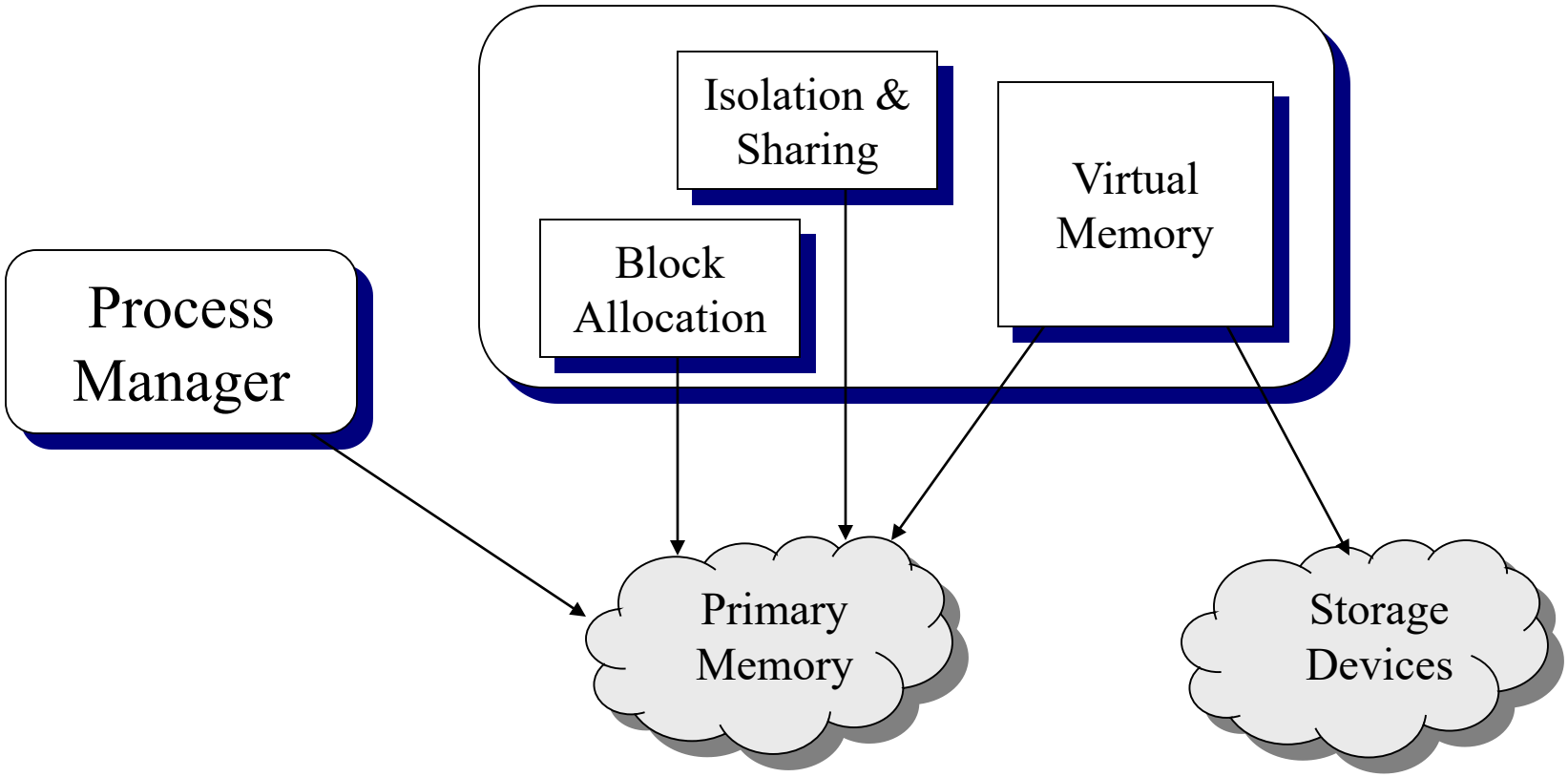
Process, Thread, and Resource Management



Memory Management

- Administer and allocate primary memory
- Enforces resource isolation
- Enables sharing between processes
- Provides virtual memory extensions
  - Abstract machine's memory appear larger than physical memory

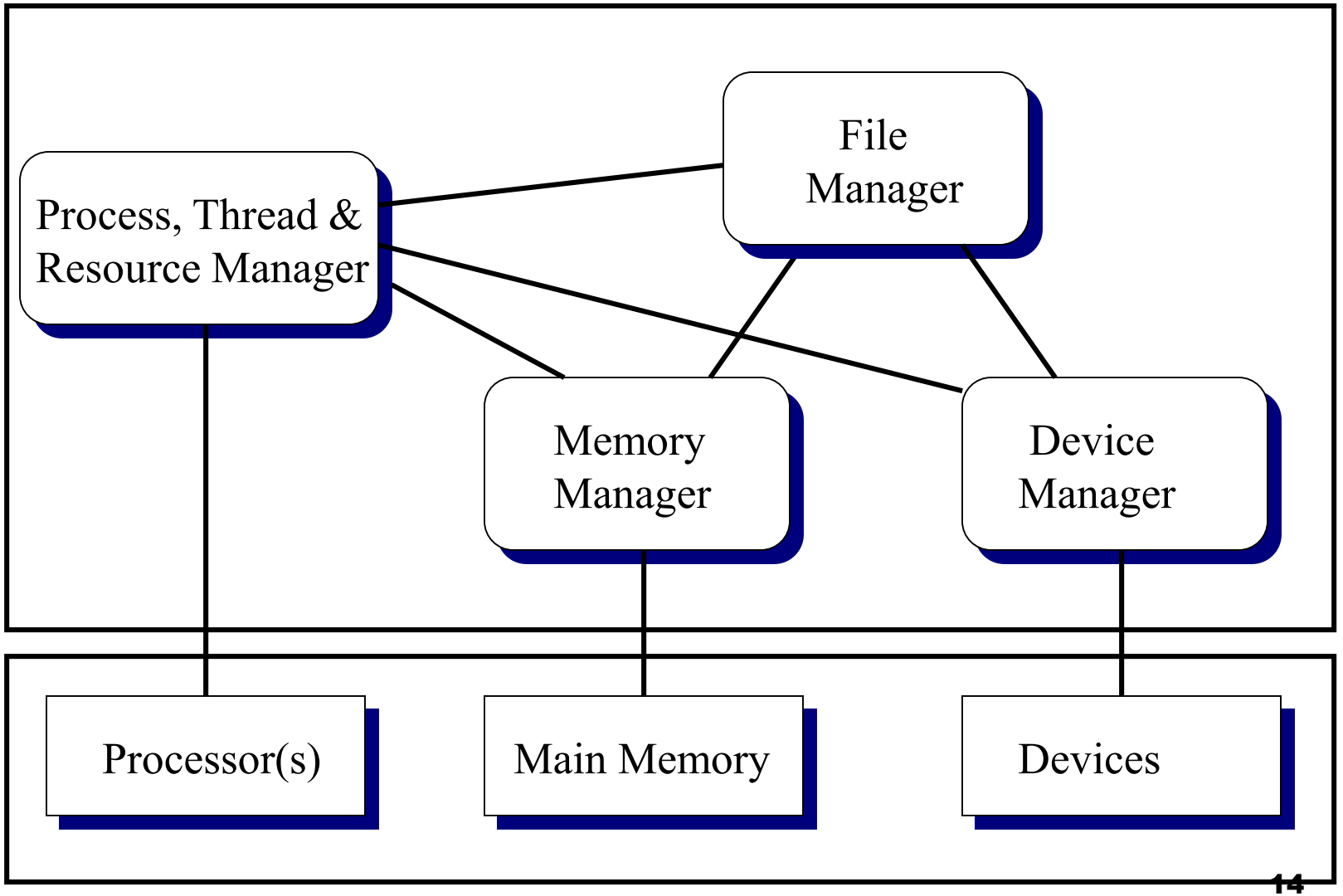
Memory Management



File Management

- Creates abstraction of storage devices i.e. I/O operations
- Range from byte stream files to indexed records
- Local and Remote file systems

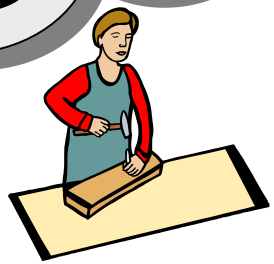
Basic Operating System Organization



Key OS Requirements

Processes

Coordinate Use of the Abstractions



Create the Abstractions

OS Requirements

- Manage resource sharing
  - Time/space-multiplexing
    - Using time/space-multiplexing where appropriate.
  - Exclusive use of a resource
    - Allow processes to use a resource exclusively as required.
  - Isolation
    - Allow a resource to save information without fear of it being modified or tampered with.
  - Managed sharing
    - Sharing must be done in an orderly fashion according to the properties of the resource. E.g., printer vs disk drive.

Implementation mechanisms

- Three basic mechanisms to address isolation and sharing :
  - Processor modes
  - Kernels
  - Method of invoking system service

Processor Modes

- Distinguish between **trusted** and **un-trusted** software
- Determine execution capability and accessible memory areas
- Modern processors provide 2 modes
- Mode bit: *Supervisor* or *User* mode
- Supervisor mode (for OS)
  - Can execute all machine instructions
  - Can reference all memory locations
- User mode (for user programs)
  - Can only execute a subset of instructions
  - Can only reference a subset of memory locations

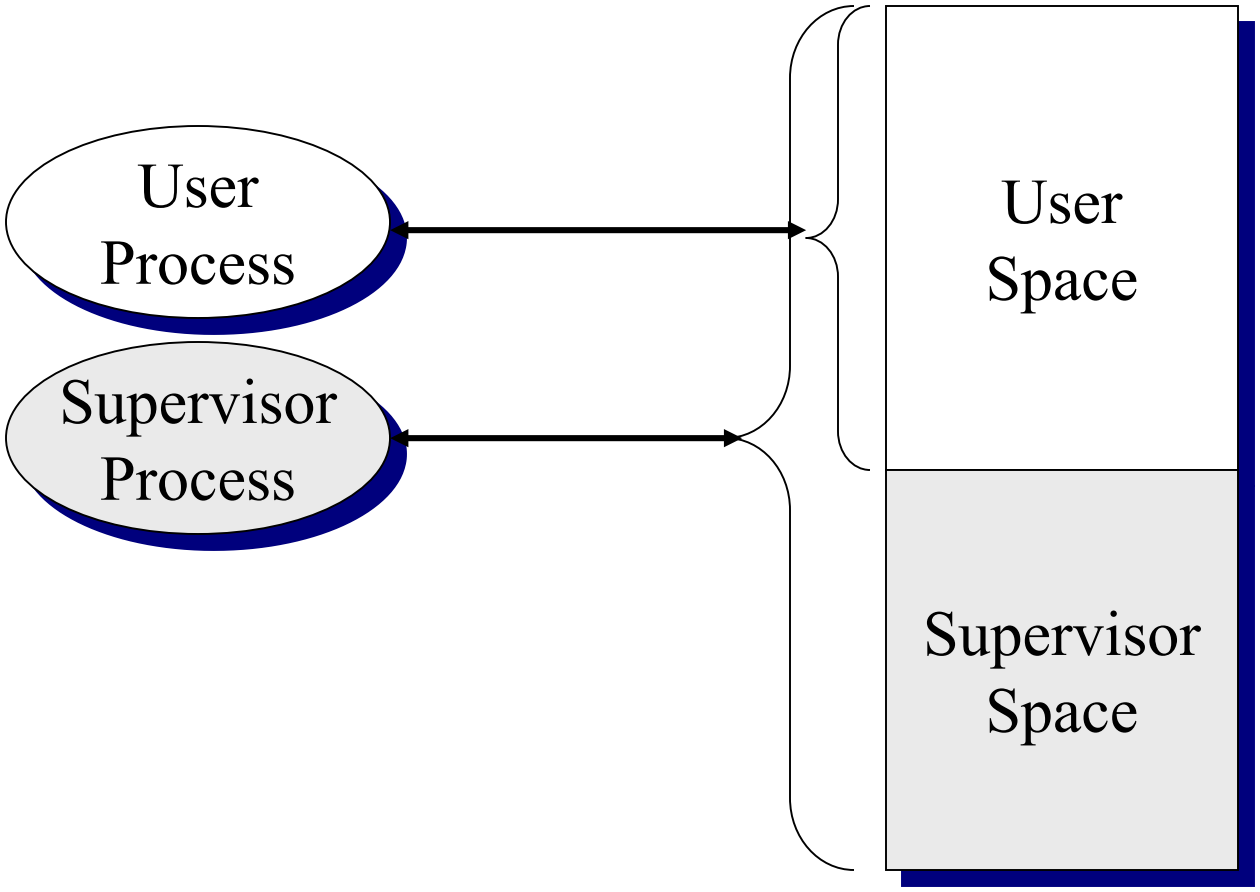
Processor Modes - Execution capability

- Processor in supervisor mode can execute all instructions
- Including **privileged** instructions. (Also called **supervisor** or **protected**.)
- Examples of privileged instructions :
  - I/O instructions
  - Memory-related instructions
  - Processor mode-change instructions
- Processor in user mode can execute only non-privileged instructions.

Processor Modes - Accessible memory

- Processor in supervisor mode can access all memory locations:
  - **System** (or **supervisor**, **kernel** or **protected**) space which refers memory area used by the OS
  - **User space** refers to memory area used by application processes
- Processor in user mode can access only user space

Processor Modes: Supervisor and User Memory



Processor Modes

- Trusted OS software executes in supervisor mode
- All other software (including some parts of the OS) executes in user mode
- Concept allows for OS to be able to control access to resources
  - User programs have to ask the OS to execute privileged instructions on their behalf.
  - Any particular configuration can isolate or permit sharing or resources according to the administrator's policy

Kernels

- The part of the OS critical to correct operation (trusted software)
- Implements the basic mechanisms that assure secure operation of entire OS
- Executes in supervisor mode
- The `trap` instruction is used to switch from user to supervisor mode, entering the OS

Requesting Services from OS

- In order to execute privileged instructions, user programs have to activate routines in the kernel, which can then execute on the user programs' behalf.
- Two techniques :
  - System call
  - Message passing

System Call

- In system call, the relevant function is activated via a **trap** instruction.
- OS provides a sub function which the user program calls
- Stub function will switch the processor to supervisor mode
- It will execute the `trap` instruction by branching to a trap table to the entry point of the system function to be invoked
- On completion, processor is switched back to user mode and control returns to user process
- Appears as ordinary function call to the application programmer

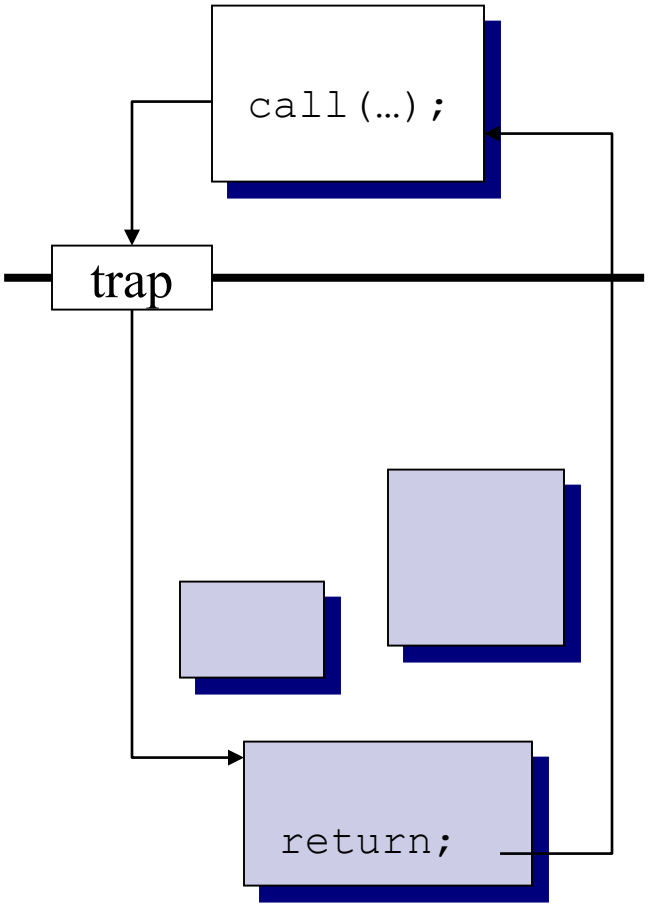
Message Passing

- In the message passing method, the user program constructs a message that requests the desired service.
- Uses OS `send()` system call
- OS kernel implements target function
- Kernel process must be started or active i.e. must be in supervisor mode, to receive message
- User process waits for result with `receive()` operation
- Kernel sends message back to user process on completion

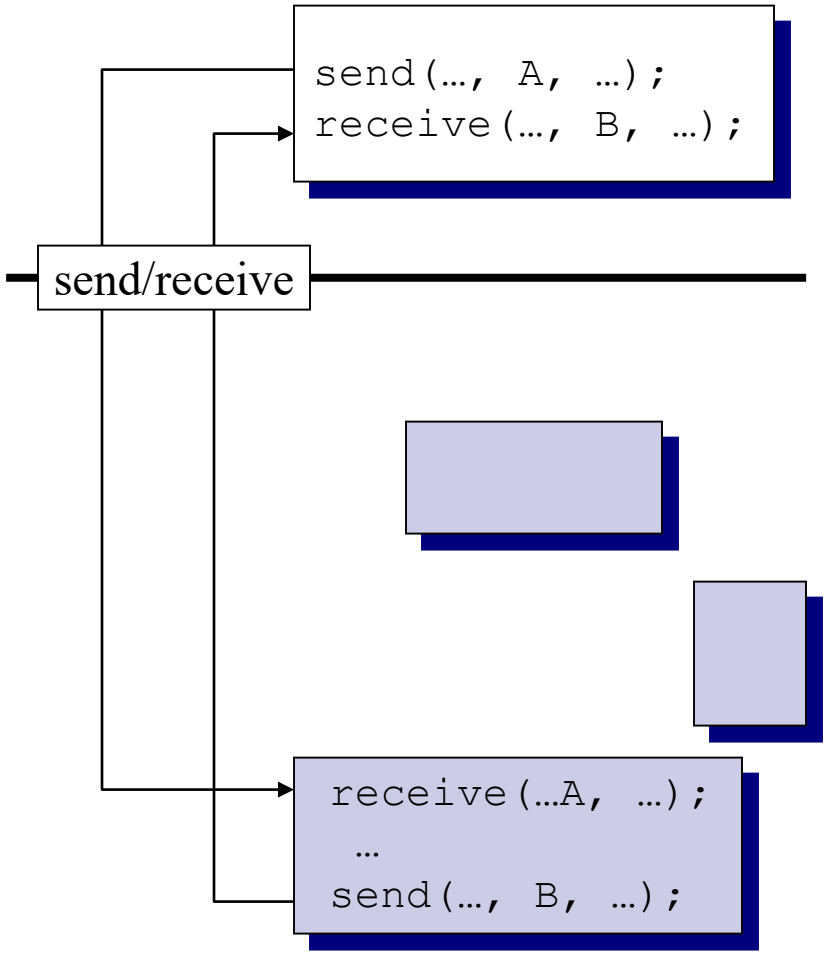
System Call versus Message Passing

- In system call, the user process/thread gains ability to execute privileged instructions
- In the message passing method, the system function is executed by the kernel process/thread
- System calls are more efficient than message passing.
  - Message passing has cost of message formation/copying and process multiplexing
  - System calls just requires a trap command.
- Most modern systems use system calls.

System Call and Message Passing Operating Systems

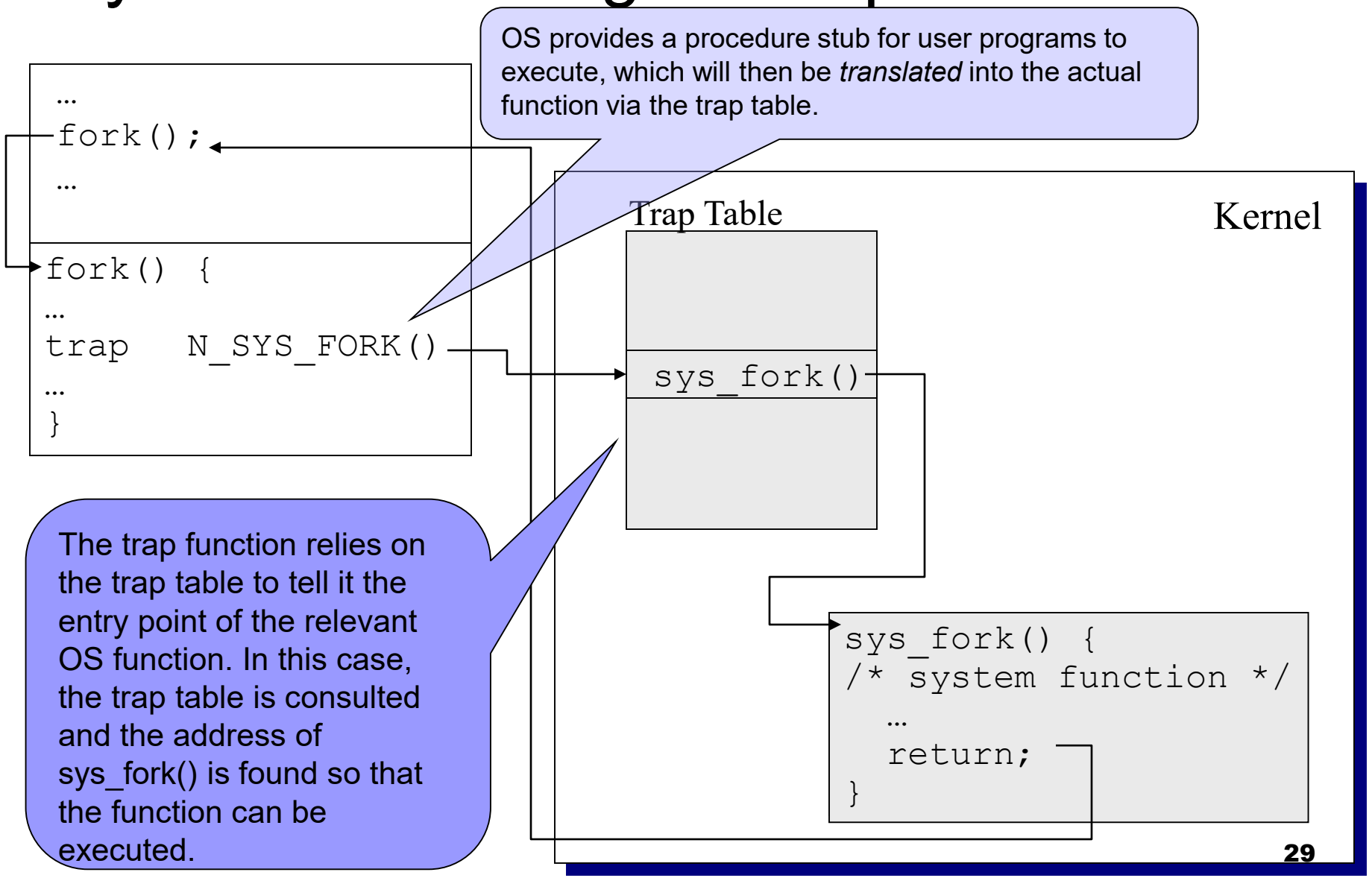


Procedure Call

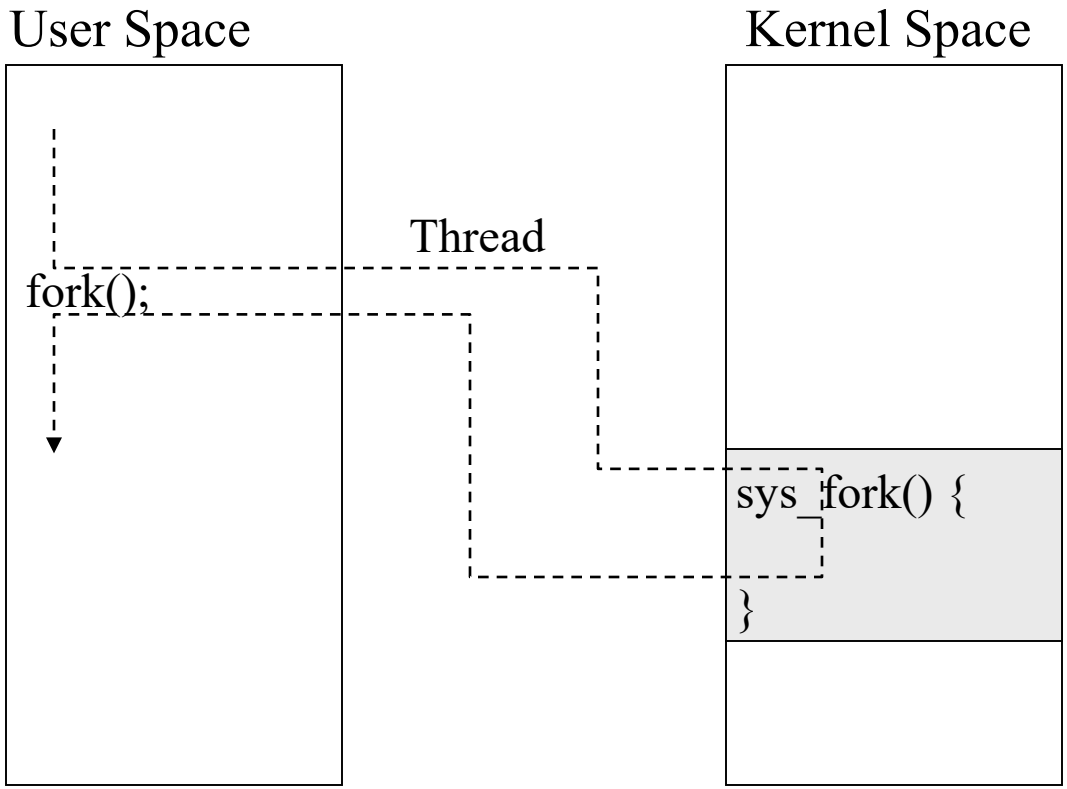


Message Passing

System Call Using the trap Instruction



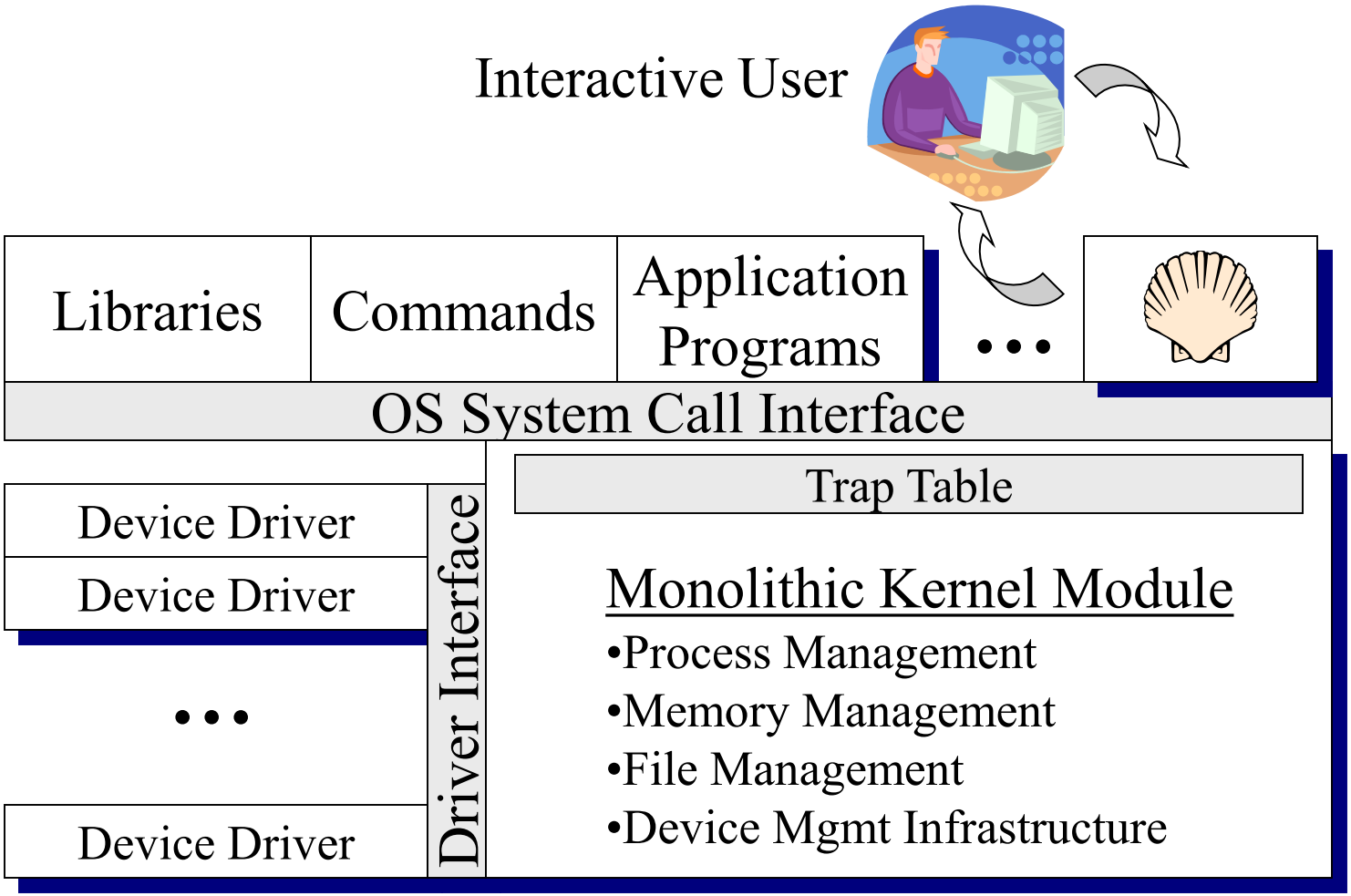
A Thread Performing a System Call



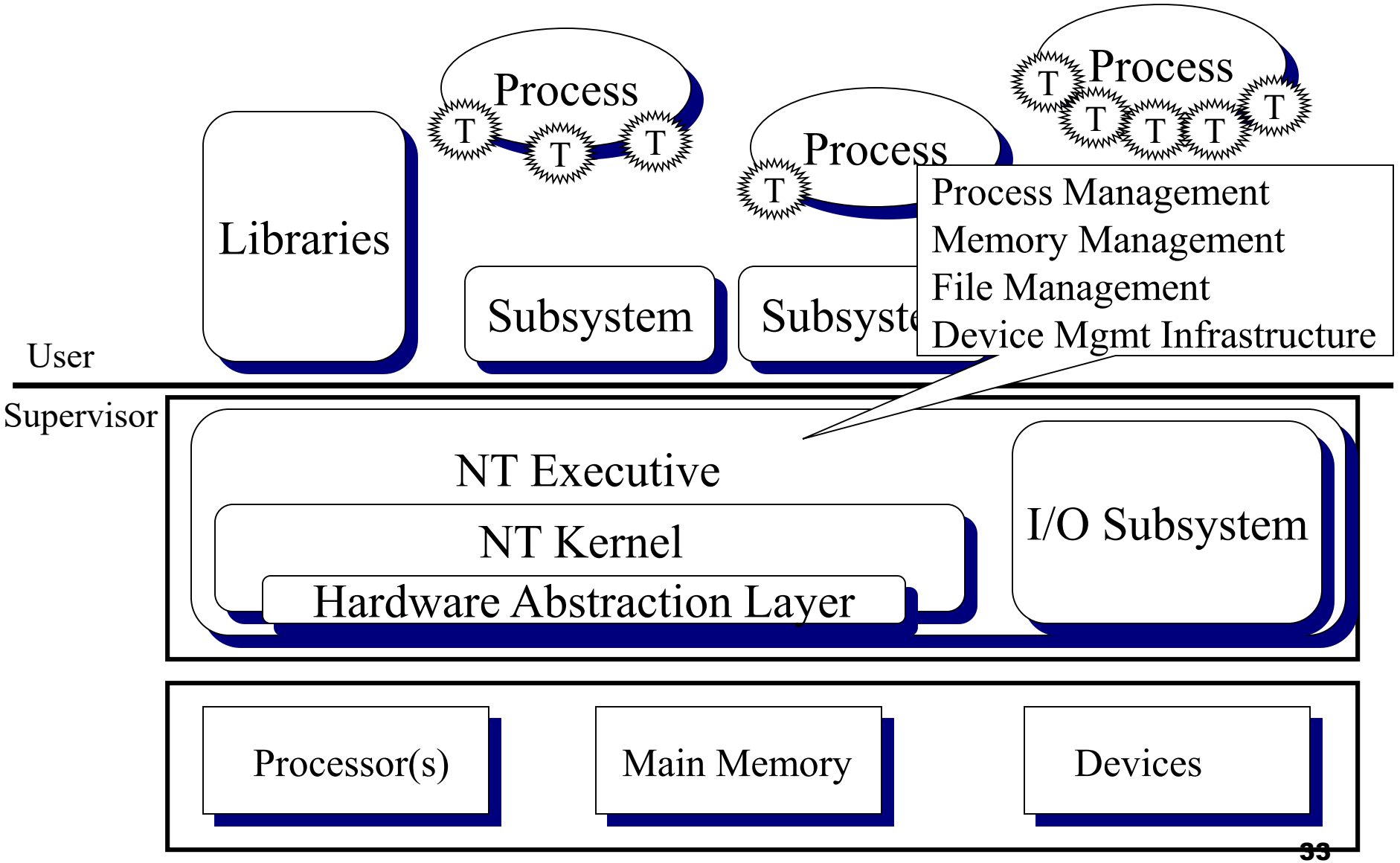
Modern OS architecture

- A modular OS architecture implements each manager in its own software module
- Interaction among various managers via abstract data type
- Frequent calls incurs performance penalty
- Sacrifice modularity for performance
  - Monolithic kernel implementation
    - Four basic modules are combined into a single software module
  - Microkernel approach
    - Employs a small kernel that implements only the essential and critical functions
    - Reminder functions are implemented outside kernel, possibly in separate modules

The UNIX Architecture



Windows NT Organization



Conclusion

- OS implements an environment which includes processes, resources and facilities to manage resources.
- Modern OS incorporate managers for processes and resources, including memory, files and devices.
- Processor modes, kernels and a method of invoking system service allows the OS to achieve resource abstraction and sharing.