1.1**What Operating Systems Do**

The operating system controls the hardware and coordinates its use among the various application programs for the various users.

1.1.1 User View

**1-single user:** the operating system is designed mostly for **ease of use**, with some attention paid to performance not the resource utilization.

**2-multiusers** sit at a terminal connected to a **mainframe** or a **minicomputer:** The operating system is designed to maximize resource utilization— to assure that all available CPU time, memory, and I/O are used efficiently.

3-users sit at **workstations** connected to networks of other workstations and **servers: they have dedicated and shared resources:** their operating system is designed to compromise between individual usability and resource utilization.

4-mobile computers: The user interface features a **touch screen**.

5-Some computers have little or no user view. EX: Embedded computers in home devices and automobiles

1.1.2 System View

1-We can view an operating system 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/O devices, and so on.

2- As a **control program** manages the execution of user programs to prevent errors and improper use of the computer. It

1.1.3 Defining Operating Systems

**Moore’s Law** predicted that **the number of transistors on an integrated circuit would double every eighteen months, and that prediction has held true.**

 Computers gained in functionality and shrunk in size, leading to a vast number of uses and a vast number and variety of operating systems.

* no completely adequate definition of an operating system

1- The common functions of controlling and allocating resources such as those controlling the I/O devices are brought together into one piece of software: the operating system.

2-A more common 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**: associated with the operating system but are not necessarily part of the kernel

**Application programs:** include all programs not associated with the operation of the system.

3-Mobile operating systems often include not only a core kernel but also **middleware**—a set of software frameworks that provide additional services to application developers.

1.2 **Computer-System Organization**

1.2.1 Computer-System Operation

**Bootstrap program (firmware):** an initial program needed for a computer to power up or reboot.

* Stored within the computer hardware in (**ROM**) or electrically erasable programmable read-only memory (**EEPROM**).
* It initializes all aspects of the system “how to load the operating system and how to start executing that system” from CPU registers to device controllers to memory contents.
* The bootstrap program locate the operating-system kernel and load it into memory.
* **System processes**, or **system daemons:** run the entire time the kernel is running.

On UNIX, the first system process is “init,” and it starts many other daemons.

* **The occurrence of an event** is usually signaled by an **interrupt from either the hardware or the software.**
* **Hardware** may triggeran interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
* **Software** may trigger an interrupt by executing a special operation called a system call (also called a **monitor call).**

1.2.2 Storage Structure

**RAM: where** computers run most of their programs from.

**Dynamic random-access memory (DRAM)**: where Main memory is implemented.

**ROM:** cannot be changed: used to store on static programs like: bootstrap

 **EEPROM**: can be changed but not frequently: used in smartphones.

* Ideally, we want the programs and data to reside in main memory permanently. But we can’t why?

**1.** Main memory is usually too small to store all needed programs and data permanently.

**2.** Main memory is a **volatile** storage device that loses its contents when power is turned off or otherwise lost.

Thus, most computer systems provide: \* **secondary storage \* Magnetic disk**

* The main differences among the various storage systems lie in speed, cost, size, and volatility.
* wide variety of storage systems can be organized in a hierarchy
* The top four levels of memory in Figure 1.4 may be constructed using semiconductor memory.
* In addition to differing in speed and cost, the various storage systems are either volatile or nonvolatile.
* Solid-state disks: faster than magnetic disks and are nonvolatile.
* Flash memory and DRAM are forms of solid-state disk. Flash memory is slower than DRAM but needs no power to retain its contents.
* Another form of nonvolatile storage is NVRAM, which is DRAM with battery backup power. This memory can be as fast as DRAM and (as long as the battery lasts) is nonvolatile.

1.2.3 I/O Structure

**A general-purpose computer system consists of:**

1-CPUs. 2- Multiple **device controllers**: connected through a common bus.

EX: seven or more devices can be attached to the **small computer-systems interface (SCSI)** controller.

\*It maintains some local buffer storage and a set of special-purpose registers.

\*It moves the data between the peripheral devices that it controls and its local buffer storage.

3- **Device driver** for each device controller: provides the rest of the operating system with a uniform interface to the device.

* To start an I/O operation:
1. Device driver loads the appropriate registers within the device controller. The device controller, in turn, examines the contents of these registers to determine what action to take.
2. The controller starts the transfer of data from the device to its local buffer.
3. The device controller informs the device driver via an interrupt that it has finished its operation.
4. Device driver then returns control to the OS, possibly returning **the data** or a **pointer** to the data if the operation was a read.
5. For other operations, the device driver returns **status** information.
* This form of interrupt-driven I/O is fine for moving **small** amounts of data but can produce high overhead when used for bulk data. To solve this problem: **direct memory access (DMA)** is used
: the device controller transfers an entire block of data WITH NO intervention by CPU except one interrupt is generated per block, to tell the device driver that the operation has completed

1.3 Computer-System Architecture

1.3.1 Single-Processor Systems

**One main CPU:** instruction set, other special-purpose processors :( device-specific processors like keyboard and general-purpose processors, such as I/O processors).

**Special-purpose processors:** run a limited instruction set and do not run user processes:

1. Managed by the operating system: EX: a disk-controller microprocessor receives a sequence of requests from the main CPU and implements its own disk queue and scheduling algorithm.
2. Special-purpose processors that are low-level components built into the hardware. The operating system cannot communicate with these processors.

1.3.2 Multiprocessor Systems (**parallel systems** or **multicore systems**)

Two or more processors sharing the computer bus and sometimes the clock, memory, and peripheral devices.

Multiprocessor systems have three main advantages:

**1. Increased throughput:** more work done in less time.

**2. Economy of scale**: cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies.

**3. Increased reliability** the failure of one processor will not halt the system, only slow it down.

**Graceful degradation”"التراجع الرشيق**: The ability to continue providing service proportional to the level of surviving hardware is called.

**Fault tolerant"قادر على تحمل الخطأ"**, systems that can suffer a failure of any single component and still continue operation by allow the failure to be detected, diagnosed, and, if possible, corrected. EX: The HP NonStop.

**The multiple-processor systems in two types:**

**Asymmetric multiprocessing**, a boss–worker relationship. A ***boss*** processor controls the system; the other processors either look to the boss for instruction or have predefined tasks.

**Symmetric multiprocessing (SMP)**, SMP means that all processors are peers; no boss–worker relationship exists between processors .EX: AIX, a commercial version of UNIX designed by IBM.

Benefit of this model: many processes can run simultaneously

The difference between symmetric and asymmetric multiprocessing may result from either hardware or software. Special hardware can differentiate the multiple processors, or the software can be written to allow only one boss and multiple workers.

**Multicore (Dual core design “two cores”):**



1-More efficient than multiple chips with single cores because on-chip communication is faster than between-chip communication.

2- Uses significantly less power than multiple single-core chips.

**Blade servers**: recent development in which multiple processor boards, I/O boards, and networking boards are placed in the same chassis. **Each blade-processor board boots independently and runs its own operating system.**

1.3.3 Clustered Systems



**Another type of multiprocessor system which gathers together multiple CPUs, composed of two or more individual systems—or nodes—joined together.**

**Benefits of clustering:**

**1-High-availability** service: service will continue even if one or more systems in the cluster fail.

**2-**Clustering can be structured asymmetrically or symmetrically.

**Asymmetric clustering**, one machine is in **hot-standby mode** while the other isrunning the applications

**Symmetric clustering**, two or more hosts are running applications and are monitoring each other. More efficient.

3- **high-performance computing** environments.

4- **Parallelization:** divides a program into separate components that run in parallel on individual computers in the cluster.

**5- Distributed lock manager (DLM)**, each machine has full access to all data in the database. To provide this shared access, the system must also supply access control and locking to ensure that no conflicting operations occur.

1.4 **Operating-System Structure**

|  |  |
| --- | --- |
| **Multiprogramming** increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute. | **Time sharing** (or **multitasking**) is a logical extension of multiprogramming. |
| in general, main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the **job pool** | 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. |
| This pool consists of all processes residing on disk awaiting allocation of main memory. | requires an **interactive** computer system,” direct communication between the user and the system with short response time.” |
| The OS 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 | uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer. |
| When ***that*** job needs to wait, the CPU switches to ***another*** job, and so on. Eventually, the first job finishes waiting and gets the CPU back. As long as at least one job needs to execute, the CPU is never idle. | \*A time-sharing system must also provide a file system\* In addition, a time-sharing system providesa mechanism for protecting resources from inappropriate use |

**Job scheduling: 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.

**CPU scheduling**, if several jobs are ready to run at the same time,the system must choose which job will run first.

Operating system must ensure reasonable response time. This accomplished through:

**1-Swapping**: whereby processes are swapped in and out of main memory to the disk.

**2-virtual memory**, a technique that allows the execution of a process that is not completely in memory.

Advantage of virtual memory:

\*it enables users to run programs that are larger than actual **physical memory**.

\*it abstracts main memory into a large, uniform array of storage, separating **logical memory** from physical memory.

1.5 **Operating-System Operations**

* Modern operating systems are **interrupt driven**.
* Events are almost always signaled by the occurrence of an interrupt or a trap.
* trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access or by a specific request from a user program .
* Interrupt-driven nature of an operating system defines that system’s general structure.
* A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

1.5.1 Dual-Mode and Multimode Operation

we need two separate ***modes*** of operation: **user mode** and **kernel mode** (also called **supervisor mode**, **system mode**, or **privileged mode**).

**Mode bit**: a bit that is added to the hardware of the computer to

* Indicate the current mode: kernel (0) or user (1).
* Distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user (user mode).



|  |  |
| --- | --- |
| User mode: when a user application requests a service from the OS (via a system call), the system must transition from user to kernel mode to fulfill the request. | Kernel mode: The system always switches to user mode (by setting the mode bit to 1) before passing control to a user program. |
| At system boot time, the hardware starts in kernel mode. The OS is then loaded and starts user applications in user mode. |
| Whenever a trap or interrupt occurs, the hardware switches to kernel mode. whenever the OS gains control of the computer, it is in kernel mode |

The dual mode of operation provides us with the means for protecting the operating system from errant users—and errant users from one another.BY:

1. **Privileged instructions:** executed only in kernel mode OR the hardware treats it as illegal and traps it to the operating system.

The instruction to switch to kernel mode is an example of a privileged instruction. Some other examples include I/O control, timer management, and interrupt management.

1. CPUs that support virtualization: virtual machine manager (VMM)—and the virtualization management software—is in control of the system. In this mode, the VMM has more privileges than user processes but fewer than the kernel.
2. CPUs supports four ***privilege levels,*** for example, and supports virtualization but does not have a separate mode for virtualization. EX: the Intel 64.

A **system call** usually takes the form of a trap to a specific location in the interrupt vector.

This trap can be executed by a generic trap instruction, although some systems (such as MIPS) have a specific syscall instruction to invoke a system call.

Now see the life cycle of instruction execution in a computer system.look page23

1.5.2 Timer

**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).

 A **variable timer** is generally implemented by a fixed-rate clock and a counter.

We can use the timer to prevent a user program from running too long.

A simple technique is to initialize a counter with the amount of time that a program is allowed to run.

1.6 **Process Management**

* A program in execution is a **process**. EX: A time-shared user program such as a compiler is a process.
* A process needs certain resources—including CPU time, memory, files, and I/O devices—to accomplish its task.
* These resources are either given to the process when it is created or allocated to it while it is running.
* A program by itself is not a process. A program is a ***passive*** entity, like the contents of a file stored on disk, whereas a process is an ***active*** entity.
* A **single-threaded** process has one **program counter** specifying the next instruction to execute.The execution of such a process must be **sequential**.
* A process is the unit of work in a system. A system consists of a collection of processes, some of which are operating-system processes (those that execute system code) and the rest of which are user processes (those that execute user code).

The operating system is responsible for:

• Scheduling processes and threads on the CPUs

• Creating and deleting both user and system processes

• Suspending and resuming processes

• Providing mechanisms for process synchronization

• Providing mechanisms for process communication

1.7 **Memory Management**

**Main memory is a large array of bytes, each byte has its own address.**

* The main memory is generally the only large storage device that the CPU is able to address and access directly
* To improve both the utilization and speed of the CPU: keep several programs in memory, creating a need for memory management.
* The operating system is responsible for the following activities in connection with memory management:

• Keeping track of which parts of memory are currently being used and who is using them.

• Deciding which processes (or parts of processes) and data to move into and out of memory.

• Allocating and deallocating memory space as needed.

1.8 **Storage Management**

The OS abstracts from the physical properties of its storage devices to define a logical storage unit, the **file**.

1.8.1 File-System Management

Computers can store information on Magnetic disk, optical disk, and magnetic tape which are the most common.

Each medium is controlled by a device, such as a disk drive or tape drive.

**A file** is a collection of related information defined by its creator.

Files **represent** programs and data. **Data files may be numeric, alphabetic, alphanumeric, or binary**.

The OS implements the abstract concept of a file by managing mass-storage media, such as tapes and disks, and the devices that control them.

Files are normally organized into directories to make them easier to use.

The operating system is responsible for the following activities in connection with file management:

• 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.

1.8.2 Mass-Storage Management

Most modern computer systems use disks as the **principal on-line storage** medium for both programs and data because main memory is too small, and because the data that it holds are lost when power is lost.

The OS is responsible for the following activities in connection with disk management:

• Free-space management

• Storage allocation

• Disk scheduling

Many uses for storage that is slower and lower in cost (and sometimes of higher capacity) than secondary storage EX:

* Backups of disk data, storage of seldom-used data, and long-term archival storage.

* **Tertiary "فوق الثانوي"storage** devices: Magnetic tape drives and their tapes.

The media (tapes and optical platters) vary between **WORM** (write-once, read-many-times) and **RW** (read– write) formats.

Some of the functions that OS can provide include mounting and unmounting media in devices,

• allocating and freeing the devices for exclusive use by processes.

• Migrating data from secondary to tertiary storage.

1.8.3 Caching

**Caching** is an important principle of computer systems.

\*When information is needed to use, it is copied from memory into a faster storage system—the cache—\*When we need a particular piece of information:

1-First check whether it is in the cache to use it directly.

2-If it’s not there use the information from the source and put a copy in the cache.

**Types of cache:**

\*internal programmable registers, such as index registers, provide a high-speed cache for main memory.

Programmer (or compiler) implements the register-allocation and register-replacement algorithms to decide whether to keep in registers or in main memory.

\* Other caches are implemented in hardware:1- **instruction cache** to hold the instructions expected to be executed next.2- **high-speed data caches** in the memory hierarchy.

Because caches have limited size, **cache management** is an important design problem.

* Data must be in main memory before being moved to secondary storage for safekeeping.
* The magnetic-disk storage is often backed up onto magnetic tapes or removable disks to protect against data loss in case of a hard-disk failure.
* Some systems automatically archive old file data from secondary storage to tertiary storage, such as tape jukeboxes, to lower the storage cost.
* The movement of information between levels of a storage hierarchy may be either ***explicit*** or ***implicit*** depending on the hardware design and the controlling operating-system software.
* The same data may appear in different levels of the storage structure
* Multiprocessor environment: in addition to maintaining internal registers, each of the CPUs also contains a local cache.
* 

**Cache coherency:** Since the various CPUs can all execute in parallel, we must make sure that an update to the value of A in one cache is done to all other caches where A resides.

Distributed environment: several copies (or replicas) of the same file can be kept on different computers.

1.8.4 I/O Systems

One of the purposes of an operating system is to hide the peculiaritiesخصائص-مميزات of specific hardware devices from the user. EX:

**The I/O subsystem consists of several components:**

• A memory-management component that includes buffering, caching, and spooling.

• A general device-driver interface.

• Drivers for specific hardware devices.

Only the device driver knows the peculiarities of the specific device to which it is assigned.

1.9 **Protection and Security**

Mechanisms ensure that files, memory segments, CPU, and other resources can be operated on by only processes that have gained proper authorization from the operating system.

**Protection**, is any mechanism for controlling the access of processes or users to the resources defined by a computer system.

PROTECTION improve **reliability** by detecting latent errors at the interfaces between component subsystems.

**Security** defend a system from external and internal attacks. These attacks include viruses and worms, denial-of service attacks…

1-**user identifiers (user IDs)**.

**2-Group identifiers** distinguish among sets of users rather than individual users.

**3-Escalate privileges** to gain extra permissions for an activity (access to a device that is restricted).

1.10 **Kernel Data Structures**

Several fundamental data structures used extensively in operating systems.

1.10.1 Lists, Stacks, and Queues

* An **array** is a simple data structure in which each element can be accessed directly.
* After arrays, **lists**. Whereas items in a list must be accessed in a particular order.

Common method for implementing this structure is a **linked list**, in which items are linked to one another to allow easy insertion and deletion of items. Linked lists are of several types:

• In a ***singly linked list,*** each item points to its successor, as illustrated in Figure 1.13.

• In a ***doubly linked list,*** a given item can refer either to its predecessor or to its successor, as illustrated in Figure 1.14.

• In a ***circularly linked list,*** the last element in the list refers to the first element, rather than to null, as illustrated in Figure 1.15.



**Disadvantage of using a list:** performance for retrieving a specified item in a list of size *n* is linear — *O*(*n*), as it requires potentially traversing all *n* elements in the worst case.

* Lists are **used for** constructing more powerful data structures, such as stacks and queues.

|  |  |
| --- | --- |
| Stack is a sequentially ordered data structure that uses the last in, first out (LIFO) principle for adding and removing items, the last item placed is the first one removed.Operation for inserting and removing items on stack: push and pop. | A queue, in contrast, is a sequentially ordered data structure that uses the first in, first out **(FIFO)** principle: items are removed from a queue in the order in which they were inserted. |
| An OS often uses a stack when invoking function calls. Parameters, local variables, and the return address are **pushed** onto the stack when a function is called; returning from the function call **pops** those items off the stack. | \*shoppers waiting in a checkout line at a store and cars waiting in line at a traffic signal.\*also quite common in operating systems—jobsthat are sent to a printer are typically printed in the order in which they weresubmitted |

1.10.2 Trees

A **tree** is a **data structure that can be used to represent data hierarchically parent–child relationships**.

**General tree**, a parent may have an unlimited number of children.

**Binary tree**, a parent may have at most two children, which we term the ***left child*** and the ***right child***. **Binary search tree** additionally requires an orderingbetween the parent’s two children in which *le f t child <*= *right child.*

* When we search for an item in a binary search tree, the worst-case performance is *O* (*n*).
* A tree containing *n* items has at most ***lg*** *n* levels, thus ensuring worst-case performance of *O* (*lg n*).

1.10.3 Hash Functions and Maps

A **hash function** takes data as its input numeric operations numeric value.

* Hash functions are used extensively in operating systems because searching for a data item through a list of size *n* can require up to *O*(*n*) comparisons in the worst case.
* Difficulty : is that two inputs can result in the same output value—that is, they can link to the same table location. **Hash collision**

Solution: linked list at that table location that contains all of the items with the same hash value.

* Usage: is to implement a hash map, which associates (or maps) [key: value] pairs using a hash function.

1.10.4 Bitmaps

A **bitmap is a string of *n* binary digits that can be used to represent the status of *n* items.**

The value of the $i^{th}$position in the bitmap is associated with the *ith* resource.

 As an example, consider the bitmap shown below:

001011101

Resources 2, 4, 5, 6, and 8 are unavailable; resources 0, 1, 3, and 7 are available.

* Bitmaps are commonly used when there is a need to represent the availability of a large number of resources .EX: disk drive might be divided into several thousand individual units, called disk blocks and bitmaps used to availability of each block.

1.11 **Computing Environments**

How operating systems are used in a variety of computing environments.

1.11.1 Traditional Computing

* Companies establish **portals**, which provide Web accessibility to their internal servers. **Network computers** (or **thin clients**)—which are essentially terminals used in place of traditional workstations.
* Mobile computers can also connect to **wireless networks** and cellular data networks to use the company’s Web portal.
* Many homes use **firewalls** to protect their networks from security breaches.
* For a period of time, systems were either batch or interactive.

Batch systems processed jobs in bulk, with predetermined input from files or other data sources.

Interactive systems waited for input from users.

* Today, traditional time-sharing systems are uncommon.

User processes, and system processes that provide services to the user, are managed so that each frequently gets a slice of computer time.

1.11.2 Mobile Computing

* **Refers to computing on handheld smartphones and tablet computers.**
* Mobile systems are used not only for e-mail and web browsing but also for playing music and video, reading digital books, and so on.

**Unique features:**

1-**GPS chip** allows a mobile device to use satellites to determine its precise location on earth.

2-**An accelerometer** allows a mobile device to detect its orientation with respect to the ground and to detect certain other forces, such as tilting and shaking.

* To provide access to on-line services, mobile devices typically use either IEEE standard 802.11 wireless or cellular data networks.
* Memory capacity and processing speed are limited.
* Two operating systems currently dominate mobile computing: **Apple iOS** and **Google Android**.

1.11.3 Distributed Systems

**A collection of physically separate, possibly heterogeneousمختلف الخصائص, computer systems that are networked to provide users with access to the various resources that the system maintains.**

* Increases computation speed, functionality, data availability, and reliability.
* A **network**, is a communication path between two or more systems.
* Distributed systems depend on networking for their functionality.
* Networks vary by the **protocols used,** the **distances between nodes**, and the **transport media**.
* TCP/IP is the most common network protocol.
* **A local-area network (LAN)** connects computers within a room, a building, or a campus.

**A wide-area network (WAN)** usually links buildings, cities, or countries.

**Metropolitan-area network (MAN)** could link buildings within a city.

**Personal-area network (PAN)** BlueTooth and 802.11 devices use wireless technology to communicate over a distance of several feet, between a phone and a headset or a smartphone and a desktop computer

* **The media to carry networks:** copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios.
* **A network operating system** is an OS that provides file sharing across the network that allows different processes on different computers to exchange messages.
* A network operating system acts autonomously from all other computers.
* A distributed operating system provides a **less** autonomous environment.

1.11.4 Client–Server Computing

Many of today’s systems act as **server systems** to satisfy requests generated by **client systems**. This form of specialized distributed system, called a client–server system.

***Server systems can be broadly categorized as:***

* The **compute-server system** provides an interface to which a client can send a request to perform an action (for example, read data). In response, the server executes the action and sends the results to the client.
* The **file-server system** provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers files to clients running web browsers.



1.11.5 Peer-to-Peer Computing

In this model**, all nodes within the system are considered peers, and each may act as either a client or a server.**

A node must first join the network of peers. Once a node has joined the network, it can begin providing services to—and requesting services from—other nodes in the network.

* Centralized lookup service on the network: Any node desiring a specific service first contacts this centralized lookup service to determine which node provides the service.EX: Napster.
* No centralized lookup service: a peer acting as a client must discover what node provides a desired service using a discovery protocol.EX: Gnutella.

**Skype** is another example of peer-to-peer computing using a technology known as voice over IP (VoIP).

1.11.6 Virtualization: **a technology that allows operating systems to run as applications within other operating systems.**

* Virtualization is one member of a class of software that also includes emulation.
* Emulation when the source CPU type is different from the target CPU type.EX: “Rosetta,” which allowed applications compiled for the IBM CPU to run on the Intel CPU.
* Emulated code can run much slower than the native code on similar performance devices.
* Common example: **interpretation**: when a computer language is not compiled but instead is either executed in its high-level form or translated to an intermediate form EX:Java.
* Windows was the **host** operating system, and the VMware application was the virtual machine manager VMM. The VMM runs the guest operating systems, manages their resource use, and protects each guest from the others.
* VMMs like VMware, ESX, and Citrix XenServer no longer run on host operating systems but rather are the hosts.
* 

1.11.7 Cloud Computing

**Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network.” logical extension of virtualization”**

 **Types of cloud computing:**

• Public cloud—a cloud available via the Internet to anyone willing to pay for the services.

• Private cloud—a cloud run by a company for that company’s own use

• Hybrid cloud—a cloud that includes both public and private cloud components

• Software as a service (**SaaS**)—one or more applications (such as word processors or spreadsheets) available via the Internet

• Platform as a service (**PaaS**)—a software stack ready for application use via the Internet (for example, a database server)

• Infrastructure as a service (**IaaS**)—servers or storage available over the Internet (for example, storage available for making backup copies of production data)

* These types are not discrete, as a cloud computing environment may provide a combination of several types.



1.11.8 Real-Time Embedded Systems

* Ex :car engines and manufacturing robots to DVDs and microwave ovens-
* have very specific task-limited features have little or no user interface
* Some are general-purpose computers, running standard OS—such as Linux—with special-purpose applications to implement the functionality.
* Other are hardware devices with a special-purpose embedded operating system.
* Embedded systems run **real-time** operating systems. **A real-time system** is used when rigid time requirements have been placed on the operation of a processor or the flow of data.
* Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are ***real time*** systems.
* A real-time system has *well-defined, fixed time constraints”* *Processing must be done within the defined constraints, or the system will fail.*

1.12 Open-Source Operating Systems

* **Open-source operating systems** are those available in source-code format rather than as compiled binary code. EX: LINUX
* Microsoft Windows is a well-known example of the opposite **closed-source** approach.
* Apple’s Mac OS X and iOS operating systems comprise a *hybrid* approach. They contain an open-source kernel named Darwin yet include proprietary, closed-source components as well.
* **Reverse engineering:** executing the source code from the binaries.
* Benefits to open-source operating systems:
* 1- A student can modify the operating system (community of interested (and usually unpaid) programmers who contribute to the code by helping to debug it, analyze it, provide support, and suggest changes.)
* 2- open-source code is more secure than closed-source code because many more eyes are viewing the code.

1.12.1 History

* In the early days of modern computing (that is, the 1950s), a great deal of software was available in open-source format.
* Computer and software companies eventually sought to limit the use of their software to authorized computers and paying customers by Releasing only the binary files compiled from the source code.
* Another issue involved copyrighted material. **Copy protection** or **digital rights management (DRM)** would not be effective if the source code that implemented these limits were published.
* Open-source, UNIX compatible operating system and the **Free Software Foundation (FSF)** with the goal of encouraging the free exchange of software source code and the free use of that software was created.
* The GNU **General Public License (GPL)** codifies copylefting and is a common license under which free software is released.

1.12.2 Linux

* The GNU project produced many UNIX-compatible tools, including compilers, editors, and utilities, but never released a kernel.
* A rudimentary UNIX-like kernel using the GNU compilers and tools was released.
* The resulting GNU/Linux OS has spawned hundreds of unique **distributions**, or custom builds, of the system. Major distributions include RedHat, SUSE,…
* Distributions vary in function, utility, installed applications, hardware support, user interface, and purpose.
* EX: **PCLinuxOS** is a **LiveCD**—an operating system that can be booted and run from a CD-ROM without being installed on a system’s hard disk.

1.12.3 BSD UNIX

**BSD UNIX** has a longer and more complicated history than Linux.

Just as with Linux, there are many distributions of BSD UNIX, including FreeBSD, NetBSD, OpenBSD, and DragonflyBSD.

Darwin, the core kernel component of Mac OS X, is based on BSD UNIX and is open-sourced as well.

1.12.4 Solaris

* **Solaris** is the commercial UNIX-based operating system of Sun Microsystems.
* Originally, Sun’s **SunOS** operating system was based on BSD UNIX.
* Project Illumes, which has expanded from the OpenSolaris base to include more features.