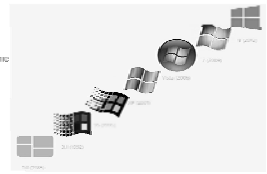
OPERATING SYSTEM

- The Slide does not contain all the information and cannot be treated as a study material for Operating System. Please refer the text book for exams.

Chapter 1: Introduction

- What Operating Systems Do
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose System

What are the different Operating systems you aware of ?



What is an Operating System?

- User-centric definition
  - A program that acts as an intermediary between a user of a computer and the computer hardware
  - Defines an interface for the user to use services provided by the system
  - Provides a "view" of the system to the user
- System-centric definition
  - Resource allocator – manages and allocates resources
  - Control program – controls the execution of user programs and operations of I/O devices

What is an Operating System?

- **Users**
  - People, machines, other computers
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
  - **Use the computer hardware in an efficient manner**.

Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
- No universally accepted definition
- *The one program running at all times on the computer* is the OS **kernel.** Anything else is either
  - a system program
    or
  - an application program

How it all started??

Batch Systems: Multiple jobs, but only one job in memory at one time and executed (till completion) before the next one starts

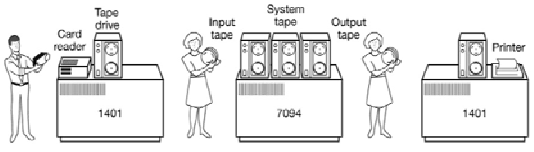
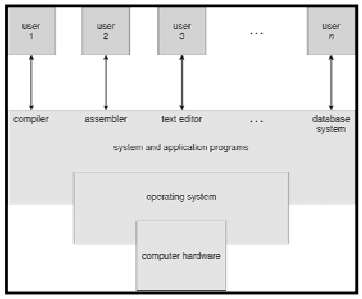


Figure 1-2. An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

Four Components of a Computer System



What OS Do??

- **User's View**

- PC
- Mainframe / Mini computer
- Workstation / Server
- Handheld devices/Laptops
- Embedded computers

- Ease of use vs Resource Utilization

What OS Do??

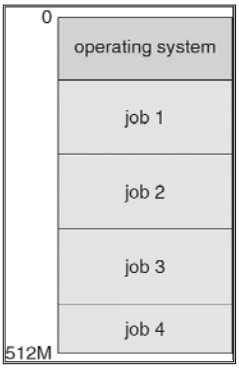
- **System's View**

- Resource Allocator
- Control Program

Operating System Structure

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

Memory Layout for Multiprogrammed System



Operating System Structure

- **Timesharing (multitasking):** a logical extension
  - CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - One job selected from job pool to main memory is **job scheduling**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out
  - **Virtual memory** allows execution of processes not completely in memory

Operating System Structure

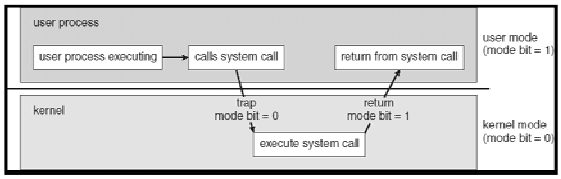
- **Time sharing and multiprogramming require several jobs be kept simultaneously in memory**
  - Job scheduling
  - Memory management
  - CPU scheduling
- **Time Sharing system must ensure that reasonable response time-typically less that one second**
  - Swapping
  - Virtual memory
- **Time shared system must also provide**
  - Disk management
  - Protecting resources from inappropriate use

Operating System Operations

- **Modern Operating system are interrupt driven**
  - Events are 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 or a specific request from user program that a operating system service be performed
  - Interrupt service routine will deal with the interrupt
- **OS must ensure that an incorrect program cannot cause other programs to execute incorrectly**

Operating System Operations

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode (supervisor mode or system mode or privileged mode)**
  - **A bit, called the mode bit** provided by hardware to indicate current mode: kernel mode (0) or user (1)



Operating System Operations

- **System Calls** -System call is the method when a user application requests a service from the OS
- **Various instructions (such as I/O, timer management, interrupt management and halt) are privileged instructions and can be executed only in kernel mode**

Operating System Operations

- **Timer**
  - **A timer prevents infinite loop**
  - **A simple technique to implement timer is to initialize a counter with the amount of time that a program is allowed to run**

Resources Managed by OS

- Physical
  - CPU, Memory, Disk, I/O Devices like keyboard, monitor, printer
- Logical
  - Process, File, …

Hence we have

1. Process management
2. Memory management
3. File management
4. I/O management

Process Management

- **What is a program?**
- Program is a *passive entity*, (contents of a file stored on a disk)
- What is a process?
- process is an *active entity*, (A process is a program in execution)
- Single thread vs Multi thread process (program counter)

Process Management Activities

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Scheduling processes and threads on the CPUs
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- Main memory is a repository of quickly accessible data shared by the CPU and I/O device
- Instructions or data must be in memory for the CPU to execute or process them

Memory Management

- **Memory management activities**
  - Keep track of which parts of memory are currently being used and by whom
  - Deciding which processes and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- File-System management
- Disk management
- Caching
- I/O Systems

File Management

- What is a file??
  - Collection of related information defined by the creater
  - Logical storage unit – abstracts from the physical properties of its storage devices
  - Files represent programs and data

File Management

- **File management activities**
  - Creating and deleting files
  - Creating and deleting directories to organize files
  - Supporting primitives for manipulating files and directories
  - Mapping files onto secondary storage
  - Backing up files on stable(nonvolatile) storage media

Disk Management

- What are the different kinds of storage media?
  - Magnetic Disk
  - Optical Disk
  - Magnetic tape

Disk Management

- What are the characteristics of storage media?
  - Access speed
  - Capacity
  - Data transfer rate
  - Access method (sequential/random)

Disk Management

- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- **Disk management Activities**
  - Free space management
  - Storage allocation
  - Disk Scheduling
- Tertiary storage CD, DVD
- WORM vs RW
- May push the task to Application Programs

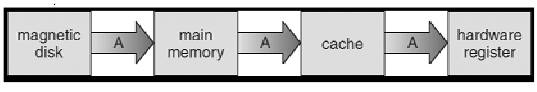
Caching

- Caching: storing parts of data in faster storage for performance.
- Information is normally kept in main memory. As it is used, it copied into a faster storage system- the cache-on a temporary basis
  - Index registers provide high speed cache for main memory
  - Limited size – cache management
  - Data at different levels of hierarchy

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000,000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Caching

- Main memory can be viewed as a fast cache for secondary storage
- Movement of information – explicit vs implicit
- Cache Coherency: In a multiprocessor environment a copy of data A exit in several caches. An update to the value of A in one cache is reflected in all other caches where A resides



I/O Systems

- Hide peculiarities of hardware from the user
- **Components of I/O subsystem**
  - Memory management component – buffering, caching, spooling
  - Device driver interface
  - Drivers for specific hardware

Protection and Security

- *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

Distributed Systems

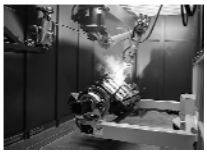
- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that networked and communicate among themselves to perform some task
- Advantages
  - Resource Sharing
  - Higher Performance
  - Fault Tolerance
  - Scalability

Special Purpose Systems

- Real time embedded Systems
- Multimedia Systems
- Handheld systems

Real time embedded systems

- Rigid time constraints - control device
- Sensors bring data – cannot miss the dead line
- Medical Imaging, Industrial Control systems, Automobiles, home appliances, avionics



Multimedia Systems

- Process audio video files
- Live webcasts
- Soft real time system - a little delay wont cause any damage

Hand held Systems

- PDA's palm and pocket PC's , Cellular telephones
- **Issues**
- Limited size – small memory,
- slow processor and
- small display screen

Chapter 2: Operating-System Structures

- Operating System Services
- User Operating System Interface
- System Calls
- Types of System Calls
- System Programs
- Operating System Structure
- Virtual Machines
- System Boot

Operating System Services

- **User Interface**
  - Command line Interface
  - Batch
  - Graphical Interface
- **Program execution** – load, run and terminate a program
- **I/O Operations** – For efficiency and protection, users usually cannot control I/O devices
- **File system manipulation** – read, write, create , delete, search files , permission management

Operating System Services

- **Communications –** Process need to exchange info within the same system or others in network via shared memory or message passing
- **Error Detection –**
  - Errors in CPU and memory h/w (memory error, power failure)
  - i/o devices(connection failure)
  - User Program (arithmetic overflow, illegal memory allocation)
- **Resource Allocation** –
  - manage CPU cycles, main memory and file storage
  - CPU Scheduling – speed of CPU, jobs, no of registers avialble

Operating System Services

- **Accounting –**Keep track of which users use how much and what kind of computer resources
- **Protections and Security**
  - No interference between two process or with OS itself
  - Security from outsiders

User Operating System Interface

- **Command Line Interface (CLI)**
- **Graphical User Interface (GUI)**

Graphical User Interface

- Employ mouse – based windows and menu
- Desktop, mouse, icons
- Xerox Parc –> Apple Macintosh -> Windows
- Unix – K Desktop Environment(KDE)
  - GNOME Desktop by GNU project
- Users choice – CLI or GUI

System Calls

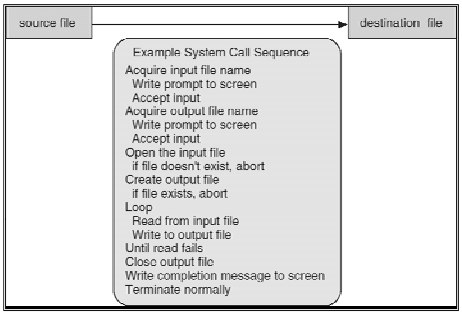
- System calls provide the interface between a running program and the OS
  - Set of functions available to the program to call (but somewhat different from normal functions)
  - These calls available as routines written in C and C++
  - Certain low-level tasks (tasks where hardware must be accessed directly) may be written using assembly-language.
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
- Three most common APIs
  - Win32 API for Windows
  - POSIX API for all versions of UNIX, Linux, Mac OS X
  - Java API for the Java virtual machine (JVM)

System Calls

Why use APIs rather than system calls?
- Portability
- Actual system calls might be more detailed and difficult to work with

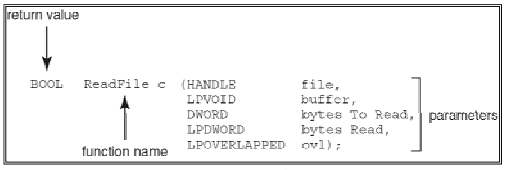
Example of System Calls

- System call sequence to copy the contents of one file to another file



Example of Standard API

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

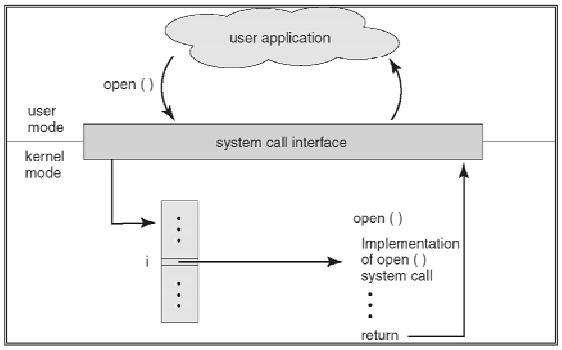


- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read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

System Call Implementation

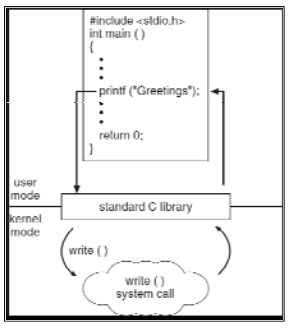
- Typically, a number associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes intended system call in OS kernel and returns status and any return values
- The caller needs to know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result
  - Most details of OS interface hidden by API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

API – System Call – OS Relationship



Standard C Library Example

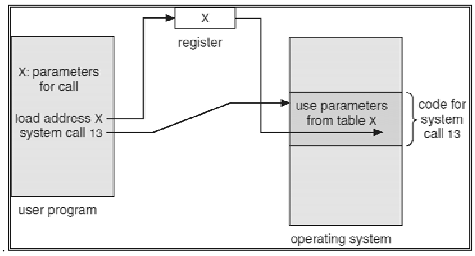
- C program invoking printf() library call, which calls write() system call



System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in ***registers***
  - Parameters stored in a ***block***, or table, in memory, and address of block passed as a parameter in a register
    - E.g. Linux and Solaris
  - Parameters placed, or *pushed,* onto the ***stack*** by the program and *popped* off the stack by the OS
  - Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table



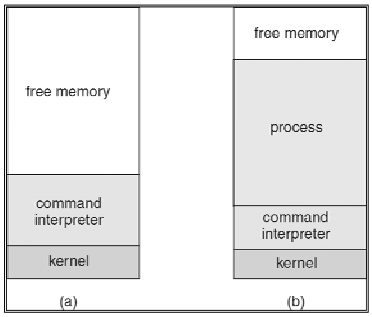
Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communication
- Protection

Examples of Windows and Unix System Calls

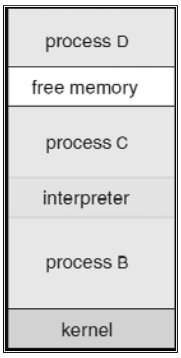
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

MS-DOS execution



(a) At system startup (b) running a program

FreeBSD Running Multiple Programs



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
- Most users' view of the OS is defined by system programs, not the actual system calls

Operating System Design and Implementation

- Design and Implementation of OS not "solvable", but some approaches have proven successful
  - Internal structure of different OS can vary widely
  - Start by defining goals and specifications
  - Affected by choice of hardware, type of system
- Requirements
  - User goals – OS should be convenient to use, easy to learn, reliable, safe, and fast
  - System goals – OS should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

Operating System Design and Implementation (Cont.)

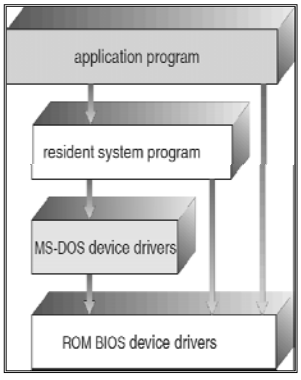
- Important principle to separate
  Policy: What will be done?
  Mechanism: How to do it?
- It allows maximum flexibility if policy decisions are to be changed later

OS Structure

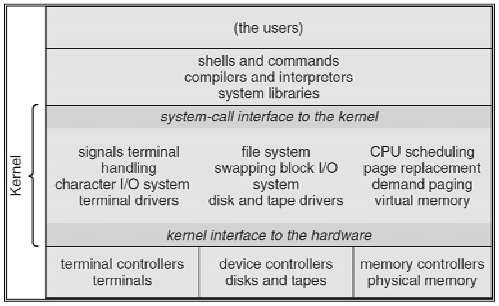
- Four types of structure
  - Simple structure
  - Layered approach
  - Microkernels
  - Modules

OS Structure

- MS-DOS Layered Style (Simple structure)
  - Not divided into modules
  - Start as small, simple, & limited systems and then grow beyond original scope
  - Limited by H/W functionality
  - Application programs access the basic I/O routines to write directly to display and disk

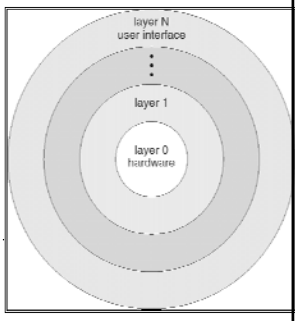


UNIX System Structure



Layered Approach

- The OS 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
- Major difficulty: appropriately defining function of each layers
- They tend to be less efficient (overhead of system calls)



Microkernels

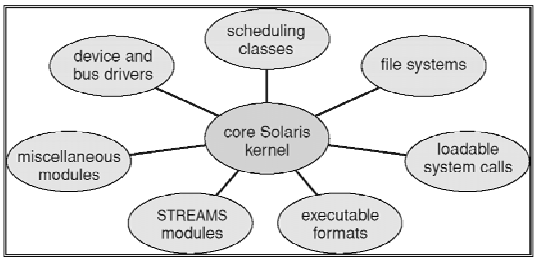
- Remove all nonessential components from the kernel and implementing them as system and user-level programs
- Pass message between process using message passing
- **Benefit**
- Ease of extending OS
- More Security and reliability

- **Disadvantage**
- Poor performance due to system overhead

Modules

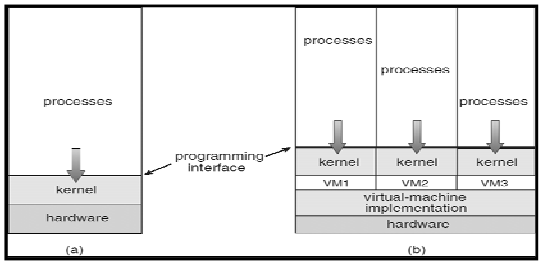
- Most modern OS implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Similar to layers, but more flexible
  - any module can call any other module
- Similar to microkernel, but more efficient
  - No message passing among modules

Solaris Loadable Modules



Virtual Machines

- H/W of a single computer (the CPU, Memory, Disk drives, network interface cards and so on) logically divided into several different execution environments
- Single machine run multiple OSs concurrently each in its own VM



(a) Nonvirtual machine (b) Virtual machine

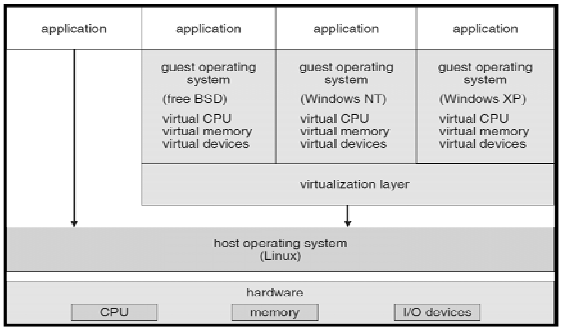
Virtual Machines

- **Benefits**
  - Host system is protected from VMs, just as the VMs are protected from each other
- **Disadvantage**
  - There is no direct sharing of resources
- **Two approaches for sharing resources**
  - First Share a file system volume and then share files
  - Define Network of Virtual Machines, each of which can send info over the virtual communication network

Virtual Machines

- **Why VM??**
  - Make and test changes in OS without modifying the host
  - Rapid porting and testing of programs in various environment
- **Simulation**
  - Emulator translates outdated machine instructions into native instructions.
  - Slower and Writing correct emulator is writing CPU again.

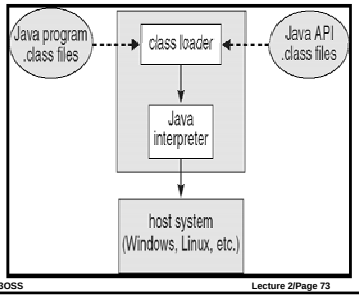
Vmware Architecture



- VMware Workstation runs as an application on a host OS (Windows or Linux) & allow this host system to concurrently run several different guest OSs as Independent VMs.

The Java Virtual Machine

- JVM is specification for an abstract computer consists of loader and interpreter
- JVM implemented in S/W on top of a host OS (such as Windows, Linux, or Mac OS X, or as part of web browser)
- JVM may be implemented in H/W
- JVM manages memory by performing garbage collection



System Boot

- **Booting:** starting a computer by loading the kernel
- **Bootstrap program or bootstrap loader** –code stored in ROM that is able to locate the kernel, load it into memory, and start its execution
- ROM vs EPROM
- All formats of ROM known as Firmware – slower and expensive
- **Boot Block**
  - The program stored in the boot block may load entire OS into memory and begins its execution

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* (***exception***) is a software-generated interrupt caused either by an error or a user request.
- The operating system is *interrupt* driven.