

Chapter 3: Operating-System Structures

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

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

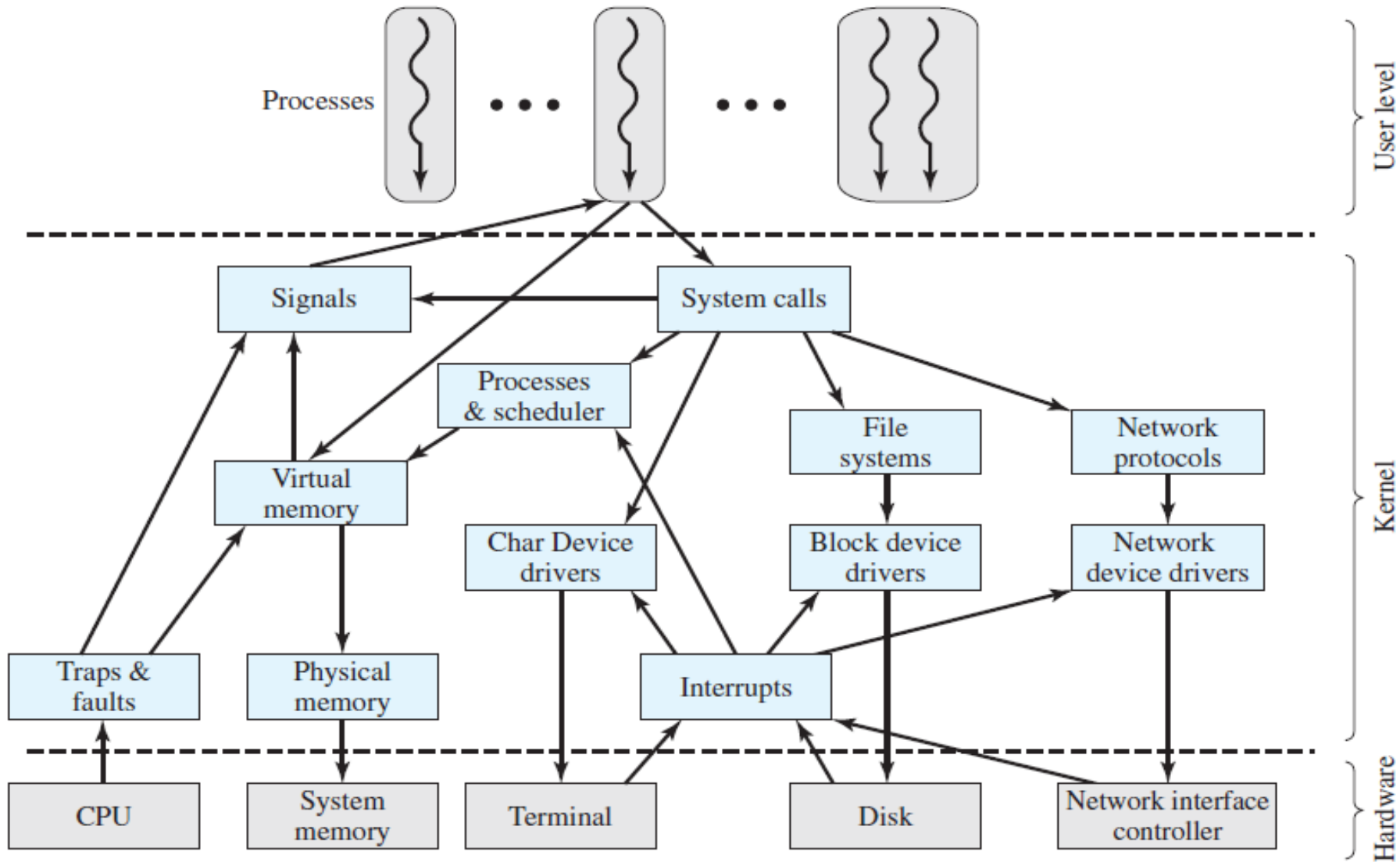
Additional functions exist not for helping the user, but rather for ensuring efficient system operations.

- 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

- 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
- 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.

Linux Kernel Components



System Calls (Cont.)

- Process control
 - end, abort
 - load, execute
 - create process, terminate process
 - get process attributes, set process attributes
 - wait for time
 - wait event, signal event
 - allocate and free memory
- File management
 - create file, delete file
 - open, close
 - read, write, reposition
 - get file attributes, set file attributes

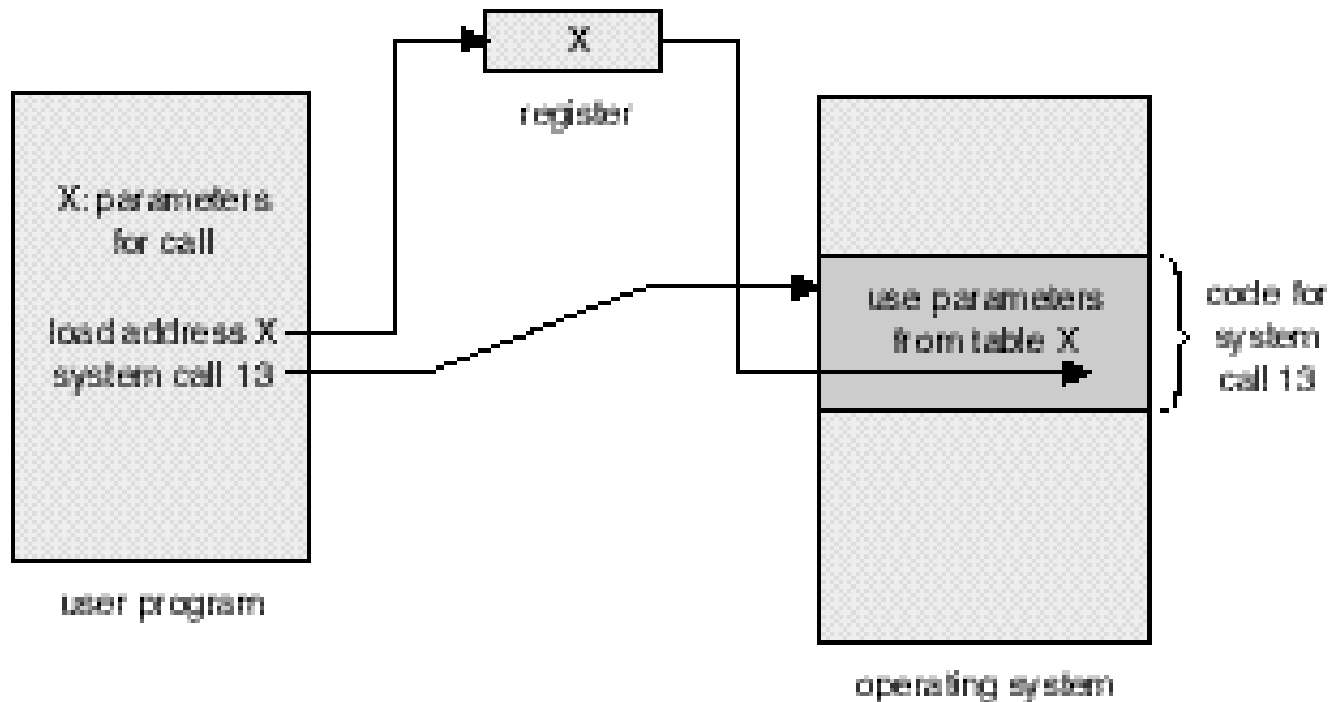
System Calls' Types (Cont.)

- Device management
 - request device, release device
 - read, write, reposition
 - get device attributes, set device attributes
 - logically attach or detach devices
- Information maintenance
 - get time or date, set time or date
 - get system data, set system data
 - get process, file, or device attributes
 - set process, file, or device attributes
- Communications
 - create, delete communication connection
 - send, receive messages
 - transfer status information
 - attach or detach remote devices

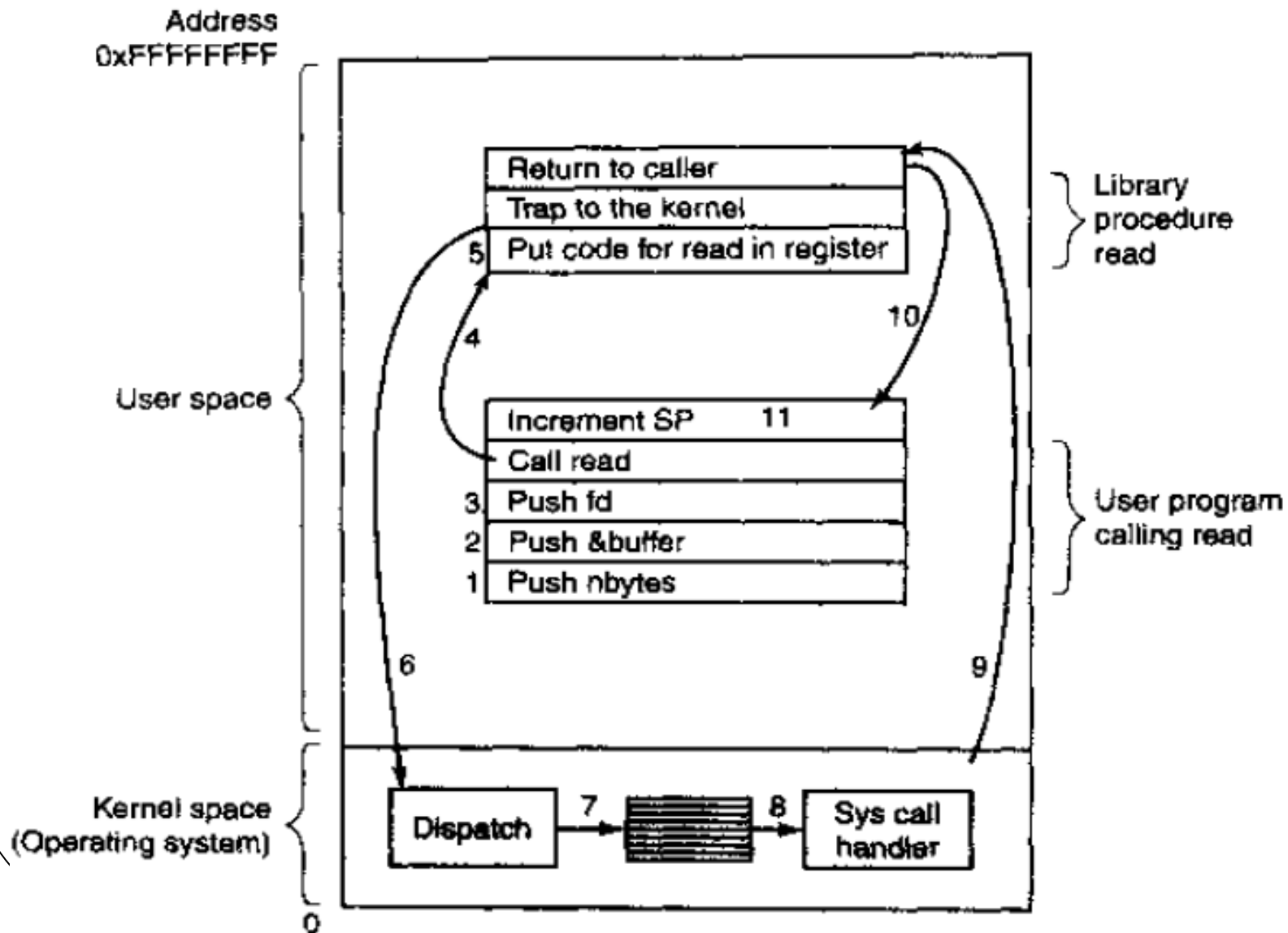
System Call Example

- Another set of system calls is helpful in debugging a program. Many systems provide system calls to dump memory. This provision is useful for debugging. A program trace lists each instruction as it is executed; it is provided by fewer systems. Even microprocessors provide a CPU mode known as **single step**, in which a trap is executed by the CPU after every instruction.

Passing of Parameters As A Table



Count=read(fd,buffer,nbytes); is a command in c language



System Programs

- System programs provide 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 operating system is defined by system programs, not the actual system calls.

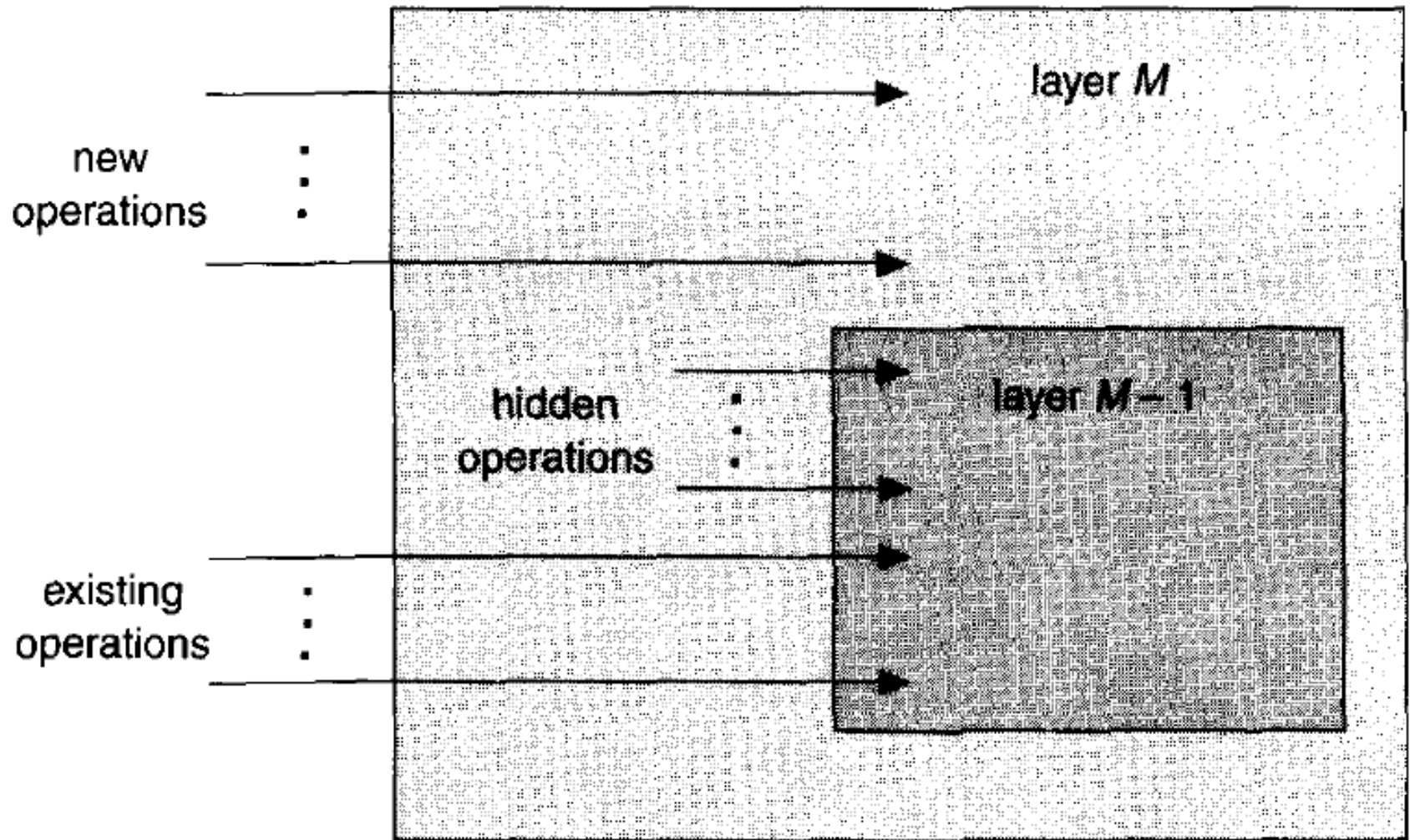
System Program Example

- Perhaps the most important system program for an operating system is the **command interpreter**, the main function of which is to get and execute the next user-specified command.
- For example, a command to **delete a file** may cause the command interpreter to jump to a section of its code that sets up the parameters and makes the appropriate system call.

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

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.
- An operating system can create the illusion that a process has its own processor with its own (virtual) memory.
- A virtual machine provides an interface *identical* to the underlying bare hardware. This solution can represent the additional features of a process, such as system calls and a file system.
- 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.

Virtual Machines (Cont.)

- The virtual-machine software can run in monitor mode, since it is the operating system.
- The virtual machine itself can execute in only user mode. Consequently, we must have a virtual user mode and a virtual monitor mode, both of which run in a physical user mode.

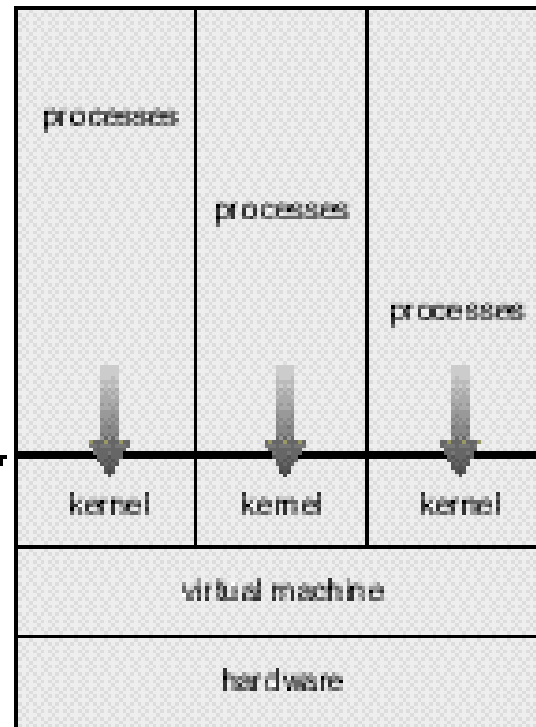
System Models

Non-virtual Machine



(a)

Virtual Machine



(b)

Advantages/Disadvantages of Virtual Machines

- 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.

System Design 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.

System Implementation

- Traditionally written in assembly language, operating systems can now be written in higher-level languages.
- 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.

System Generation (SYSGEN)

- Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site.
- SYSGEN program obtains information concerning the specific configuration of the hardware system.
- *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.

Metric Units

| Exp. | Explicit | Prefix | Exp. | Explicit | Prefix |
|------------|----------------------------|--------|-----------|-----------------------------------|--------|
| 10^{-3} | 0.001 | milli | 10^3 | 1,000 | Kilo |
| 10^{-6} | 0.000001 | micro | 10^6 | 1,000,000 | Mega |
| 10^{-9} | 0.000000001 | nano | 10^9 | 1,000,000,000 | Giga |
| 10^{-12} | 0.000000000001 | pico | 10^{12} | 1,000,000,000,000 | Tera |
| 10^{-15} | 0.000000000000001 | femto | 10^{15} | 1,000,000,000,000,000 | Peta |
| 10^{-18} | 0.000000000000000001 | atto | 10^{18} | 1,000,000,000,000,000,000 | Exa |
| 10^{-21} | 0.000000000000000000001 | zepto | 10^{21} | 1,000,000,000,000,000,000,000 | Zetta |
| 10^{-24} | 0.000000000000000000000001 | yocto | 10^{24} | 1,000,000,000,000,000,000,000,000 | Yotta |