Operating-System Services

- Operating system provides environment for execution of programs.
- Operating systems provides services to programs and users that use those programs.
- We identify common classes of services for all operating systems.
- **One set of operating-system services provide functions helpful to the user:**
    - **User interface**
        - Almost all operating systems have a user interface (UI)
        - Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
    - **Program execution**
        - The system must be able to load a program into memory and run it
        - The program must be able to end its execution, (ab)normally
    - **I/O operations**
        - For specific devices, special functions may be desired (e.g. to rewind a tape drive or to blank a CRT screen)
        - For efficiency and protection, users can't control I/O devices directly, so the OS must provide a means to do I/O
    - **File-system manipulation**
        - Programs need to read, write, create, and delete files
    - **Communications**
        - Communications between processes may be implemented via shared memory, or by the technique of message passing, in which packets of information are moved between processes by the OS
    - **Error detection**
        - Errors may occur in the hardware, I/O devices, user programs…
        - For each type of error, the OS should take appropriate action

        - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system
- **Another set of operating-system functions exists to ensure the efficient operation of the system itself:**
    - **Resource allocation**
        - When multiple users are logged on, resources must be allocated
        - Some resources have a special allocation code, whereas others have a general request & release code
    - **Accounting**
        - You can keep track of which users use how many & which resources
        - Usage statistics can be valuable if you want to reconfigure the system to improve computing services
    - **Protection and Security**
        - Concurrent processes shouldn't interfere with one another
        - **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

User Operating-System Interface

- Two ways that users interface with the operating system:
    - Command Interpreter (Command-line interface)
    - Graphical User Interface (GUI)

Command Interpreter (Command-line interface)

- Main function of command interpreter is to get and execute the next user-specified command.
- Many of the commands are used to manipulate, create, copy, print, execute, etc. files.
- Two general ways to implement these commands:
    - Command interpreter self contains code to execute command;
    - Commands are implemented through system programs.

Graphical User Interface (GUI)

- No entering of commands but the use of a mouse-based window-and-menu system characterized by a **desktop** metaphor.
- The mouse is used to move a pointer to the position of an icon that represents a file, program or folder and by clicking on it the program is invoked.

System Calls

- **System calls** provide an interface to the services made available by an operating system.
- Look at figure 2.4 p.56 TB for an example of a sequence of system calls.
- Application developers design programs according to an application programming interface (API).
    - The API defines a set of functions that are available to an application programmer.
    - This includes the parameters passed to functions and the return values the programmer can accept.

Operating-System Design and Implementation

- Problems faced in designing and implementing an operating system
- 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
- *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 principle to separate
    - **Policy:** What will be done?
    - **Mechanism:** How to do it?
- 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

Design Goals

- Firstly define goals and specification.
    - E.g. Convenience, reliability, speed, flexibility, efficiency…

Mechanisms and Policies

- Mechanisms determine how to do something
- Policies determine what will be done
- Separating policy and mechanism is important for flexibility
- Policies change over time; mechanisms should be general

Implementation

- OS's are nowadays written in higher-level languages like C / C++
- Advantages of higher-level languages: faster development and the OS is easier to port (i.e. move to other hardware)
- Disadvantages of higher-level languages: reduced speed and increased storage requirements

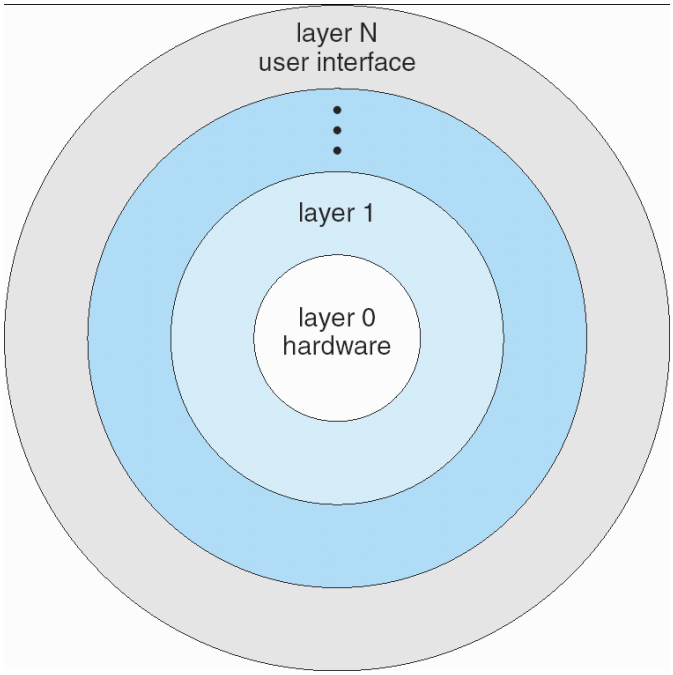
Operating-System Structure

Simple Structure

- MS-DOS and UNIX started as small, simple, limited systems

Layered Approach

- The OS is broken into layers: lowest = hardware, highest = GUI
- A typical layer has routines that can be invoked by higher ones
- Advantage: modularity (which simplifies debugging)
- A layer doesn't need to know how lower-level layer operations are implemented, only what they do
- Problems:
    - Layers can use only lower ones so they must be well defined
    - Layered implementations are less efficient than other types
- Nowadays fewer layers with more functionality are being designed



Microkernels

- Microkernel approach: all nonessential components are removed from the kernel and are implemented as system & user programs
- The smaller kernel provides minimal process & memory management
- Advantages:
    - Ease of extending the OS (new services are added to the user space and don't modify the kernel)
    - The OS is easier to port from 1 hardware design to another
    - More reliability: a failed user service won't affect the OS
- Main function of the microkernel: to provide a communication facility between the client program and the various services
- E.g. If the client program wants to access a file, it must interact with the file server indirectly through the microkernel
- QNX is a real-time OS that is based upon the microkernel design
- Windows NT uses a hybrid structure

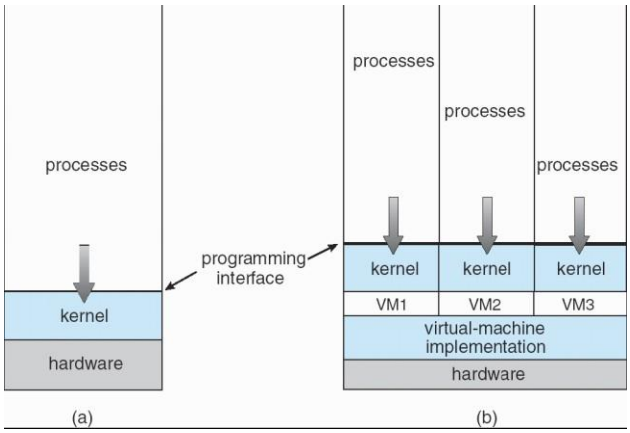
Modules

- 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

Virtual Machines

- A *virtual machine* 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
- 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



Non-virtual Machine Virtual Machine

- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine