Operating System

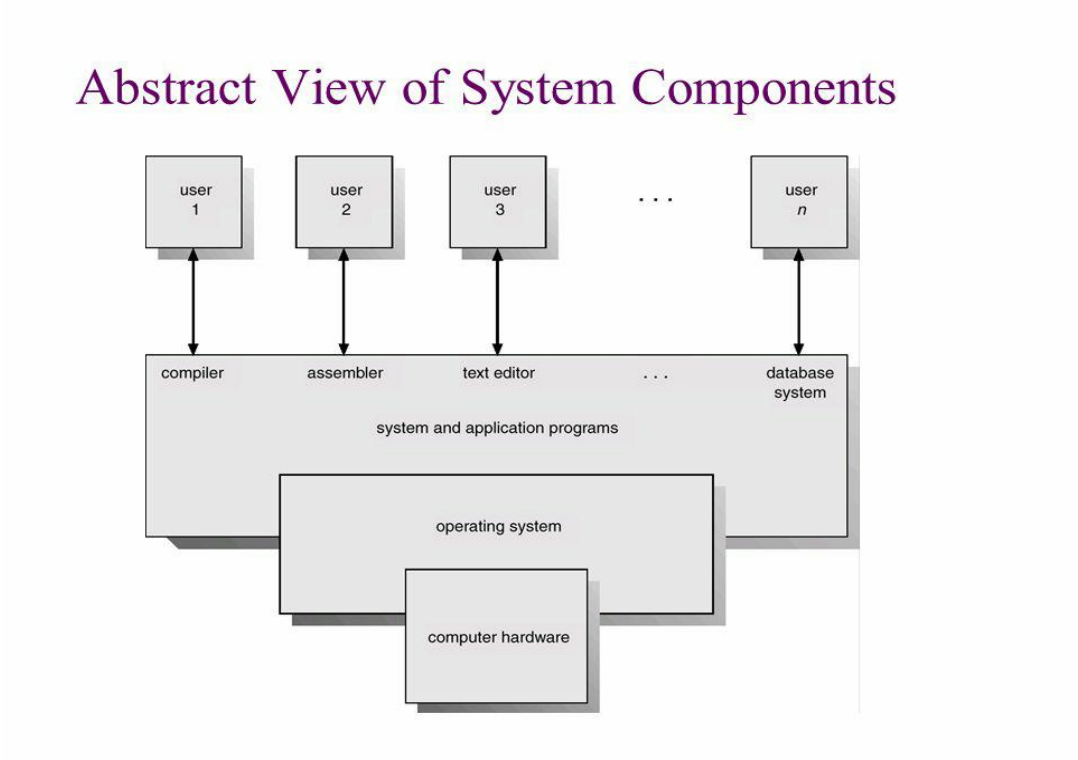
Unit - 1

A program that acts as an intermediary between the user of the computer and its hardware is an operating system.

- An operating system is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware.
- Mainframe operating systems are designed primarily to optimize utilization of hardware.
- Personal computer (PC) operating systems support complex games, business applications, and everything in between.
- ***Thus, some operating systems are designed to be convenient, others to be efficient, and others some combination of the two.***

Operating system's role in the overall computer system

- A computer system can be divided into four components:
  - The hardware
  - The operating system,
  - The application programs
  - The users.
- The hardware- the CPU, Input-Output devices and memory provides the basic computing resources for the system.
- The application programs such as word processors/ spreadsheets/ compilers, and Web browsers-define the ways in which these resources are used to solve users computing problems.
- The operating system controls the hardware and coordinates its use among the various application programs for the various users.



1. **User View**
    - The user's view of the computer varies according to the interface being used.
    - In case of **PC**, consisting of a monitor/ keyboard/ mouse, and system unit, the operating system is designed mostly for **ease of use** with some attention paid to performance and none paid to resource utilization.
    - In case of **terminal** connected to mainframe or a minicomputer, (users are accessing the same computer through other terminals) the operating system is designed to **maximize resource utilization**.
    - In case of **workstations connected to networks of other workstations and servers**, the operating system is designed to **compromise between individual usability and resource utilization.**

Some computers have little or no user view. For example, embedded computers in home devices and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems are designed primarily to run without user intervention.

2. System View

- From the computer's point of view, the operating system is the program most intimately involved with the hardware.
- Hence an operating system can be viewed as a resource allocator.
- A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/0 devices, and so on. **The operating system acts as the manager of these resources**.
- Facing numerous and possibly conflicting requests for resources, the ***operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly***.
- An operating system is a control program.
- A control program manages the execution of user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of I/O devices.

STORAGE DEFINITIONS AND NOTATION

*A **bit** is the basic unit of computer storage. It can contain one of two values, zero and one.*

*A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.*

*A **word** is generally made up of one or more bytes. For example, a computer may have instructions to move 64-bit (8-byte) words. A kilobyte, or KB, is 1,024 byte*

*A **megabyte**, or MB, is 1,0242 bytes*

*A **gigabyte**, or GB, is 1,024^3 bytes*

Defining Operating Systems

- The **fundamental goal of computer** systems is to execute user programs and to make solving user problems easier.
- Toward this goal, **computer hardware is constructed**.
- Since bare hardware alone is not particularly easy to use, **application programs are developed**. These programs require certain common operations, such as those controlling the I/O devices.
- The common functions of controlling and allocating resources are then brought together into one piece of software: **the operating system.**

Definition: **The operating system is the one program running at all times on the computer-usually called the kernel.**

(Along with the kernel, there are two other types of programs**: System programs** which are associated with the operating system but are not part of the kernel, and **application programs**, which include all programs not associated with the operation of the system.)

Operating System Architecture

1. Single Processor Systems

- On a single processor system, there is one main CPU capable of executing a general purpose instruction set, including instructions from user processes.
- Special purpose processors come in the form of device specific processors such as disk, keyboard and graphics controller.
- All these special purpose processors run a limited instruction set and do not run user processes.

- Sometimes operating system monitors them and sends them information about their next task and monitors their status.
- In other systems or circumstances, special purpose processors are low level components built into the hardware. Operating systems cannot communicate with these processors.

2. Multiprocessor Systems

- They are also known as parallel systems or tightly coupled systems.
- They have 2 or more processors in close communication, sharing the computer bus and sometimes clock memory and peripheral devices.
- Three main advantages of multiprocessor systems

i. **Increased throughput**: Increasing the number of processors – more work done in less time.

When multiple processors are used, certain amount of overhead is incurred.

Overhead + contention for shared resources = Low expected gain

ii. **Economy of scale:** They cost less than equivalent multiple single processor systems because they share mass storage and power supplies.

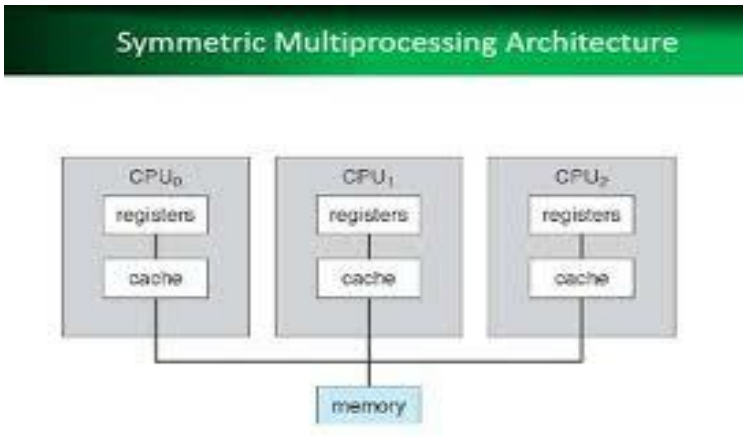
iii. **Increased reliability:** If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

Graceful degradation: The ability to continue providing service proportional to the level of surviving hardware.

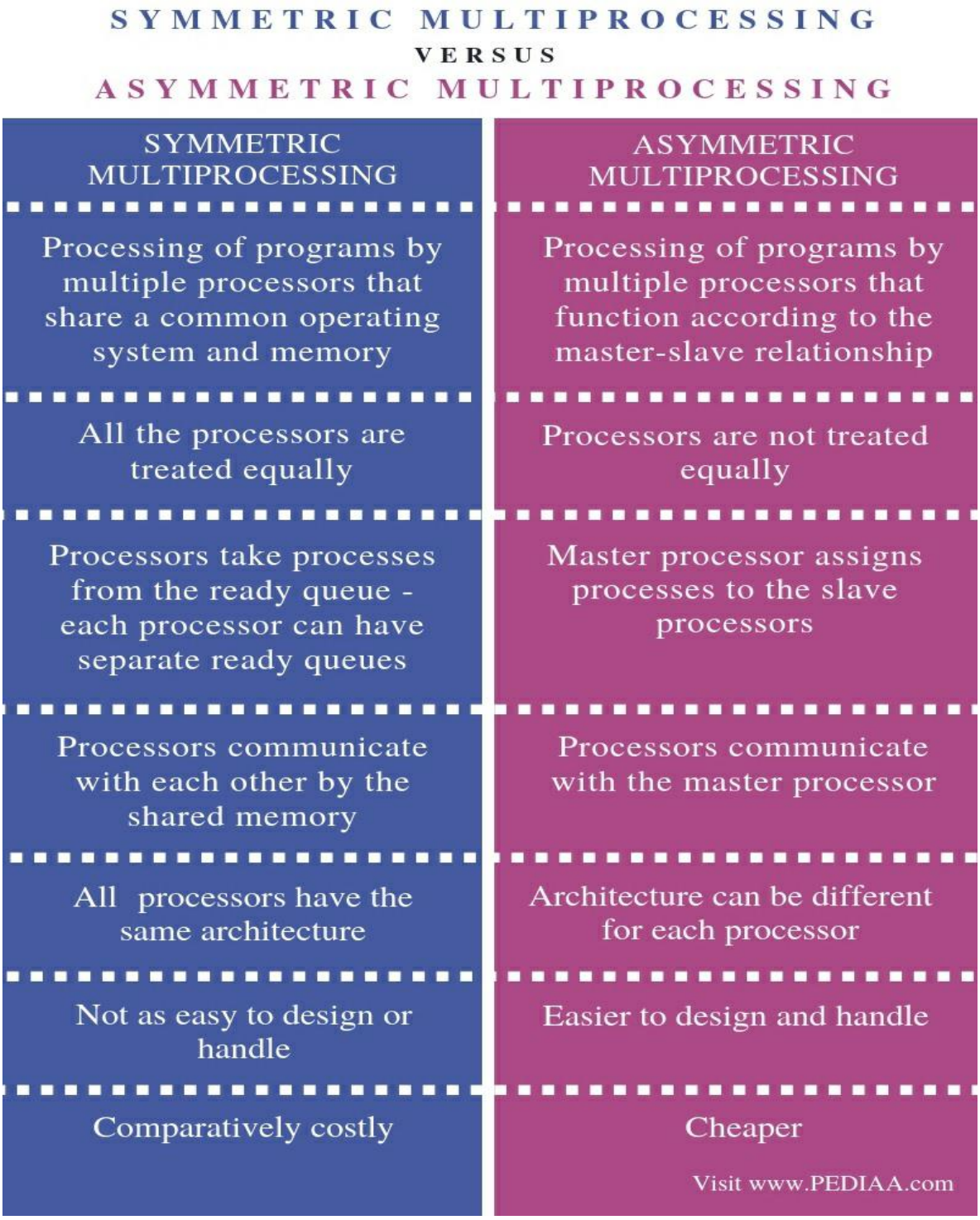
Fault tolerant: System are called so when they can suffer a failure of any single component and still continue operation.

- Multiprocessor system are of two types

- **Asymmetric multiprocessing**: Each processor is assigned a specific task. Master processor controls the system, other processors either look to the master for instruction or have predefined tasks. Master processor schedules and allocates work to slave processors.
- **Symmetric multiprocessing(SMP)** : Each processor performs all tasks within the operating system. Here all processors are peers, no master slave relationship.

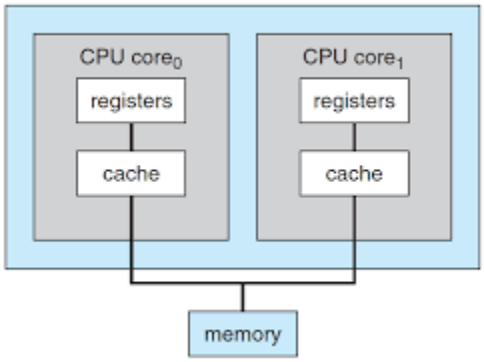


- Each processors has its own set of registers and also private or local cache.
- Example of SMP system is Solaris, a commercial version of UNIX designed by Sun Microsystems.
- The benefit of this model is that many processes can run simultaneously -N processes can run if there are N CPUs-without causing a significant deterioration of performance.
- Since the CPUs are separate, one may be sitting idle while another is overloaded, resulting in inefficiencies. These inefficiencies can be avoided if the processors share certain data structures.



- Virtually all modern operating systems-including Windows, Windows XP, Mac OS X, and Linux provide support for SMP.

- Multiprocessing adds CPUs to increase computing power. If the CPU has an integrated memory controller, then adding CPUs can also increase the amount of memory addressable in the system.
- **UMA (Uniform Memory Access)** is defined as the situation in which access to any RAM from any CPU takes the same amount of time.
- In **NUMA (Non-Uniform Memory Access)** some parts of memory may take longer to access than other parts, creating a performance penalty.
- **Dual core design**



- In this design, each core has its own register set as well as its own local cache; other designs might use a shared cache or a combination of local and caches.
- Blade Servers are a recent development in which multiple processor boards, I/0 boards, and networking boards are placed in the same chassis.
- The difference between these and traditional multiprocessor systems is that each blade-processor board boots independently and runs its own operating system.

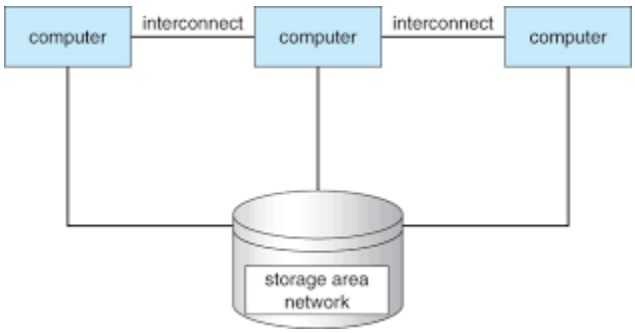
3. <u>Clustered Systems</u>

- It is a type of multiple-CPU system.

- Clustered systems gather together multiple CPUs to accomplish computational work.
- Clustered computers share storage and are closely linked through a local area network (LAN) or a faster interconnect, such as InfiniBand.
- Clustering is usually used to provide high availability service; that is, service will continue even if one or more systems in the cluster fail.
- High availability is generally obtained by adding a level of redundancy in the system. A layer of cluster software runs on the cluster nodes. Each node can monitor one or more of the others (over the LAN). If the monitored machine fails, the monitoring machine can take ownership of its storage and restart the applications that were running on the failed machine.
- Clustering can be structured asymmetrically or symmetrically. In **asymmetrically**, one machine is in hot-standby mode while the other is running the applications. The hot-standby host machine does nothing but monitor the active server. If that server fails, the hot-standby host becomes the active server. In **symmetric mode**, two or more hosts are running applications and are monitoring each other. This mode is obviously more efficient, as it uses all of the available hardware. It does require that more than one application be available to run.
- Clusters may also be used to provide high performance computing environments.
- **Advantage of the cluster** can be **taken by using** a technique known as **Parallelization,** which consists of dividing a program into separate components that run in parallel on individual computers in the cluster.
- Other forms of clusters include parallel clusters and clustering over a wide-area network (WAN).

- **Parallel clusters** allow multiple hosts to access the same data on the shared storage.
- To provide this shared access to data, the system must also supply access control and locking to ensure that no conflicting operations occur. This function, commonly known as **distributed lock manager (DLM)** is included in some cluster technology.

General structure of a clustered system



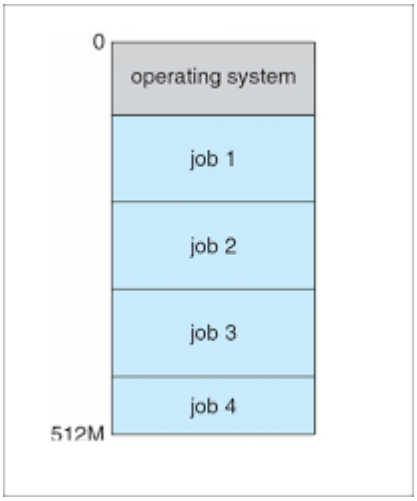
Operating System Structure

- One of the most important aspects of operating systems is the ability to multiprogram. A single program cannot, in general keep either the CPU or the I/O devices busy all times: Single users frequently have multiple programs running.
- Multiprogramming **increases CPU utilization** by organizing jobs (code and data) so that CPU has always has one to execute.
- The idea is as follows: The operating system keeps several jobs in memory simultaneously. Since, in general main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the **job**

pool. This pool consists of all processes residing on disk awaiting allocation of main memory.

- The set of jobs in memory can be a subset of the jobs kept in the job pool. The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete.
- In a **non-multiprogrammed system,** the CPU would sit idle. In **multiprogrammed system**, the operating system simply switches to, and executes, another job
- Time sharing or multitasking is a logical extension of multiprogramming.
- In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.
- Time sharing requires an interactive or hands-on computer system, which provides direct communication between the user and the system. The user gives instructions to the operating system or to a program directly, using an input device such as a keyboard or a mouse, and waits for immediate results on an output device.
- A program loaded into memory and executing is called **process.** When a process executes, it typically executes for only a short time it either finishes or needs to perform I/0.
- Time sharing and multiprogramming require that several jobs be kept simultaneously in memory. If several jobs are ready to be brought into memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is **job scheduling.**

- If several jobs are ready to run at the same time, the system must choose among them. Making this decision **CPU scheduling.**
- In a time-sharing system, the operating system must ensure reasonable response time, which is sometimes accomplished through **swapping**, where processes are swapped in and out of main memory to the disk.
- Time-sharing systems must also provide a file system.



Operating system operations

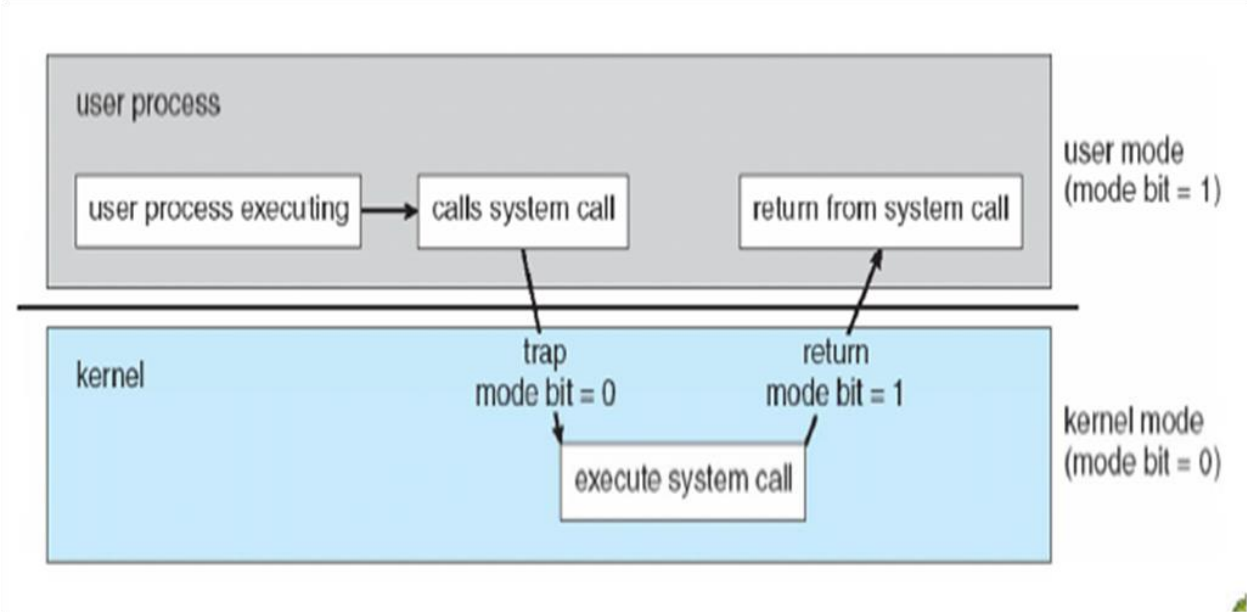
- Modern operating systems are interrupt driven. If there are no processes to execute, no I/0 devices to service, and no users to whom to respond, an operating system will sit quietly waiting for something to happen
- Events are almost always signaled by the occurrence of an interrupt or a trap.
- A trap or an exception is a software generated interrupt caused by an error (for division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed.

- For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt.
- A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

Dual-Mode Operation

- Two separate modes of operation are user mode and kernel mode (Supervisor mode, system mode or privileged mode)
- A bit, called the mode bit is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- With the mode bit, we are able to distinguish between a task that is executed on behalf of the operating system and the one that is executed on behalf of the user.
- When the computer system is executing on behalf of a user application, the system is in user mode. However, when a user application requests a service from the operating system (via a system call), it must transit from user to kernel mode to fulfill the request.
- At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the state of the mode bit to 0). Thus, whenever the operating system gains control of the computer, it

is in kernel mode. The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program.



- Protection of operating system can be accomplished by some machine instructions that may cause harm as privileged instructions. Hardware allows these instructions to be executed only in kernel mode.
- If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.
- Examples of privileged instructions are I/0 control, timer management and interrupt management.
- **Life cycle of instruction execution in computer system**

Initial control resides in the operating system, where instructions are executed in kernel mode. When control is given to a user application, the mode is set to user mode. Eventually, control is switched back to the operating system via an interrupt, a trap, or a system call.

- System calls provide the means for a user program to ask the operating system to perform tasks that are reserved for the operating system on the user's program behalf.

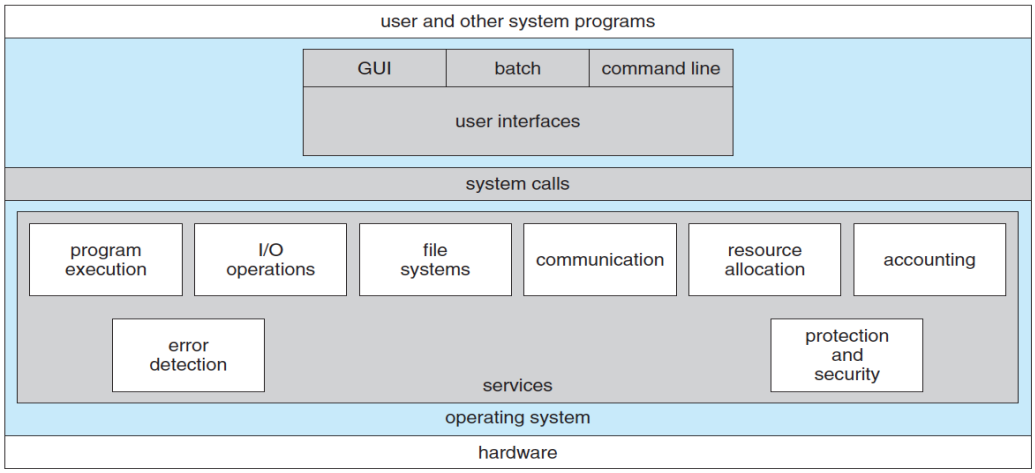
When a system call is executed, it is treated by the hardware as a software interrupt. Control passes through the interrupt vector to a service routine in the operating system and the mode bit is set to kernel mode.

- The kernel examines the interrupting instruction to determine what system call has occurred; a parameter indicates what type of service the user program is requesting.
- The kernel verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call

Timer

- It should be ensured that operating system must have control over CPU.
- Timer is used for this purpose.
- A timer can be set to interrupt the computer after a specified period.
- The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second).
- The operating system sets the counter. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs.
- If the timer interrupts, control transfers automatically to the operating system, which may treat the interrupt as fatal error or may give the program more time.

Operating System Services



Below are the set of operations that are helpful for the users.

- **1. User interface** : It can be of different forms

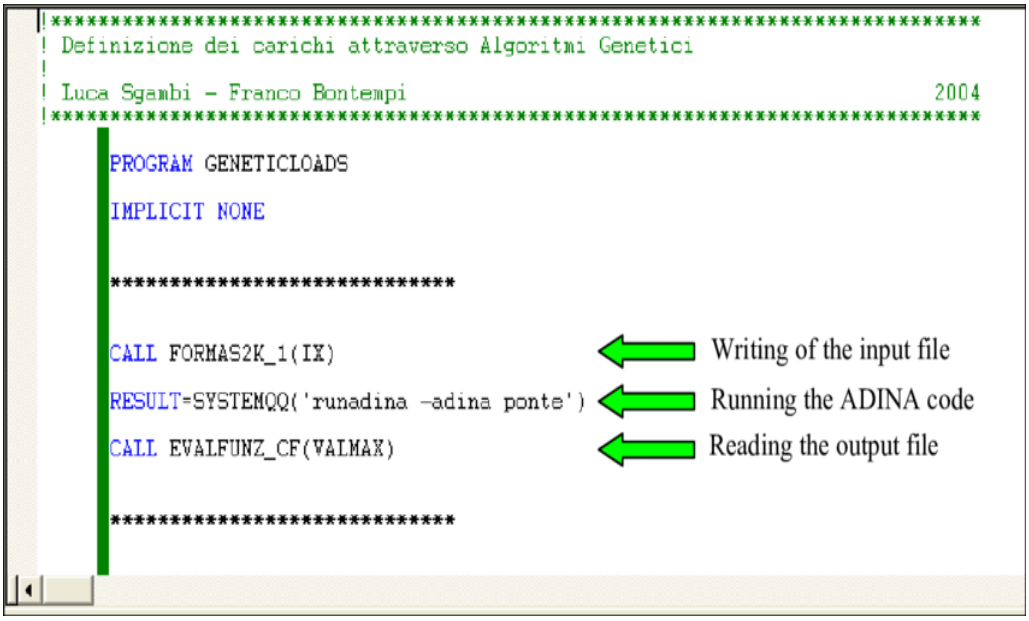
  a. **DTrace command line interface (CLI)**: It uses text commands and a method for entering them.

printf() Example

```
# dtrace -n 'syscall::exece:return { printf("%6d %s\n", pid, execname); }'
dtrace: description 'syscall::exece:return ' matched 1 probe
CPU     ID                    FUNCTION:NAME
  0  74415                      exece:return   4301 sh
  0  74415                      exece:return   4304 neqn
  0  74415                      exece:return   4305 nroff
  0  74415                      exece:return   4306 sh
  0  74415                      exece:return   4308 sh
[...]
```

- DTrace ships with a powerful printf(), to print formatted output.

b. **Batch interface**: Here commands and directives to control those commands are entered into files, and those files are executed.



c. **Graphical user interface (GUI)**: This interface is a window system with a pointing device to direct I/0, choose from menus, and make selections and a keyboard to enter text.

2. Program execution: The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).

3. I/O Operations: A running program may require I/0, which may involve a file or an I/0 device. For efficiency and protection, users usually cannot control I/0 devices directly. Therefore, the operating system must provide a means to do I/0.

4. File system manipulation: Programs need to read and write files and directories. They also need to create and delete them by name, search for a given file, and list file information. Finally, some programs include permissions management to allow or deny access to files or directories based on file ownership. Many operating systems provide a variety of file systems, sometimes to allow personal choice, and sometimes to provide specific features or performance characteristics.

5. Communications: Communications may be implemented via shared memory or through message passing, in which packets of information are moved between processes by the operating system.

6. Error Detection: The operating system needs to be constantly aware of possible errors. Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/0 devices (such as a parity error on tape, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too-great use of CPU time). For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.

Below are the set of operations that are for ensuring the efficient operation of the system itself .

1. Resource allocation: When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them. Some (such as CPU cycles, main memory, and file storage) may have special allocation code, whereas others (such as I/0 devices) may have much more general request and release code. For example, CPU-scheduling routines are used to determine how best the CPU is used.

2. Accounting: Track of which users use how much and what kinds of computer resources should be maintained. This record keeping may be used for accounting so that users can be billed or simply for accumulating usage statistics.

3. Protection and security: The owners of information stored in a multiuser or networked computer system may want to control use of that information. Protection involves ensuring that all access to system Resources is controlled. Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access to system resources.

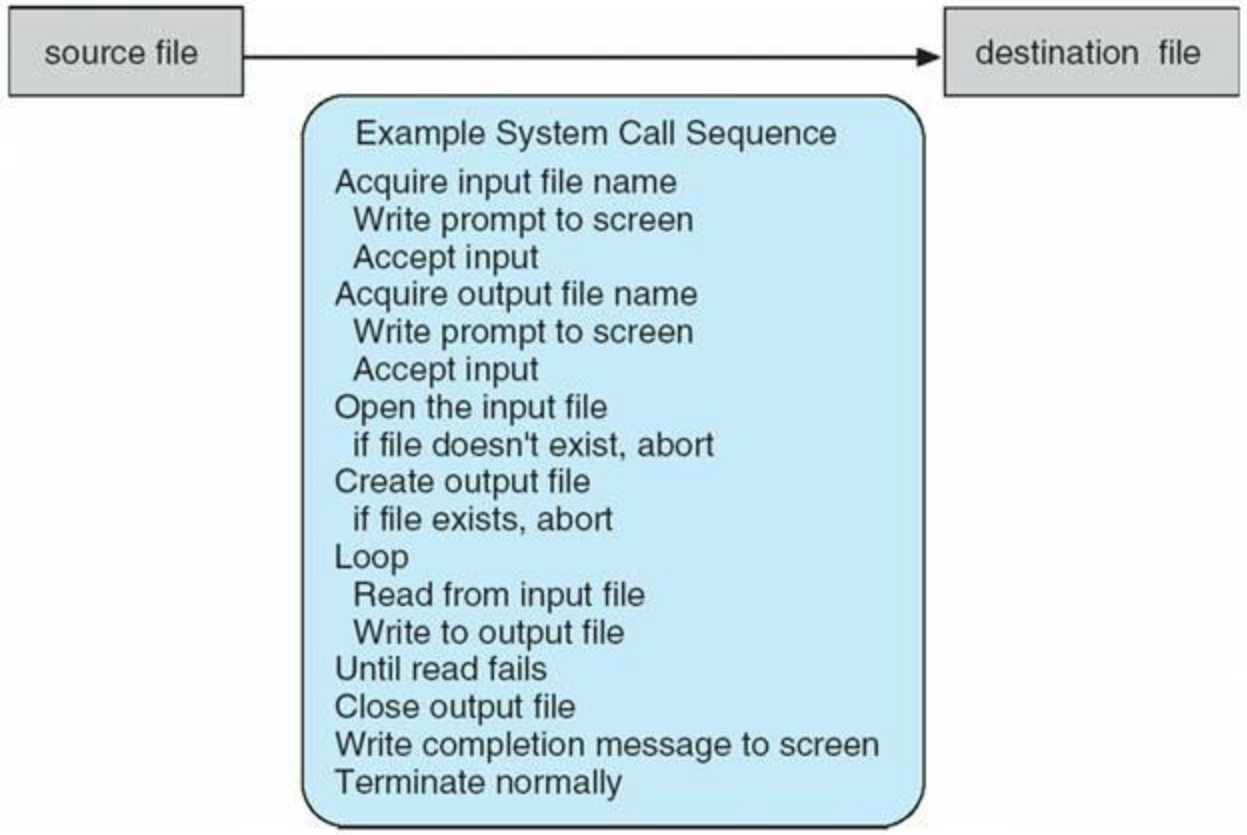
System Calls

- System calls provide an interface to the services made available by an operating system.
- These calls are generally available as routines written in C and C++.
- **Example:**

Writing a simple program to read data from one file and copy them to another file.

1. The first input that the program will need is the names of the two files: the input file and the output file.
    - These names can be specified in many ways, depending on the operating-system design.
    - **One approach** is for the program to ask the user for the names of the two files. In an interactive system, this approach will require a **sequence of system calls**, first to write a prompting message on the screen and then to read from the keyboard the characters that define the two files.
    - On mouse-based and icon-based systems, a menu of file names is usually displayed in a window. The user can then use the mouse to select the source name, and a window can be opened for the destination name to be specified. This sequence requires many I/0 system calls.
2. Once the two file names are obtained, the program must open the input file and create the output file. Each of these operations requires **another system call.**
3. When the program tries to open the input file, it may find that there is no file of that name or that the file is protected against access. In these cases, the program should print a message on the console **(another sequence of system calls**) and then terminate abnormally **(another system call)**.
4. If the input file exists, then we must create a new output file.
5. We may find that there is already an output file with the same name. This situation may cause the program to abort **(a system call),** or we may delete the existing file **(another system call)** and create a new one **(another system call).**

5. Another option, in an interactive system, is to ask the user (via a **sequence of system calls** to output the prompting message and to read the response from the termin.al) whether to replace the existing file or to abort the program.
6. Now that both files are set up, we enter a loop that reads from the input file **(a system call)** and writes to the output file (**another system call).** Each read and write must return status information regarding various possible error conditions.
7. Finally, after the entire file is copied, the program may close both files **(another system call),** write a message to the console or window **(more system calls)**, and finally terminate normally **(the final system call).**

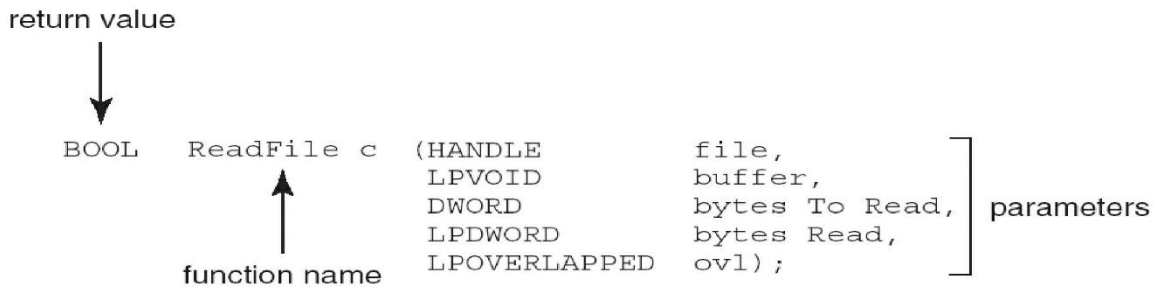


- Application developers design programs according to an **application programming interface (API).**
- An **API** specifies a set of functions that are available to an application programmer including the parameters that are passed to each function and the return values the programmer can expect.
- **Three of the most common APIs** available to application programmers are the Win32 API for Windows systems, the POSIX API for POSIX-based systems (which include virtually all versions of UNIX, Linux/ and Mac OS X), and the Java API for designing programs that run on the Java virtual machine.

- One **benefit of programming according to an API** concerns **program portability**: An application programmer designing a program using an API can expect the program to compile and run on any system that supports the same API.
- Actual system calls can often be more detailed and difficult to work with than the API available to an application programmer.

Example of Standard API

- Consider the ReadFile() function
- Win32 API—a function for reading from a file

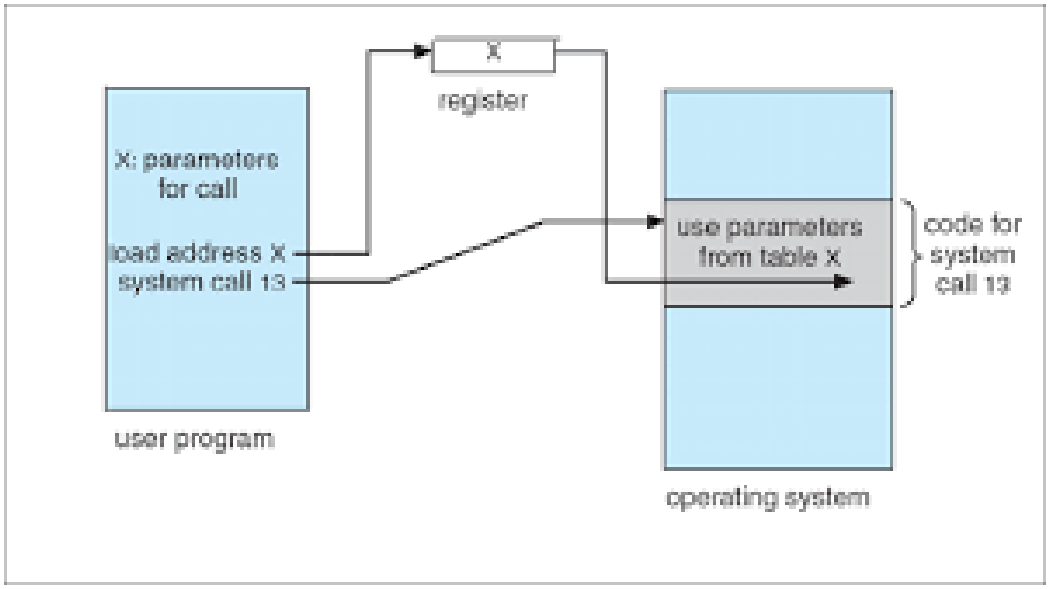


- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read
  - LPOVERLAPPED ovl—indicates if overlapped I/O is being used

2.14

- The run-time support system (a set of functions built into libraries included with a compiler) for most programming languages provides a system-call interface that serves as the link to system calls made available by the operating system.

- The system-call interface intercepts function calls in the API and invokes the necessary system calls within the operating system.
- Typically, a number is associated with each system call, and the system-call interface maintains a table indexed according to these numbers. The system call interface then invokes the intended system call in the operating-system kernel and returns the status of the system call and any return values.
- **Three general methods are used to pass parameters to the operating system**. The simplest approach is to **pass the parameters in registers**. In some cases, however, there may be more parameters than registers. In these cases, **the parameters are generally stored in a block, or table, in memory, and the address of the block is passed as a parameter in a register.**
- **Parameters also can be placed, or pushed, onto the stack by the program** and popped off the stack by the operating system.



Types of system calls

System calls can be grouped into six major categories:

1. Process control
2. File manipulation

3. Device manipulation

4. Information maintenance,

5. Communications

6. Protection

- 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
- 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

Figure 2.8 Types of system calls.

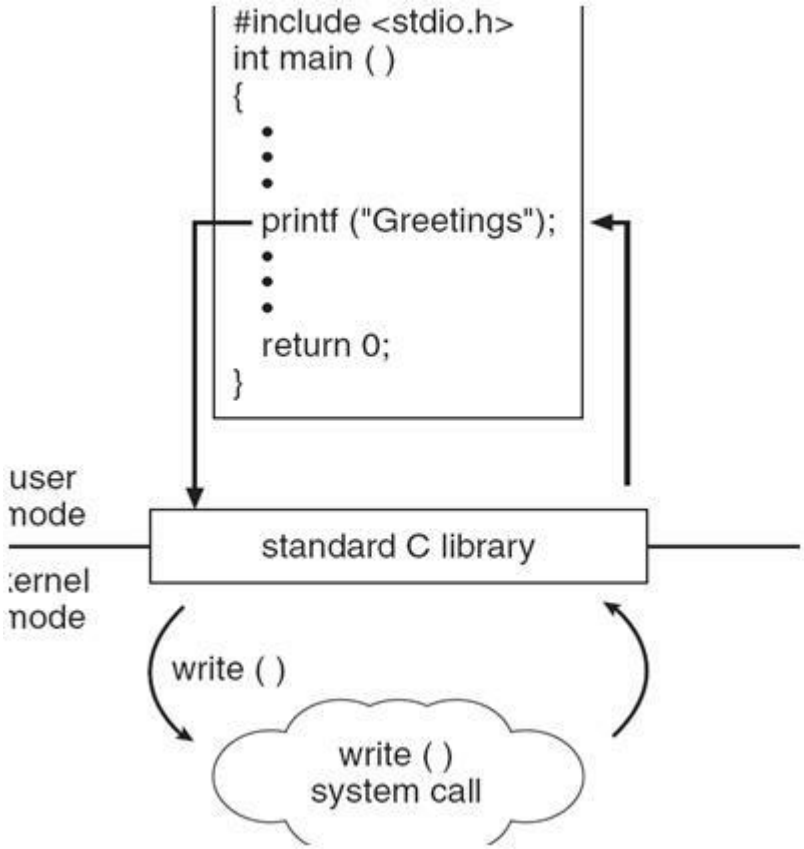
Process Control

- A running program needs to be able to **halt its execution either normally (end) or abnormally (abort)**. If a system call is made to terminate the currently running program **abnormally**, or if the program runs into a problem and causes an error trap, **a dump of memory is sometimes taken and an error message generated**. The dump is written to disk and may be examined by a **debugger - system program designed to aid the programmer in finding and correcting bugs**-to determine the cause of the problem.
- Under either normal or abnormal circumstances, the operating system must transfer control to the invoking command interpreter. The command interpreter then reads the next command.
- In an interactive system, the command interpreter simply continues with the next command.
- In a GUI system, a pop-up window might alert the user to the error and ask for guidance.
- In a batch system, the command interpreter usually terminates the entire job and continues with the next job.
- A control card is a batch-system concept. It is a command to manage the execution of a process.

	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()

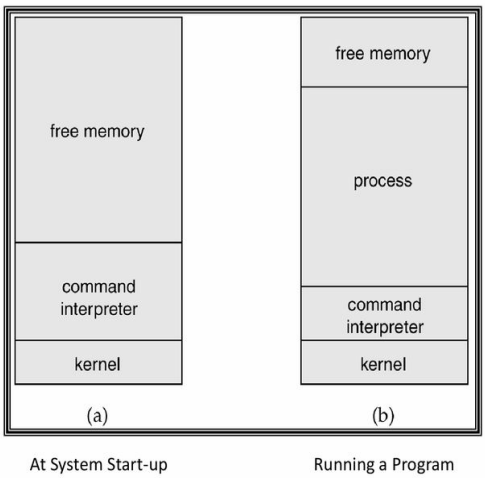
- If we create a new job or process, or perhaps even a set of jobs or processes, we should be able to **control its execution**. This control requires the ability to determine and reset the attributes of a job or process, including the job's priority, its maximum allowable execution time, and so on (get process attributes and set process attributes). We may also want to terminate a job or process that we created (terminate process) if we find that it is incorrect or is no longer needed.
- Having created new jobs or processes, we may need to wait for them to finish their execution. We may want to wait for a certain amount of time to pass (wait time); more probably, we will want to wait for a specific event to occur (wait event). The jobs or processes should then signal when that event has

occurred (signal event). Quite often, two or more processes may share data. To ensure the integrity of the data being shared, operating systems often provide system calls allowing a process to lock shared data, thus preventing another process from accessing the data while it is locked.



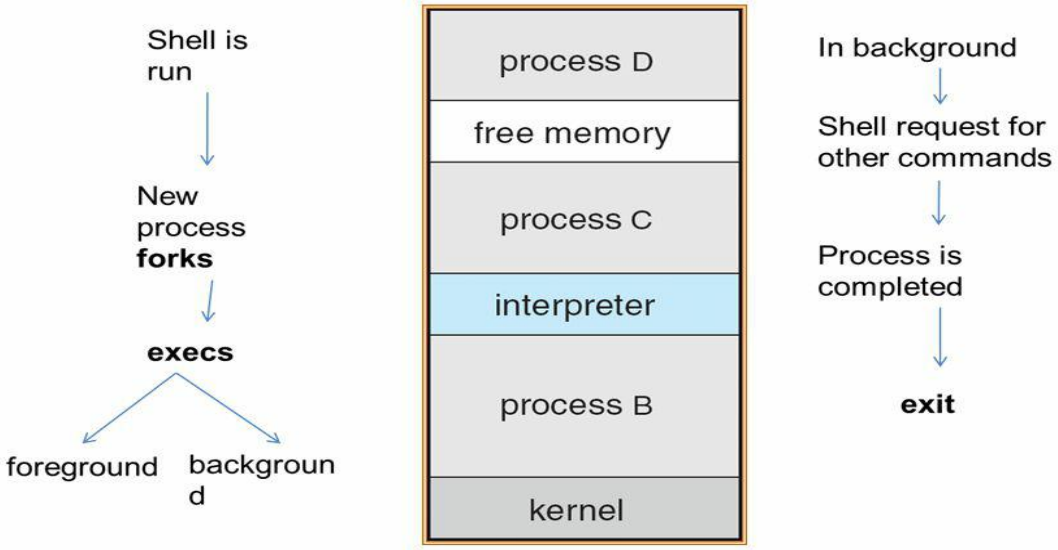
- Since MS-DOS is single-tasking, it uses a simple method to run a program and does not create a new process. It loads the program into memory, writing over most of itself to give the program as much memory as possible.
- Next, it sets the instruction pointer to the first instruction of the program. The program then runs, and either an error causes a trap, or the program executes a system call to terminate.

MS-DOS Execution



- FreeBSD (derived from Berkley UNIX) is an example of a multitasking system.

FreeBSD Running Multiple Programs



File management

- We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes. Once the file is created, we need to open it and to use it. We may also read, write, or reposition (rewinding or skipping to the end of the file, for example). Finally, we need to close the file, indicating that we are no longer using it.
- We may need these same sets of operations for directories if we have a directory structure for organizing files in the file system. In addition, for either files or directories, we need to be able to determine the values of various attributes and perhaps to reset them if necessary.

Device Management

- A process may need several resources to execute-main memory, disk drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user process. Otherwise, the process will have to wait until sufficient resources are available.
- Once the device has been requested (and allocated to us), we can read, write, and (possibly) reposition the device, just as we can with files.

Information maintenance

- Many system calls exist simply for the purpose of transferring information between the user program and the operating system. For example, most systems have a system call to return the current time and date.
- Other system calls may return information about the system, such as the number of current users, the version number of the operating system, the amount of free memory or disk space, and so on.
- Another set of system calls is helpful in debugging a program. Many systems provide system calls to dump memory.

- Many operating systems provide a time profile of a program to indicate the amount of time that the program executes at a particular location or set of locations.
- In addition, the operating system keeps information about all its processes, and system calls are used to access this information.

Communication

- There are two models of interprocess communication: the message passing model and the shared-memory model
- In the message passing model, the communicating process exchange messages with one another to transfer information. Messages can be exchanged between the processes either directly or indirectly through a common mailbox. Before communication can take place, a connection must be opened.
- Each computer in a network has a host name by which it is commonly known. A host also has a network identifier, such as an IP address. Similarly, each process has a process name, and this name is translated into an identifier by which the operating system can refer to the process.
- The get hosted and get processed system calls do this translation.
- The recipient process usually must give its permission for communication to take place with an accept connection call.
- In the shared memory model, processes use shared memory create and shared memory attach system calls to create and gain access to regions of memory owned by other processes.

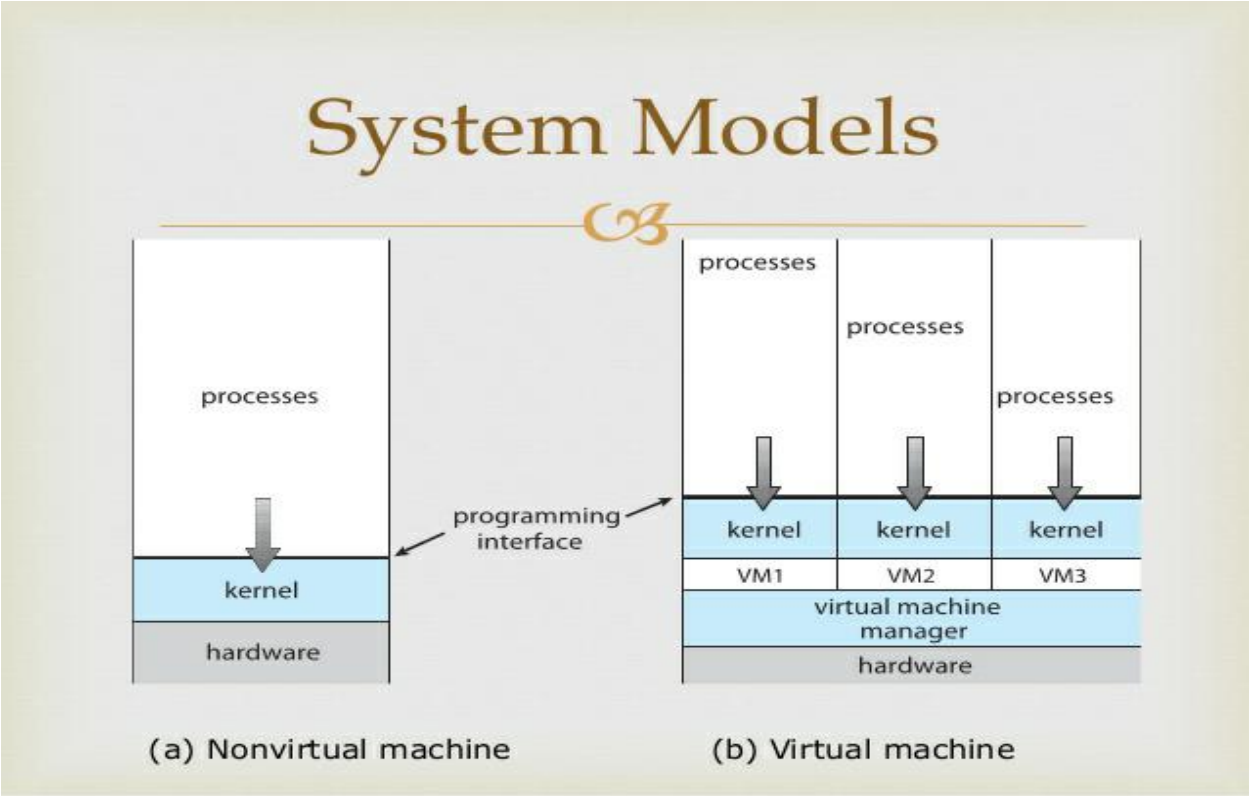
Protection

- Protection provides a mechanism for controlling access to the resources provided by a computer system.

- System calls providing protection include set permission and get permission, which manipulate the permission settings of resources such as files and disks. The allow user and deny user system calls specify whether particular users can-or cannot-be allowed access to certain resources.

Virtual Machines

- The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.
- The virtual machine provides an interface that is identical to the underlying bare hardware.



History

Benefits

1. Able to share the same hardware yet run several different execution environments (that is, different operating systems) concurrently.

2. The host system is protected from the virtual machines, just as the virtual machines are protected from each other.

3. Because each virtual machine is completely isolated from all other virtual machines, there are no protection problems.

4. There is no direct sharing of resources.

- Two approaches to provide sharing have been implemented. First, it is possible to share a file-system volume and thus to share files. Second, it is possible to define a network of virtual machines, each of which can send information over the virtual communications network.

5. Operating systems are large and complex programs, and it is difficult to be sure that a change in one part will not cause obscure bugs to appear in some other part. Therefore, the current system must be stopped and taken out of use while changes are made and tested. This period is commonly called system development time.

A virtual-machine system can eliminate much of this problem. System programmers are given their own virtual machine, and system development is done on the virtual machine instead of on a physical machine.

6. Multiple operating systems can be running on the developer's workstation concurrently.

7. A major advantage of virtual machines in production data-center use is System consolidation which involves taking two or more separate systems and running them in virtual machines on one system.

Simulation

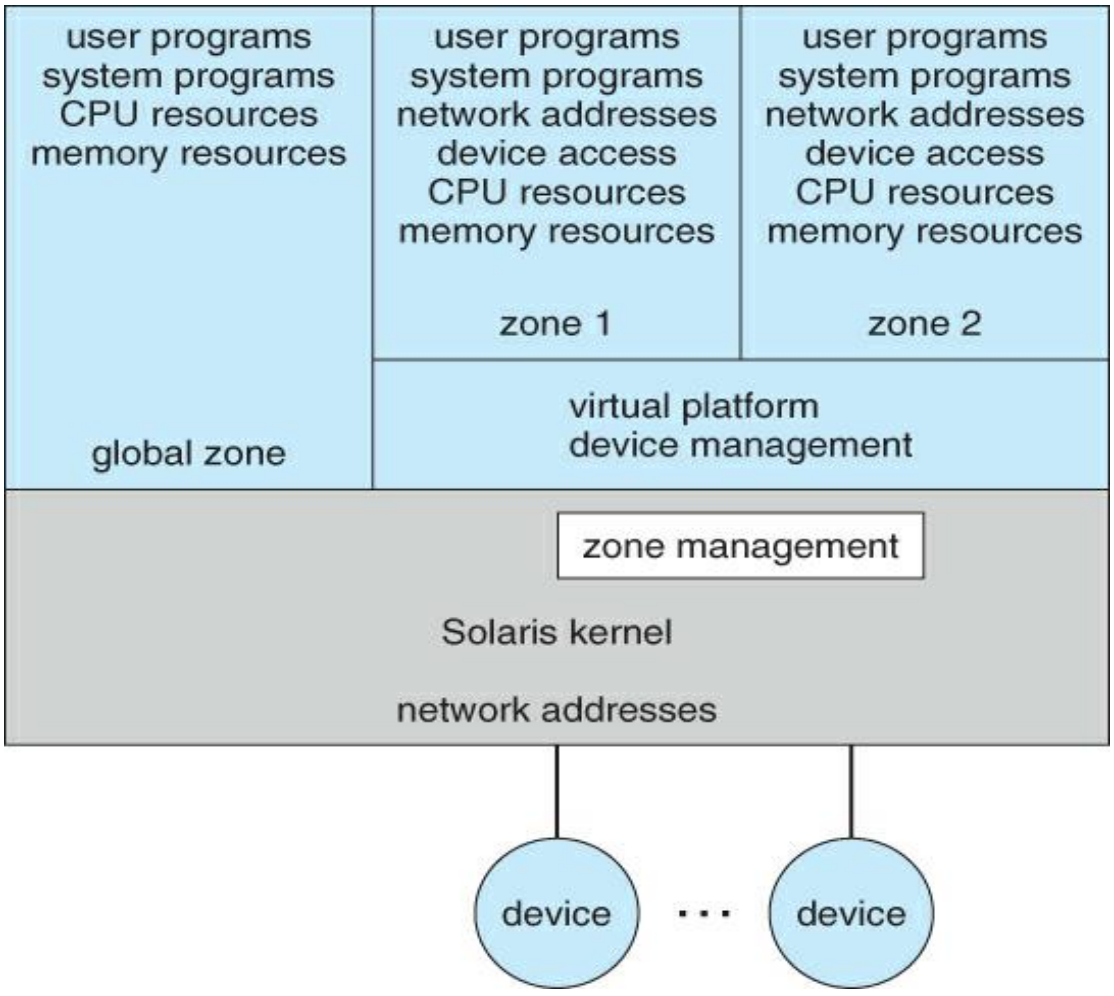
- It is a method in which the host system has one system architecture and the guest system was compiled for a different architecture.
- **For example,** suppose a company has replaced its outdated computer system with a new system but would like to continue to run certain important programs that were compiled for the old system. The programs could be run in an **emulator** that translates each of the outdated system's instructions into the native instruction set of the new system. Emulation can increase the life of programs and allow us to explore old architectures without having an actual old machine, but its major challenge is performance.

Para Virtualization

- Para virtualization presents the guest with a system that is similar but not identical to the guest's preferred system.
- The guest must be modified to run on the Para virtualized hardware.
- The **gain** for this extra work is more efficient use of resources and a smaller virtualization layer.
- Solaris 10 includes **containers** that create a virtual layer between the operating system and the applications.
- In this system, only one kernel is installed, and the hardware is not virtualized. Rather, the operating system and its devices are virtualized, providing

processes within a container with the impression that they are the only processes on the system.

- One or more containers can be created, and each can have its own applications, network stacks, network address and ports, user accounts, and so on



Implementation

- Although the virtual-machine concept is useful it is difficult to implement. Much work is required to provide an *exact* duplicate of the underlying

machine. Just as the physical machine has two modes, however, so must the virtual machine.

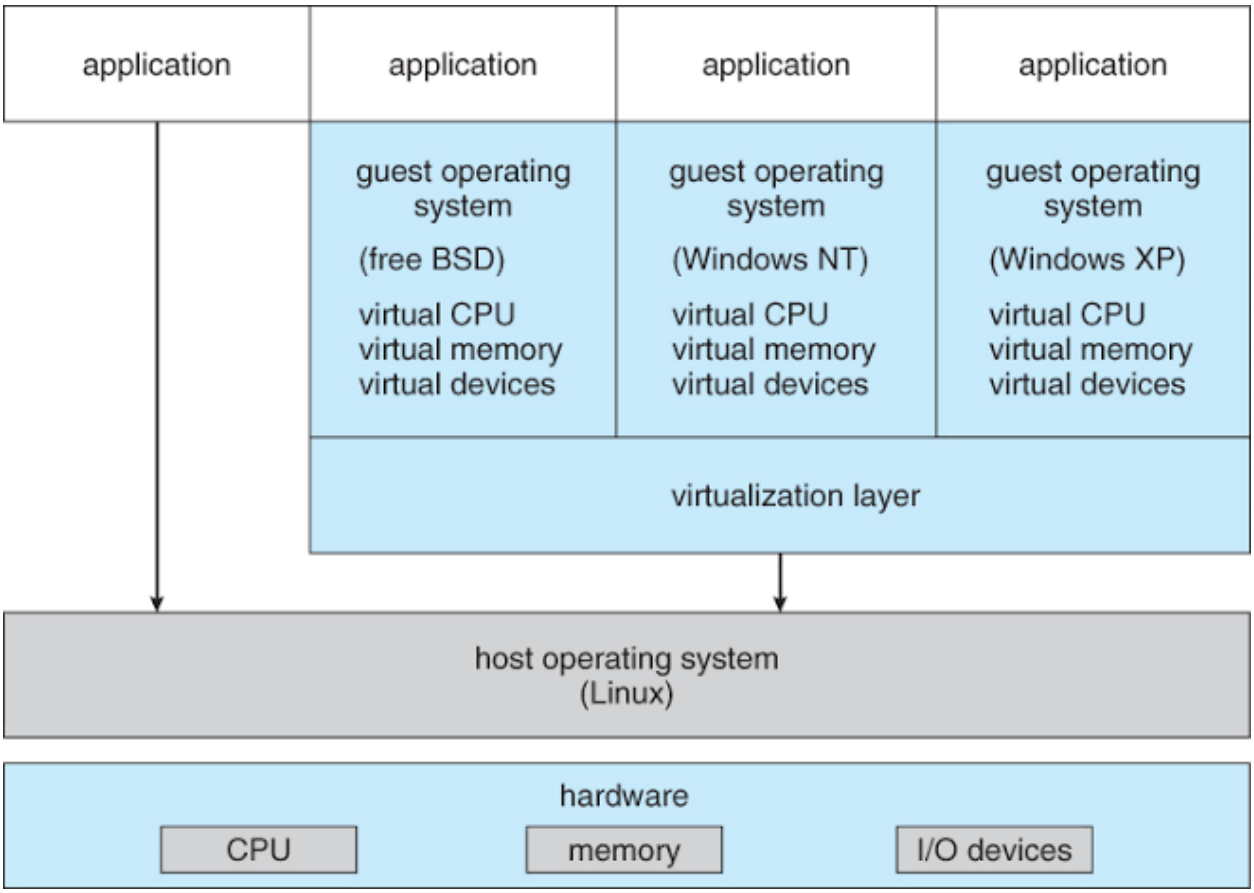
- Consequently, we must have a virtual user mode and a virtual kernel mode, both of which run in a physical user mode.
- Those actions that cause a transfer from user mode to kernel mode on a real machine (such as a system call or an attempt to execute a privileged instruction) must also cause a transfer from virtual user mode to virtual kernel mode on a virtual machine.
- **The major difference**, of course, is time. Whereas the real I/O might have taken 100 milliseconds, the virtual I/O might take less time (because it is spooled) or more time (because it is interpreted).
- Without some level of hardware support, virtualization would be impossible. The more hardware support available within a system, the more feature rich, stable, and well performing the virtual machines can be.

Examples

- VMware Workstation and the Java virtual machine

VMware

- VMware Workstation is a popular commercial application that abstracts Intel X86 and compatible hardware into isolated virtual machines.
- VMware Workstation runs as an application on a host operating system such as Windows or Linux and allows this host system to concurrently run several different guest operating systems as independent virtual machines.

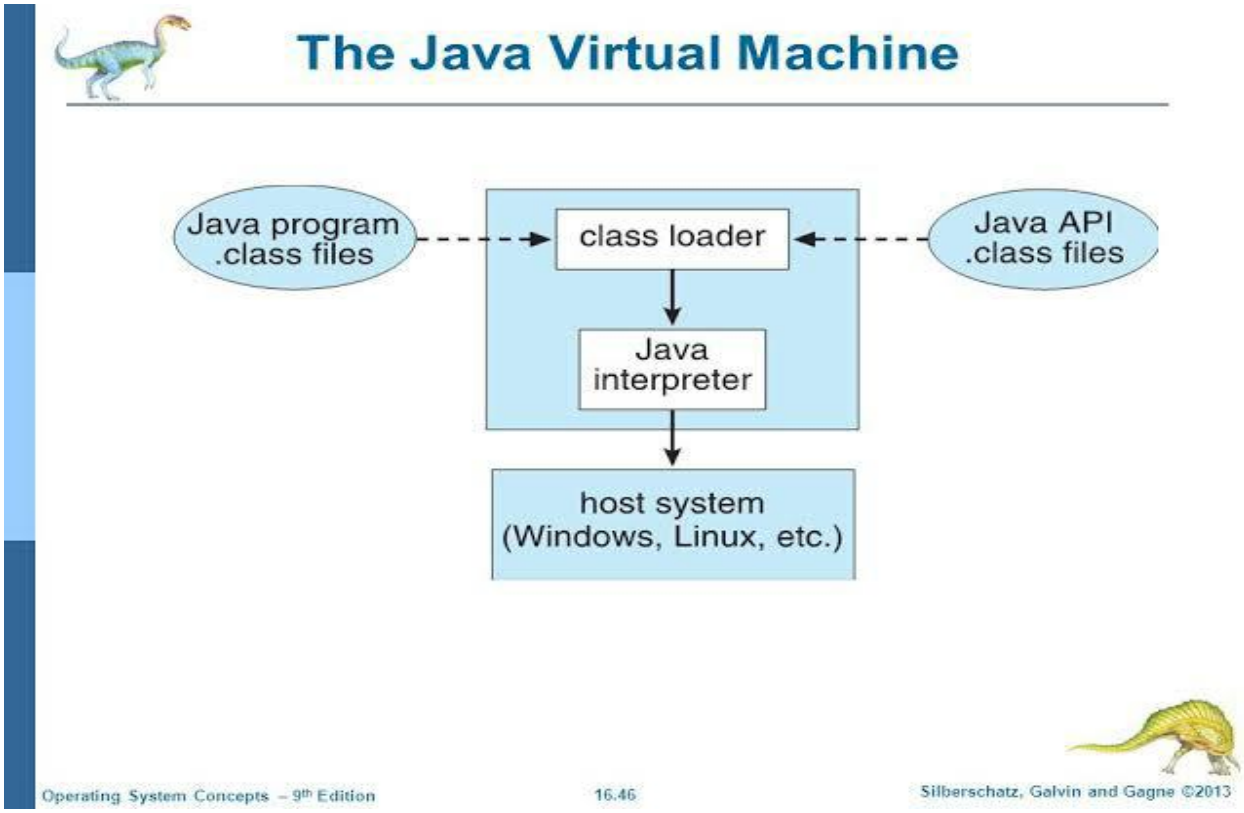


- Linux is running as the host operating system and FreeBSD, Windows NT, and Windows XP are running as guest operating systems.
- The virtualization layer is the heart of VMware, as it abstracts the physical hardware into isolated virtual machines running as guest operating systems.
- Each virtual machine has its own virtual CPU, memory, disk drives, network interfaces, and so forth.

The Java Virtual Machine

- Java is a popular object-oriented programming language introduced by Sun Microsystems in 1995.
- Java objects are specified with the class construct; a Java program consists of one or more classes. For each Java class, the compiler produces an

architecture-neutral bytecode output (.class) file that will run on any implementation of the JVM.



- The JVM is a specification for an abstract computer. It consists of a class loader and a Java interpreter that executes the architecture-neutral bytecodes.

- The class loader loads the compiled .class files from both the Java program and the Java API for execution by the Java interpreter.

- It also ensures that the bytecode does not perform pointer arithmetic, which could provide illegal memory access.

- If the class passes verification, it is run by the Java interpreter.

- The JVM also automatically manages memory by performing garbage collection -the practice of reclaiming memory from objects no longer in use and returning it to the system.

- The JVM may be implemented in software on top of a host operating system, such as Windows, Linux, or Mac OS X, or as part of a Web browser.

- If the JVM is implemented in. software, the Java interpreter interprets the bytecode operations one at a time.

- A faster software technique is to use a just-in-time (JIT) compiler.

- Here, the first time a Java method is invoked, the bytecodes for the method are turned into native machine language for the host system. These operations are then cached so that subsequent invocations of a method are performed using the native machine instructions and the bytecode operations need not be interpreted all over again.