Module 3: Operating-System Structures

- System Components
- Operating System Services
- System Calls
- System Programs
- System Structure
- Virtual Machines
- System Design and Implementation
- System Generation

Common System Components

- Process Management
- Main Memory Management
- Secondary-Storage Management
- I/O System Management
- File Management
- Protection System
- Networking
- Command-Interpreter System

Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

I/O System Management

- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.

Networking (Distributed Systems)

- A *distributed* system is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Command-Interpreter System

- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking

Command-Interpreter System (Cont.)

- The program that reads and interprets control statements is called variously:
  - control-card interpreter
  - command-line interpreter
  - shell (in UNIX)

  Its function is to get and execute the next command statement.

Operating System Services

- Program execution – system capability to load a program into memory and to run it.
- I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation – program capability to read, write, create, and delete files.
- Communications – exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via *shared memory* or *message passing*.
- Error detection – ensure correct computing by detecting errors in the CPU and memory hardware, in I/O devices, or in user programs.

Additional Operating System Functions

Additional functions exist not for helping the user, but rather for ensuring efficient system operations.

- Resource allocation – allocating resources to multiple users or multiple jobs running at the same time.
- Accounting – keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection – ensuring that all access to system resources is controlled.

System calls

interface between the operating system and the outside

available in assembly language and languages like C, Perl designed to replace assembly

Not available directly in Java (cf: portability)

An example of use of system call: the copy of a file

```
cp file1 file2
```

A program that implements `cp` would make use of system calls:

- those for reading the names of the two files
- opening the input file (possible errors: file does not exist, file protected)
- creating the output file
- if a file named file2 already existed: delete a previous file file2, or return an error, or ask user for action
- in case of errors: print message and exit
- if no error: a loop where characters are read from file1 and copied to file2 (2 system calls, with possible errors returned in case of a problem, such as space on disk exhausted)
- close both files
- terminate

`cp` is an example of **system program**, also called **utility**

Utilities like `cp` hide all these details to the user

The syntax of each system call depends on

- the particular operating system
- the operation involved (for instance a system call for a read operation might need to specify the input device used, the length of the data to transfer, the position of memory where the trasfer should go to)

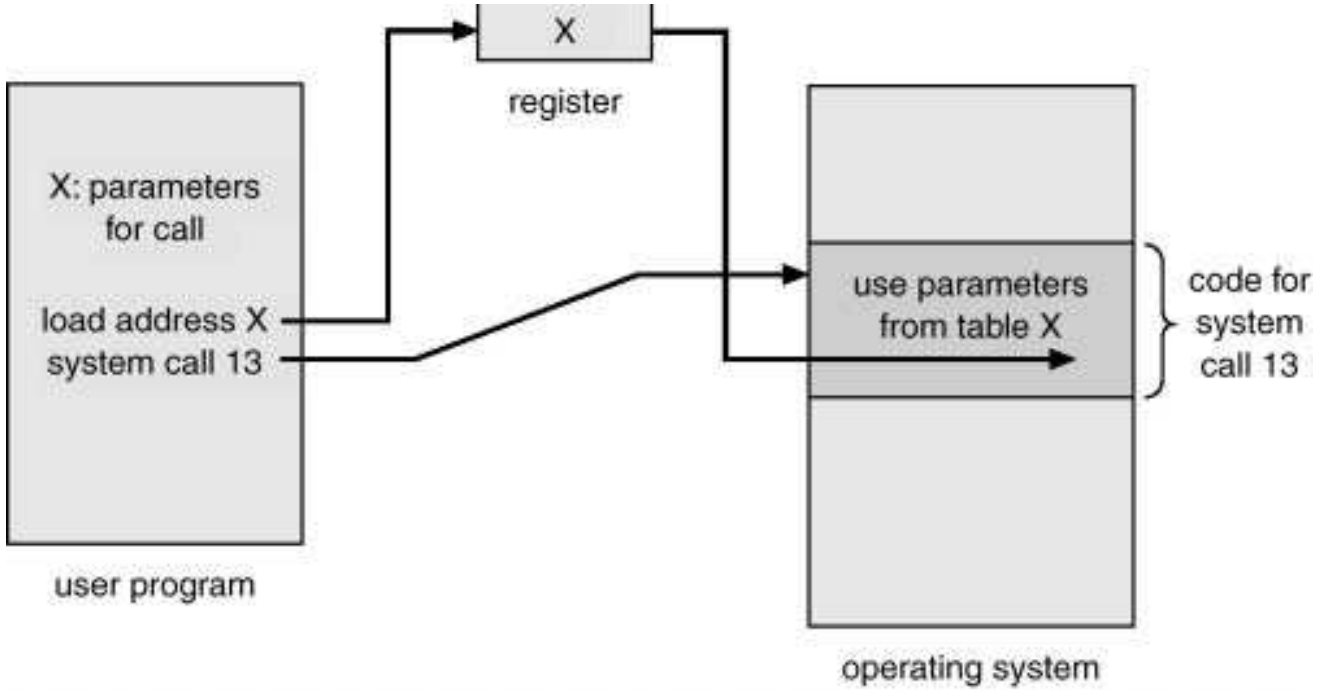
Main groups of system calls:

- process management
- file manipulation
- device manipulation
- communication

System Calls

- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C. Bliss, PL/360)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system.

Passing of Parameters As A Table



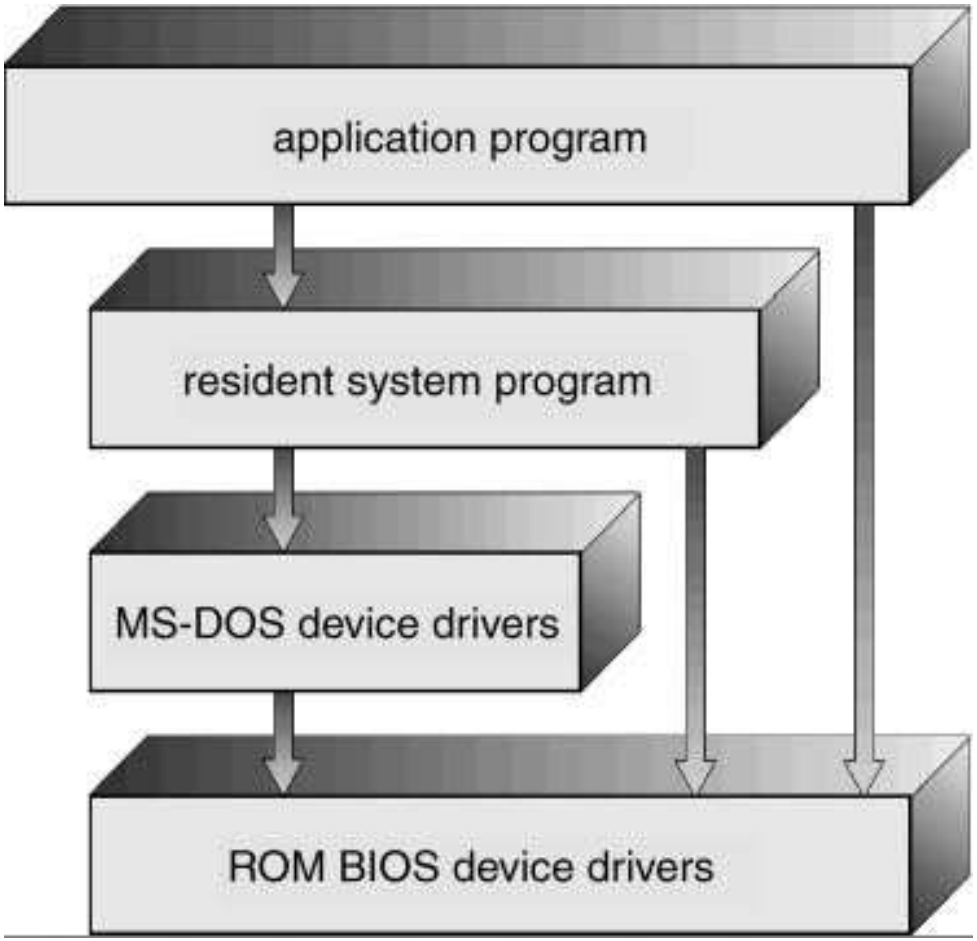
System Programs

- System programs provide a convenient environment for program development and execution. The 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 Structure – Simple Approach

- 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

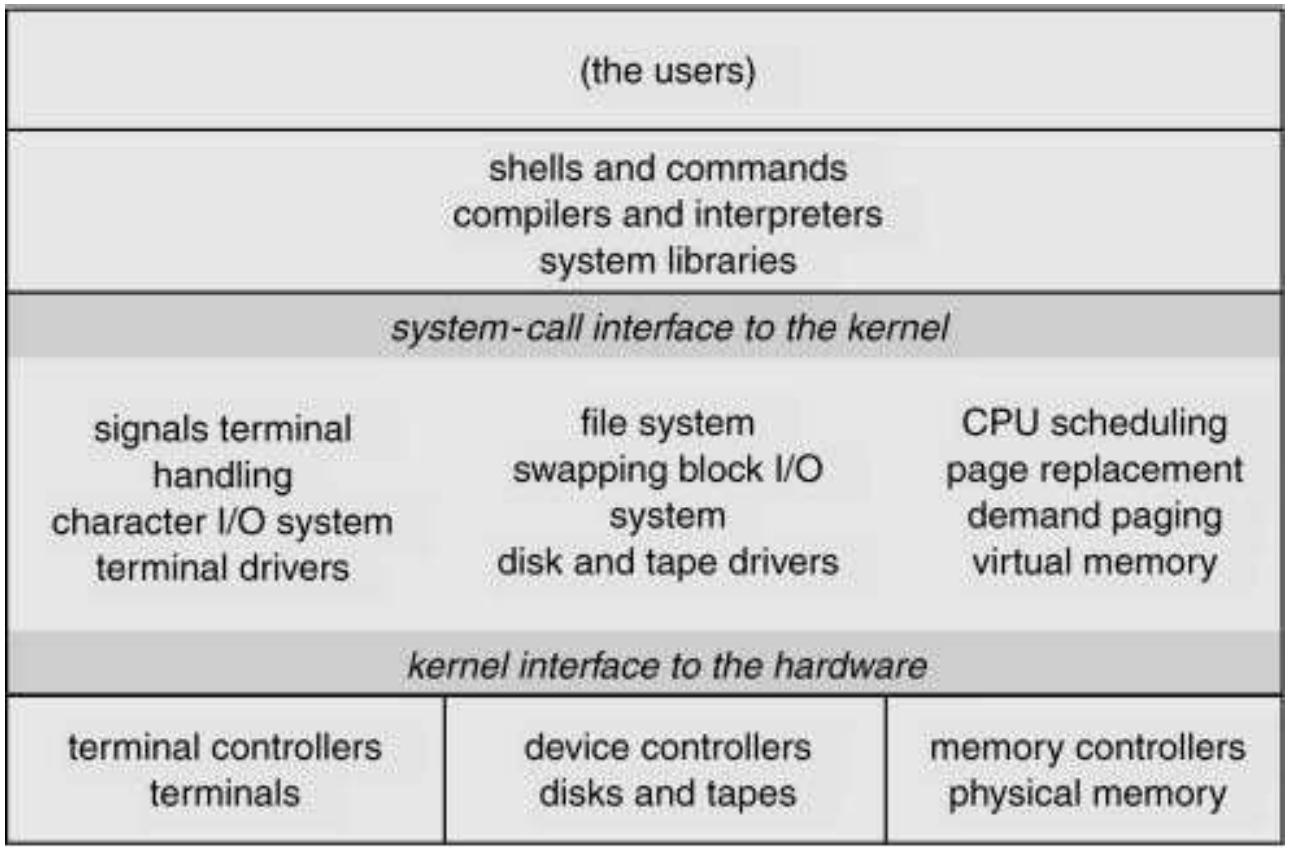
MS-DOS Layer Structure



System Structure – Simple Approach (Cont.)

- 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.

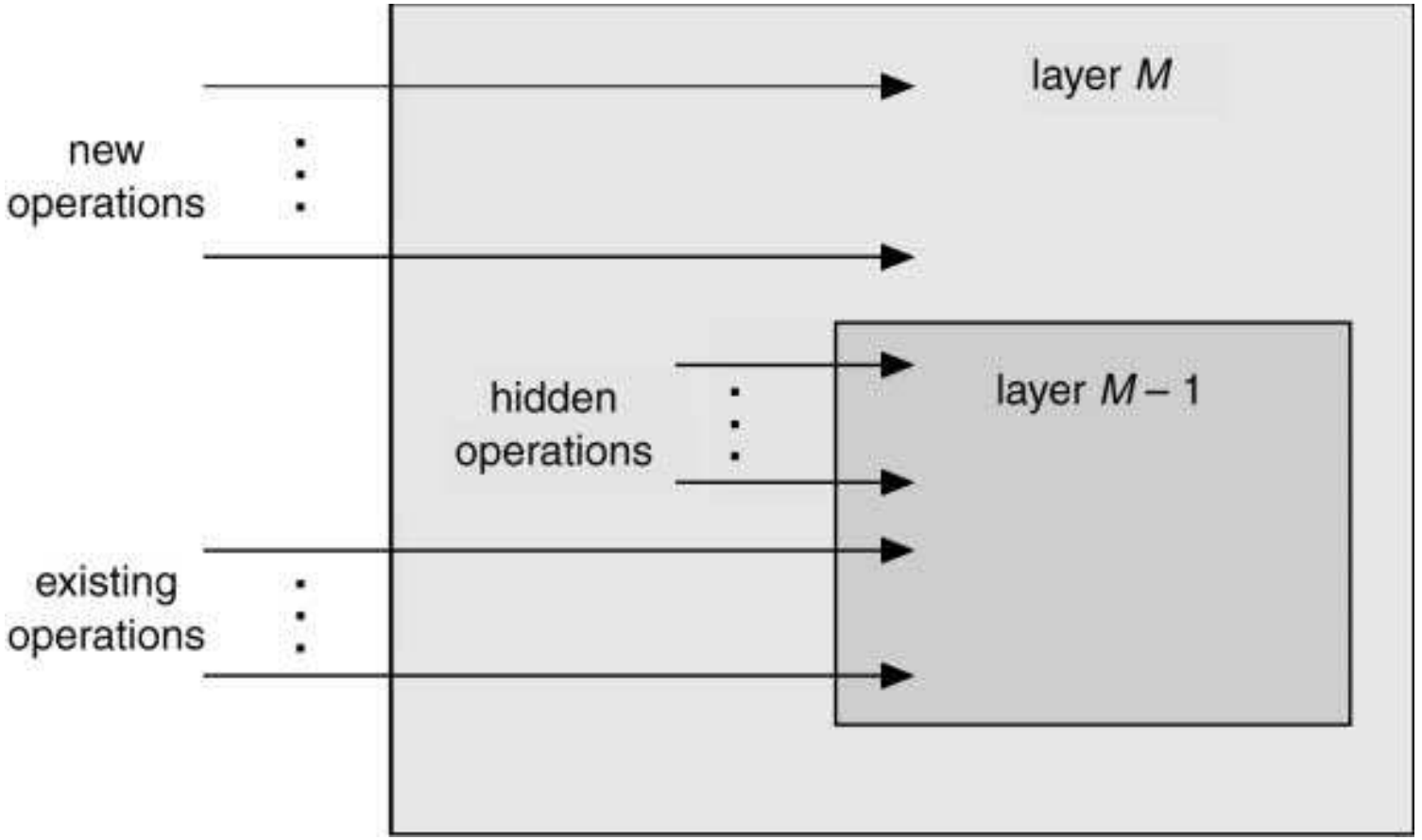
UNIX System Structure



System Structure – 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.

An Operating System Layer

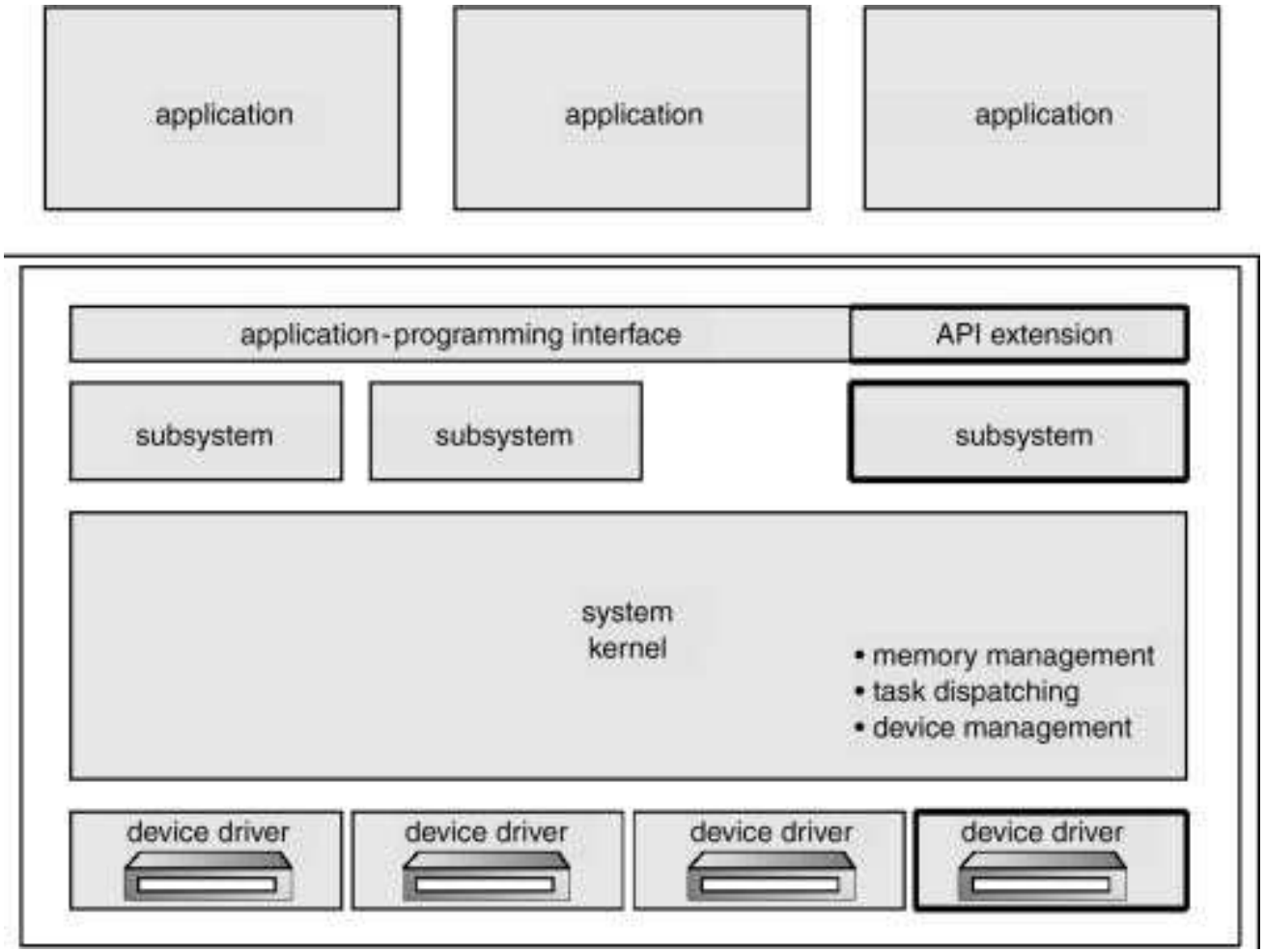


Layered Structure of the THE OS

- A layered design was first used in THE operating system. Its six layers are as follows:

layer 5:	user programs
layer 4:	buffering for input and output
layer 3:	operator-console device driver
layer 2:	memory management
layer 1:	CPU scheduling
layer 0:	hardware

OS/2 Layer Structure



Microkernel

Another trend in the design of OS

The kernel contains only the very essential features:

- basic process and memory management,
- some communication facility, normally message-passing (this is the most important feature of a microkernel)

The main services of the OS are added on top, as modules that interact with each other using the communication facility of the microkernel

Advantages of microkernel:

- extension
- portability
- easier to modify (to modify a service, only one module is touched)
- reliability (if a service fails, the rest of the SO remains untouched)

Examples: Apple MacOS, WindowNT, and Linux (partly)

Windows NT Client-Server Structure