1. **With a neat diagram, explain the functions of the operating systems.**

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Definition

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



Following are some of important functions of an operating System.

* Memory Management
* Processor Management
* Device Management
* File Management
* Security
* Control over system performance
* Job accounting
* Error detecting aids
* Coordination between other software and users

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management −

* Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
* In multiprogramming, the OS decides which process will get memory when and how much.
* Allocates the memory when a process requests it to do so.
* De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management −

* Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.
* Allocates the processor (CPU) to a process.
* De-allocates processor when a process is no longer required.

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management −

* Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
* Decides which process gets the device when and for how much time.
* Allocates the device in the efficient way.
* De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management −

* Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
* Decides who gets the resources.
* Allocates the resources.
* De-allocates the resources.

Other Important Activities

Following are some of the important activities that an Operating System performs −

* Security − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
* Control over system performance − Recording delays between request for a service and response from the system.
* Job accounting − Keeping track of time and resources used by various jobs and users.
* Error detecting aids − Production of dumps, traces, error messages, and other debugging and error detecting aids.
* Coordination between other softwares and users − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

1. **List and explain breifly the types of processing in Operating Systems.**

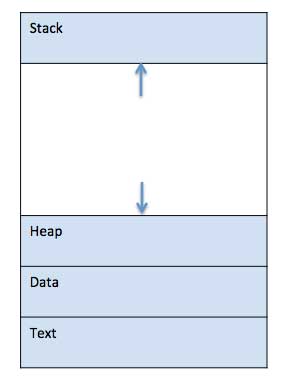
## Process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data. The following image shows a simplified layout of a process inside main memory −



|  |  |
| --- | --- |
| **S.N.** | **Component & Description** |
| 1 | **Stack**  The process Stack contains the temporary data such as method/function parameters, return address and local variables. |
| 2 | **Heap**  This is dynamically allocated memory to a process during its run time. |
| 3 | **Text**  This includes the current activity represented by the value of Program Counter and the contents of the processor's registers. |
| 4 | **Data**  This section contains the global and static variables. |

## Program

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C programming language −

#include <stdio.h>

int main() {

printf("Hello, World! \n");

return 0;

}

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

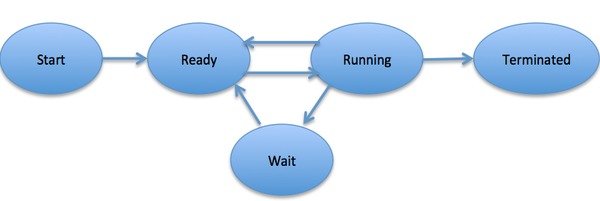
A part of a computer program that performs a well-defined task is known as an **algorithm**. A collection of computer programs, libraries and related data are referred to as a **software**.

## Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

In general, a process can have one of the following five states at a time.

|  |  |
| --- | --- |
| **S.N.** | **State & Description** |
| 1 | **Start**  This is the initial state when a process is first started/created. |
| 2 | **Ready**  The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after **Start** state or while running it by but interrupted by the scheduler to assign CPU to some other process. |
| 3 | **Running**  Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions. |
| 4 | **Waiting**  Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available. |
| 5 | **Terminated or Exit**  Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory. |

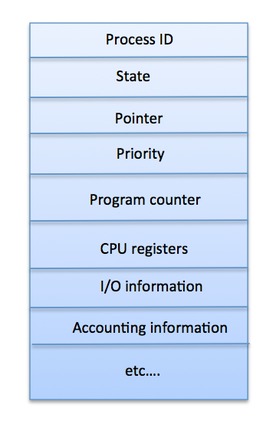


## Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table −

|  |  |
| --- | --- |
| **S.N.** | **Information & Description** |
| 1 | **Process State**  The current state of the process i.e., whether it is ready, running, waiting, or whatever. |
| 2 | **Process privileges**  This is required to allow/disallow access to system resources. |
| 3 | **Process ID**  Unique identification for each of the process in the operating system. |
| 4 | **Pointer**  A pointer to parent process. |
| 5 | **Program Counter**  Program Counter is a pointer to the address of the next instruction to be executed for this process. |
| 6 | **CPU registers**  Various CPU registers where process need to be stored for execution for running state. |
| 7 | **CPU Scheduling Information**  Process priority and other scheduling information which is required to schedule the process. |
| 8 | **Memory management information**  This includes the information of page table, memory limits, Segment table depending on memory used by the operating system. |
| 9 | **Accounting information**  This includes the amount of CPU used for process execution, time limits, execution ID etc. |
| 10 | **IO status information**  This includes a list of I/O devices allocated to the process. |

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB −



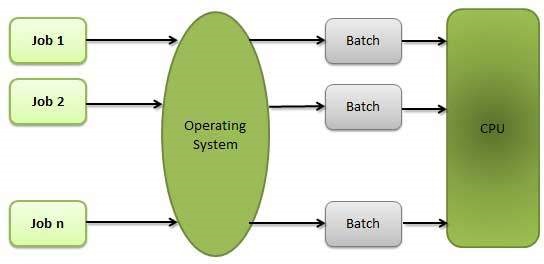
The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

1. **With a neat diagram, explain the concept of batch processing.**

## Batch processing

Batch processing is a technique in which an Operating System collects the programs and data together in a batch before processing starts. An operating system does the following activities related to batch processing −

* The OS defines a job which has predefined sequence of commands, programs and data as a single unit.
* The OS keeps a number a jobs in memory and executes them without any manual information.
* Jobs are processed in the order of submission, i.e., first come first served fashion.
* When a job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.



### **Advantages**

* Batch processing takes much of the work of the operator to the computer.
* Increased performance as a new job get started as soon as the previous job is finished, without any manual intervention.

### **Disadvantages**

* Difficult to debug program.
* A job could enter an infinite loop.
* Due to lack of protection scheme, one batch job can affect pending jobs.

1. **List and explain briefly about the classification of software applications.**

Applications software is capable of dealing with user inputs and helps the user to complete the task. It is also called end-user programs or only an app. It resides above system software. First user deal with system software after that he/she deals with application software. The end user uses applications software for a specific purpose. It programmed for simple as well as complex tasks. It either be installed or access online. It can be a single program or a group of small programs that referred to as an application suite. Some examples of **Application Software are**Word processing software, Spreadsheets Software, Presentation, Graphics, CAD/CAM, Sending email etc.

**Types of Application Software:**According to the need of users it is categorized into following types.

**1) Presentation Software:**Presentation program is a program to show the [information](https://ecomputernotes.com/fundamental/information-technology/what-do-you-mean-by-data-and-information) in the form of slides. We can add text, graphics video and images to slides to make them more informative.

The software has three components:

1) Text editor for inputting and formatting text.

2) Inserting graphics, text, video and other multimedia files.

3) Slideshow to display the [information](https://ecomputernotes.com/fundamental/information-technology/what-do-you-mean-by-data-and-information).

Presentation software helps the presenter to present their ideas with ease and visual information easy to understand. **Example of presentation software:**Microsoft’s PowerPoint and Apple’s Keynote.

**2) Spreadsheet Software:**Spreadsheet software is used to perform manipulate and calculations. In spreadsheet software data is stored in intersection row and column. The intersection of row and column is known as a cell. The cell labelled with the row and column label like A1, A2 etc. While entering data into the cell, we can also define the data value like text, date, time, number. It provides many formula and function to perform calculations like arithmetic operations, logical operations, text operation etc. It provides charts, graphs to display data graphically. For example Microsoft Excel, lotus 1-2-3 for windows and number for MAC OS

**3) Database software:** Database is a collection of data related to any applications. Today is environment every application has some database where data regarding users stored. For this purpose, we used database software. When we operate the application data is accessed from the database, and after manipulation, it gets back stored in the database.

Database Management System (DBMS) software tool used for storing, modifying extracting and searching for information within a database. MySQL, MS Access, Microsoft SQL Server and Oracle is the example of database application Software.

**4) Multimedia Software:**Multimedia is a combination of text, graphics, audio and Multimedia software used in the editing of video, audio and text. Multimedia software used in the growth of business, educations, information, remote system and entertainment.

**5) Simulation Software:**Simulation is an imitation of real world and environment. The simulation creates a physical environment of the real world to represent the similar behaviour, function and key nature of the selected topic. Simulation is technology for education, engineering, testing, training, video games and for scientific modelling of natural systems to gain insight into their functioning. The simulation used in the area of the real world where the real system cannot be accessible or may be dangerous or unacceptable. Area of technology flight, economics, automobiles, Robotics, digital lifecycle, Space Shuttle Navigation, weather.

**6) Word Processing Software:** Word Processing software is used to manipulate, format the text, to create memos, letters, faxes and documents. Processing Software is used to format and beautify the text. It provides a list of features. Likethesaurus, the option provides synonyms, antonyms and related words for chosen word or phrase. Find and replace feature enables users to scan and replace selected words or phrases in the document. Font option provides font colour, font style, font effect, font size to modify the txt. Word Art option to modify or animated titles, hyphens, columns and text boxes in documents. Grammar and Spelling check option available for checking errors. Many more option is listed here in software.

**For example**Microsoft Word, Lotus Word Pro, Word pad and Corel WordPerfect.

1. **List and explain difference between single task vs multitask vs multiapplications.**

Multi-tasking vs. single-tasking  
A [multi-tasking](http://en.wikipedia.org/wiki/Multi-tasking) operating system allows more than one program to be running at the same time, from the point of view of human time scales. A single-tasking system has only one running program. Multi-tasking can be of two types: preemptive and co-operative. In preemptive multitasking, the operating system slices the CPU time and dedicates one slot to each of the programs. Unix-like operating systems such as Solaris and Linux support preemptive multitasking, as does [Amiga OS](http://en.wikipedia.org/wiki/AmigaOS). Cooperative multitasking is achieved by relying on each process to give time to the other processes in a defined manner. [16-bit](http://en.wikipedia.org/wiki/16-bit) versions of Microsoft Windows used cooperative multi-tasking. [32-bit](http://en.wikipedia.org/wiki/32-bit) versions of both Windows NT and  , used preemptive  multi-tasking. Mac OS prior to OS X used to support cooperative multitasking.

1. **What are device drivers. Explain briefly its functions**

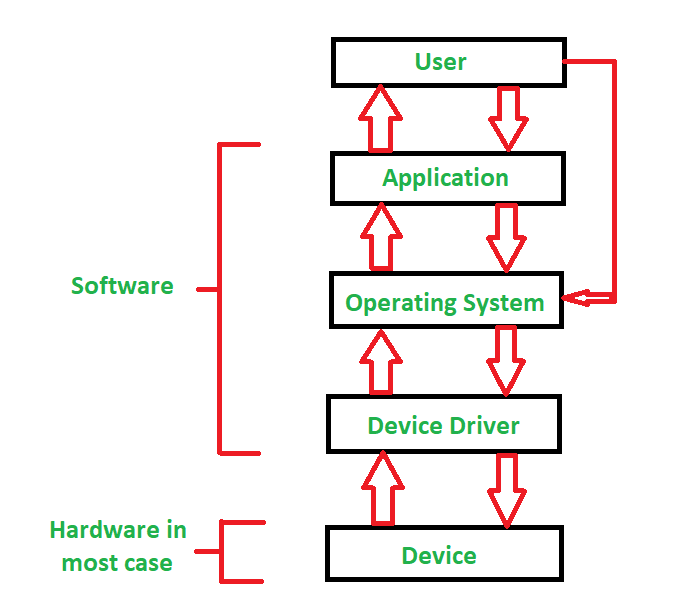
**Device Driver** in computing refers to a special kind of software program or a specific type of software application which controls a specific hardware device that enables different hardware devices for communication with the computer’s Operating System

A device driver communicates with the computer’s hardware by computer subsystem or computer bus connected to the hardware.

**Device Drivers** are very essential for a computer system to work properly because without device driver the particular hardware fails to work accordingly means it fails in doing a particular function/action for which it has been created.

In a very common way most term it as only a **Driver** also when someone says **Hardware Driver** that also refers to this **Device Driver.**

**Working of Device Driver :**  
Device Drivers depend upon the Operating System’s instruction to access the device and performing any particular action. After the action they also shows their reactions by delivering output or status/message from hardware device to the Operating system.For Example a printer driver tells the printer in which format to print after getting instruction from OS, similarly A sound card driver is there due to which 1’s and 0’s data of MP3 file is converted to audio signals and you enjoy the music. Card reader, controller, modem, network card, sound card, printer, video card, USB devices, RAM, Speakers etc need Device Drivers to operate.



1. **What is a kernel. Explain the type of Kernel briefly**

**Kernel** is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

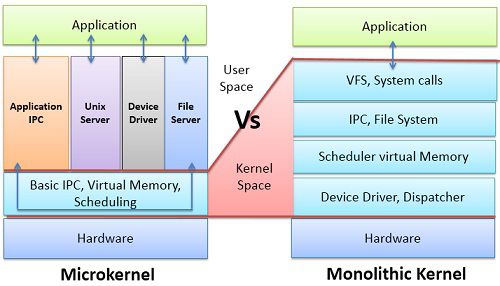
Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again. It is responsible for various tasks such as disk management, task management, and memory management.

It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute. It basically acts as an interface between user applications and hardware. The major aim of kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.

**There are five types of kernels:**

* A **micro** kernel, which only contains basic functionality;
* A monolithic kernel, which contains many device drivers.
* Hybrid Kernel.
* Exo Kernel.
* Nano Kernel.

1. **With a neat diagram explain the concept of Monolithic vs Microkernel**



| **BASIS FOR COMPARISON** | **MICROKERNEL** | **MONOLITHIC KERNEL** |
| --- | --- | --- |
| Basic | In microkernel user services and kernel, services are kept in separate address space. | In monolithic kernel, both user services and kernel services are kept in the same address space. |
| Size | Microkernel are smaller in size. | Monolithic kernel is larger than microkernel. |
| Execution | Slow execution. | Fast execution. |
| Extendible | The microkernel is easily extendible. | The monolithic kernel is hard to extend. |
| Security | If a service crashes, it does effect on working of microkernel. | If a service crashes, the whole system crashes in monolithic kernel. |
| Code | To write a microkernel, more code is required. | To write a monolithic kernel, less code is required. |
| Example | QNX, Symbian, L4Linux, Singularity, K42, Mac OS X, Integrity, PikeOS, HURD, Minix, and Coyotos. | Linux, BSDs (FreeBSD, OpenBSD, NetBSD), Microsoft Windows (95,98,Me), Solaris, OS-9, AIX, HP-UX, DOS, OpenVMS, XTS-400 etc. |

1. **What is virtualization? How virtualization helps? Give the command to check the support of virtualization by a system.**

**Virtualization** is the process of creating a software-based, or virtual, representation of something, such as virtual applications, servers, storage and networks. It is the single most effective way to reduce IT expenses while boosting efficiency and agility for all size businesses. **Virtualization** Basics.

**Virtualization** can increase IT agility, flexibility and scalability while creating significant cost savings. Greater workload mobility, increased performance and availability of resources, automated operations – they're all benefits of **virtualization** that make IT simpler to manage and less costly to own and operate.

1. Use Task Manager To Check CPU Virtualization

To check if you have virtualization technology support on your processor, open Task Manager using **CTRL**+ **SHIFT**+ **ESC**. Now, if your processor supports virtualization, you will find it mentioned where other details are shown.

Under Performance Tab :

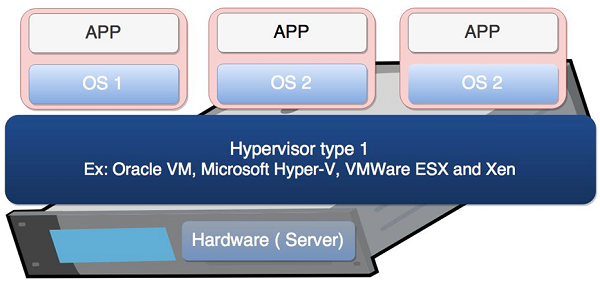
**Enabled** mentioned or else **Disabled**. And if your processor doesn’t support virtualization technology, you will not see Virtualization and Hyper-V mentioned anywhere at all.

1. **Bare metal and hosted visualization**

Native or Bare Metal Hypervisor

Native hypervisors are software systems that run directly on the host's hardware to control the hardware and to monitor the **Guest Operating Systems**. The guest operating system runs on a separate level above the hypervisor. All of them have a Virtual Machine Manager.

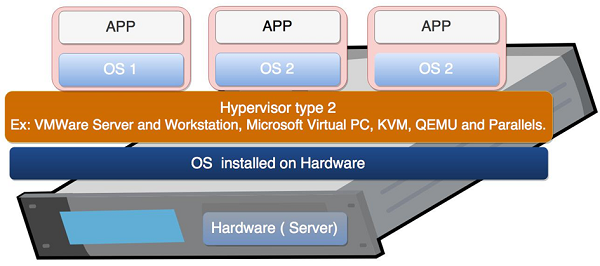
Examples of this virtual machine architecture are **Oracle VM, Microsoft Hyper-V, VMWare ESX** and **Xen**.



Hosted Hypervisor

Hosted hypervisors are designed to run within a traditional operating system. In other words, a hosted hypervisor adds a distinct software layer on top of the host operating system. While, the guest operating system becomes a third software level above the hardware.

A well-known example of a hosted hypervisor is **Oracle VM VirtualBox**. Others include **VMWare Server and Workstation, Microsoft Virtual PC, KVM, QEMU** and **Parallels**.



**A****[hypervisor](https://www.vmware.com/topics/glossary/content/hypervisor.html?SRC=WWW_US_GP_bare-metal-hypervisor_SiteLink" \t "_blank), also known as a virtual machine monitor or VMM, is a type of virtualization software that supports the creation and management of**[**virtual machines**](https://www.vmware.com/topics/glossary/content/virtual-machine)**(VMs) by separating a computer’s software from its hardware. Hypervisors translate requests between the physical and virtual resources, making virtualization possible. When a hypervisor is installed directly on the hardware of a physical machine, between the hardware and the operating system (OS), it is called a bare metal hypervisor. Some bare metal hypervisors are embedded into the firmware at the same level as the motherboard basic input/output system (BIOS). This is necessary for some systems to enable the operating system on a computer to access and use virtualization software.**

**Because the bare metal hypervisor separates the OS from the underlying hardware, the software no longer relies on or is limited to specific hardware devices or drivers.  This means bare metal hypervisors allow operating systems and their associated applications to run on a variety of types of hardware. They also allow multiple operating systems and virtual machines (guest machines) to reside on the same physical server (host machine). Because the virtual machines are independent of the physical machine, they can move from machine to machine or platform to platform, shifting workloads and allocating networking, memory, storage, and processing resources across multiple servers according to needs. For example, when an application needs more processing power, it can seamlessly access additional machines through the virtualization software. This results in greater cost and energy efficiency and better performance, using fewer physical machines.**

**The term bare metal refers to the fact that there is no operating system between the virtualization software and the hardware. The virtualization software resides on the “bare metal” or the hard disk of the hardware, where the operating system is usually installed.**

**Bare metal isn’t only used to describe hypervisors. A bare metal server is a regular, single-tenant server. However, it can be a host machine for virtual machines with the addition of a hypervisor and virtualization software. A bare metal cloud refers to a customer renting the actual servers that host the****[public cloud](https://www.vmware.com/topics/glossary/content/public-cloud.html?SRC=WWW_US_GP_bare-metal-hypervisor_SiteLink" \t "_blank) from a cloud service provider, in addition to renting the public cloud services**.

**The bare metal hypervisor is the most commonly deployed type of hypervisor. This is where the virtualization software is installed directly on the hardware, where the operating system is normally installed. Bare metal hypervisors are extremely secure since they are isolated from the attack-prone operating system. They perform better and more efficiently than hosted hypervisors, and most companies choose bare metal hypervisors for enterprise and**[**data center**](https://www.vmware.com/topics/glossary/content/data-center)**computing needs.**

**There is another type of hypervisor, known as a client or hosted hypervisor. While bare metal hypervisors run directly on the computing hardware, hosted hypervisors run within the operating system of the host machine. Although hosted hypervisors run within the OS, additional operating systems can be installed on top of it. Hosted hypervisors have higher latency than bare metal hypervisors because requests between the hardware and the hypervisor must pass through the extra layer of the OS. Hosted hypervisors are also known as client hypervisors because they are most often used with end users and software testing, where the higher latency is not as much of a concern.**

**Hardware acceleration technology can boost processing speed for both bare metal and hosted hypervisors by doing some of the resource-intensive work of creating and managing virtual resources. A virtual Dedicated Graphics Accelerator (vDGA) is a type of hardware accelerator that can take care of sending and refreshing high-end 3-D graphics, freeing up the main system for other tasks and greatly increasing the speed at which images are displayed. This technology is very useful for industries such as oil and gas exploration, where companies need to quickly visualize complex data.**

**A bare metal server is a dedicated server with a single tenant, controlled by a single client. For bare metal virtualization, that client can install a bare metal hypervisor directly onto the hardware of that server to enable virtualization technology including virtual applications, multiple**[**virtual machines**](https://www.vmware.com/topics/glossary/content/virtual-machine.html?SRC=WWW_US_GP_bare-metal-hypervisor_SiteLink)**, or**[**private clouds**](https://www.vmware.com/topics/glossary/content/private-cloud.html?SRC=WWW_US_GP_bare-metal-hypervisor_SiteLink)**.**

**Hosted hypervisors, in contrast, are installed on top of the operating system, not on the bare metal of the server. Both types of hypervisors can run multiple virtual servers for multiple tenants on one physical machine. Public cloud service providers lease server space on the different virtual servers to different companies. So one server can host several virtual servers that are running workloads for multiple companies. The sharing of resources can pose more of a security risk than a dedicated bare metal server and can result in a “noisy neighbor” effect when one of the tenants runs a large workload that interferes with the server performance for other tenants.**

**Since a single company has full control over a bare metal server, that server will always provide higher performance than a virtual server that is sharing a physical server’s bandwidth, memory, and processing power with other virtual servers. Companies also have more control over the hardware for bare metal servers and can optimize them to increase performance. Because bare metal servers do not share resources with other tenants, they are often used to host private clouds, especially for companies that need to comply with regulations that require physical separation of resources.**

**Hypervisors hosting multiple virtual machines do offer some advantages over bare metal servers. Hypervisors allow virtual machines to be created instantly, providing more resources as needed for dynamic workloads. It is much harder to provide an additional physical server when it is needed. Hypervisors also allow for more utilization of a physical server, since it is able to run several virtual machines on one physical machine’s resources. Running several virtual machines on one physical machine is more cost and energy efficient than running multiple underutilized physical machines for the same task**

1. **Types of visualization**

* Server Virtualization
* Client & Desktop Virtualization
* Services and Applications Virtualization
* Network Virtualization
* Storage Virtualization

Server Virtualization

It is virtualizing your server infrastructure where you do not have to use any more physical servers for different purposes.

### **Client & Desktop Virtualization**

This is similar to server virtualization, but this time is on the user’s site where you virtualize their desktops. We change their desktops with thin clients and by utilizing the datacenter resources.

### **Services and Applications Virtualization**

The virtualization technology isolates applications from the underlying operating system and from other applications, in order to increase compatibility and manageability. For example – Docker can be used for that purpose.

### **Network Virtualization**

It is a part of virtualization infrastructure, which is used especially if you are going to visualize your servers. It helps you in creating multiple switching, Vlans, NAT-ing, etc.

### **Storage Virtualization**

This is widely used in datacenters where you have a big storage and it helps you to create, delete, allocated storage to different hardware. This allocation is done through network connection. The leader on storage is SAN.

1. **Data types in c**
2. **Primary data types**: These are fundamental data types in C namely integer(int), floating point(float), character(char) and void.

* **int** - integer: a whole number.
* **float** - floating point value: ie a number with a fractional part.
* **double** - a double-precision floating point value.
* **char** - a single character.
* **void** - valueless special purpose type which we will examine closely in later sections.

1. **Derived data types**: Derived data types are nothing but primary datatypes but a little twisted or grouped together like **array**, **structure**, **union** and **pointer**.

* **array -**An array is a collection of data items, all of the same type, accessed using a common name.
* **structure -** A structure in C is a collection of items of different types. You can think of a structure as a "record" is in Pascal or a class in Java without methods.
* **union -**A union is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes.
* **pointer-**A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address.

C program

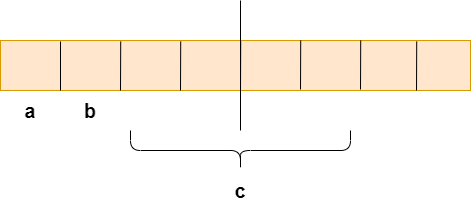
1. #include <stdio.h>
2. #include <conio.h>
3. main()
4. {
5. clrscr();
6. printf(" short int is %2d bytes \n", sizeof(short int));
7. printf(" int is %2d bytes \n", sizeof(int));
8. printf(" int \* is %2d bytes \n", sizeof(int \*));
9. printf(" long int is %2d bytes \n", sizeof(long int));
10. printf(" long int \* is %2d bytes \n", sizeof(long int \*));
11. printf(" signed int is %2d bytes \n", sizeof(signed int));
12. printf(" unsigned int is %2d bytes \n", sizeof(unsigned int));
13. printf("\n");
14. printf(" float is %2d bytes \n", sizeof(float));
15. printf(" float \* is %2d bytes \n", sizeof(float \*));
16. printf(" double is %2d bytes \n", sizeof(double));
17. printf(" double \* is %2d bytes \n", sizeof(double \*));
18. printf(" long double is %2d bytes \n", sizeof(long double));
19. printf("\n");
20. printf(" signed char is %2d bytes \n", sizeof(signed char));
21. printf(" char is %2d bytes \n", sizeof(char));
22. printf(" char \* is %2d bytes \n", sizeof(char \*));
23. printf("unsigned char is %2d bytes \n", sizeof(unsigned char));
24. getch();
25. }
26. **What is padding of byte. Explain in C how the padding works in struct.**

In order to align the data in memory, one or more empty bytes (addresses) are inserted (or left empty) between memory addresses which are allocated for other structure members while memory allocation. This concept is called structure padding.

Architecture of a computer processor is such a way that it can read 1 word (4 byte in 32 bit processor) from memory at a time. To make use of this advantage of processor, data are always aligned as 4 bytes package which leads to insert empty addresses between other member’s address.

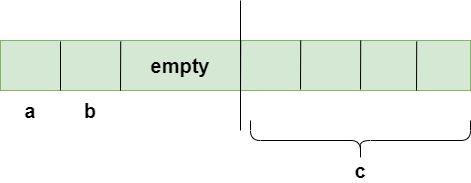
1. **struct** student
2. {
3. **char** a; // 1 byte
4. **char** b; // 1 byte
5. **int** c; // 4 bytes
6. }

If we have a 32-bit processor (4 bytes at a time), then the pictorial representation of the memory for the above structure would be:



As we know that structure occupies the contiguous block of memory as shown in the above diagram, i.e., 1 byte for char a, 1 byte for char b, and 4 bytes for int c. Both **char a** and **char b**  variables can be accessed in one [CPU](https://www.javatpoint.com/cpu-full-form) cycle, but we will face the problem when we access the **int c** variable as 2 CPU cycles are required to access the value of the 'c' variable. So in the first cycle two bytes and in the second cycle, the other two bytes are accessed.

To overcome this problem, structure padding is used.



In order to achieve the structure padding, an empty row is created on the left, as shown in the above diagram, and the two bytes which are occupied by the 'c' variable on the left are shifted to the right. So, all the four bytes of 'c' variable are on the right. Now, the 'c' variable can be accessed in a single CPU cycle. After structure padding, the total memory occupied by the structure is 8 bytes (1 byte+1 byte+2 bytes+4 bytes), which is greater than the previous one. Although the memory is wasted in this case, the variable can be accessed within a single cycle.

1. **What is endianness? Explain the types of endianness with neat examples**

Endianness refers to the order of storing and reading multi-byte words in memory. Endianness determines if the least significant byte of a word that we want to store in memory will go to lowest or the highest address of the assigned memory space.

There are two possibilities of Endianness: Little Endian and Big Endian

A big-endian system stores the most significant byte of a word at the smallest memory address and the least significant byte at the largest.

Diagram

Description automatically generated

A little-endian system, in contrast, stores the least-significant byte at the smallest address.

Diagram

Description automatically generated

1. **List the features of Windows, Mac and Linux Operating SYstems. Compare and Contrast.**

|  |  |  |  |
| --- | --- | --- | --- |
| **The basis of Comparison** | **Windows** | **MAC** | **Linux** |
| Basic difference and history | Windows was first released in 1985. It was supposed to be a graphical user interface on top of MS-DOS. All features of MS-DOS were later integrated with Windows 95 release. It was a huge success in and led to Windows transition. | This operating system from Apple stands older than Windows. It was first released in 1984. It began as a graphical user interface right from its inception. In 2005 the design and structure of MAC OS were changed to Intel x86 based architecture. | It was initially developed in Finnish University. It was released in 1991 and designed for GNU developers. GNU developers later integrated it into Linux. It is open to consumers and everyone can use as per their specifications. |
| File structure | Windows follows a directory structure to store the different kinds of files of the user. It has logical drives and cabinet drawers. It also has folders. Some common folders like documents, pictures, music, videos, and downloads. All these files can be stored in these folders and also new folders can be created. It also has files which can be a spreadsheet or an application program. It can have extensions as .txt, .jpg etc.  In addition to this Windows also provides recycle bin where all deleted files can be stored. Recycle bin can be configured to increase its size. | The file structure of MAC is commonly known as MAC OS X. If you go to dig into your MAC’s hard disk through finder you will see many directories. The root directory of MAC may encounter when they visit their own MAC book. You can explore the file system and directory structure by going to directories like /Application, /Developer, /sbin, /tmp, etc. | Linux has a completely different file structure form Windows and MAC. It was developed with a different code base. It stores data in the form of a tree. There is a single file tree and all your drives are mounted over this tree. |
| Registry | Windows registry is a master database which is used to store all settings on your computer. It is responsible to store all user information with its passwords and device relate information. The registry also has an editor which allows you to view all keys and values or even drivers if necessary. | MAC stores all application settings in a series of .plist files which have the various preferences folder in MAC. This .plist file contains all properties in either plain text or binary format. These are stored at:  /Library/Preferences folder | Linux also does not have a specific registry of its own. All application setting is stored on program basis under the different users in the same hierarchy format of the files being stored. There is no centralized database for storing these details and so periodic cleaning is also not required. |
| Interchangeable Interfaces | Windows interface was not interchangeable until Windows 8. Windows XP had some improvements but not par. Start menu, taskbar, system tray, and Windows Explorer. | MAC has a facility to bridge virtual network interfaces. This can be done by going to system preferences and managing the interfaces. | Linux is easy to switch interfaces. You can switch the environment without having to carry all installations. There are utilities like GNOME and KDE which help in catering to these needs. They help in focusing on different aspects. |
| Command terminal | A terminal or command prompt is a black box ideally used to execute commands. It is also called the Windows Command Processor. It is used to execute commands and different batch files. It can also be used for administrative functions and troubleshoot and solve all windows issues. | MAC provides a console as a terminal application. It has a console, command line, prompt and terminal. Command line is used to type your commands. Prompt will provide you with some information and also enable you to run commands. A terminal is an actual interface which will provide the modern graphical user interface as well.  You can find terminal at Applications -> Utilities. | Linux also provides a terminal. You can find terminal at: Applications -> System or Applications -> Utilities. In addition to this, there is also a shell prompt. The most common shell used in bash. It defines how the terminal will behave and look when it is run. |

1. **What is distributed operating system. Explain in detail.**

**Distributed Operating System** is one of the important type of operating system.

Multiple central processors are used by Distributed systems to serve multiple real-time applications and multiple users. Accordingly, Data processing jobs are distributed among the processors.

Processors communicate with each other through various communication lines (like high-speed buses or telephone lines). These are known as **loosely coupled systems** or distributed systems. Processors in this system may vary in size and function. They are referred as sites, nodes, computers, and so on.

**High-speed buses:**

A mechanism that transfers data between components inside a computer.

A DOS involves a collection of autonomous computer systems, which are able to communicate with each other through LAN/WAN. This system will provide a virtual machine abstraction to its users and wide sharing of resources like computational capacity and input/output etc.

This system as mentioned above, incorporates various autonomous interconnected computers that communicate with each other using a shared communication network, furthermore they are independent systems that possess their own memory unit and CPU.

Another term which is used along side-distributed operating systems is a loosely coupled system. The processors in these systems may differ in size and function.

One of the big advantages of working with DOS is that it is always possible that one user can access the files or software, which they require, and utilise them, however in reality these files are present on another system network (so think of it similar to remote working).

Distributed systems can be considered to be more reliable than a central system because if the system has only one instance of a critical peripheral/component, like the CPU, network interface, disk, and so if that one instance fails, the system will go down completely.

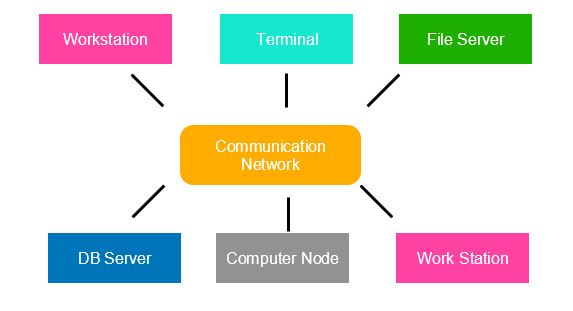
However when there are multiple instances in the system, like in a distributed operating system, then if a component fails, the system may be able to continue to function despite the failure. Distributed systems also allow software failures to be dealt with, rather than stopping the whole system.

Architecture of a Distributed Operating System:

In a DOS the following occurs:

* All software and hardware compounds are located remotely. In order for them to communicate with each other, they pass messages.
* One of the most important aspects of a distributed system is resource sharing. Resources are managed by servers and clients use these resources.

A DOS runs on a number of independent sites which are connected through a communication network. However it is portrayed to the user that they run their own operating system.



1. **What is the difference between gets vs scanf vs fgets. Justify why gets is dangerous to use.**

**scanf() function in c**:

scanf() works great while taking inputs such as integer, character, float etc. It certainly falls behind while taking string inputs containing whitespaces. Let’s take a look at an example,

|  |
| --- |
| #include<stdio.h>  int main()  {      char string[10];      printf("Enter the string: ");      scanf("%s", string);      printf("\n %s",string);      return 0;  } |

  Scanf () stops scanning as soon as it encounters whitespace or newline. This, in fact, makes taking string inputs using scanf() a bit troublesome. This can be easily avoided by using some other input functions like gets() and fgets().

## gets() function in C

## it is a pre-defined function in C which is used to read a string or a text line. And store the input in a well-defined string variable. The function terminates its reading session as soon as it encounters a newline character.

Example:

#include<stdio.h>

int main()

{

    char string[10];

    printf("Enter the String: ");

    gets(string);

    printf("\n%s",string);

    return 0;

}

Compare the output with the one while using scanf(). The whole output is now treated as a single string.

## fgets() function in C

The standard C library also provides us with yet another function, the fgets() function. The function reads a text line or a string from the specified file or console. And then stores it to the respective string variable.

Similar to the gets() function, fgets also terminates reading whenever it encounters a newline character. But furthermore, unlike gets(), the function also stops when EOF is reached or even if the string length exceeds the specified limit, n-1.

Syntax,

fgets(char \*str, int n, FILE \*stream)

* str – It is the variable in which the string is going to be stored
* n – It is the maximum length of the string that should be read
* stream – It is the filehandle, from where the string is to be read.

Fortunately, we can both read text lines from a file or the standard input stream by using the fgets() function. Let us see how

For example

#include<stdio.h>

int main()

{

    char string[20];

    FILE \*fp;

    fp=fopen("file.txt","r");

    fgets(string,20,fp);

    printf("The string is: %s",string);

    fclose(fp);

    return 0;

}

Even though both the functions, gets() and fgets() can be used for reading string inputs. The biggest difference between the two is the fact that the latter allows the user to specify the buffer size. Hence it is highly recommended over the gets() function.

The gets() function doesn’t have the provision for the case if the input is larger than the buffer. As a result, memory clogging may occur. This is the part where the fgets() function shines and provides an ultimate solution.

1. **With a neat example explain the handling of string in C language.**

In C programming, a string is a sequence of characters terminated with a null character \0. For example:

char c[] = "c string";

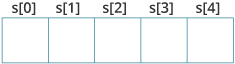
When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

Memory diagram of strings in C programming

## Declaration of strings:

Here's how you can declare strings:

char s[5];



Here, we have declared a string of 5 characters.

## Initialization of strings:

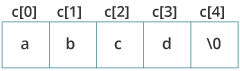
You can initialize strings in a number of ways.

char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0'};



Let's take another example:

char c[5] = "abcde";

Here, we are trying to assign 6 characters (the last character is '\0') to a char array having 5 characters. This is bad and you should never do this.

## Assigning Values to Strings

Arrays and strings are second-class citizens in C; they do not support the assignment operator once it is declared. For example,

char c[100];

c = "C programming"; // Error! array type is not assignable.

**Note:**Use the [strcpy() function](https://www.programiz.com/c-programming/library-function/string.h/strcpy" \o "C strcpy) to copy the string instead.

## Read String from the user

You can use the scanf() function to read a string.

The scanf() function reads the sequence of characters until it encounters [whitespace](https://stackoverflow.com/questions/30033582/what-is-the-symbol-for-whitespace-in-c) (space, newline, tab, etc.).

### Example 1: scanf() to read a string

#include <stdio.h>

int main()

{

char name[20];

printf("Enter name: ");

scanf("%s", name);

printf("Your name is %s.", name);

return 0;

}

.

Even though *Dennis Ritchie* was entered in the above program, only *"Dennis"* was stored in the *name* string. It's because there was a space after *Dennis*.

### How to read a line of text?

we can use the fgets() function to read a line of string. And, you can use puts() to display the string.

### Example 2: fgets() and puts()

#include <stdio.h>

int main()

{

char name[30];

printf("Enter name: ");

fgets(name, sizeof(name), stdin); // read string

printf("Name: ");

puts(name); // display string

return 0;

}

Here, we have used fgets() function to read a string from the user.

fgets(name, sizeof(name), stdlin); // read string

The sizeof(name) results to 30. Hence, we can take a maximum of 30 characters as input which is the size of the *name* string.

To print the string, we have used puts(name);.

.  
  
It's because gets() allows you to input any length of characters. Hence, there might be a buffer overflow.

## Passing Strings to Functions

Strings can be passed to a function in a similar way as arrays. Learn more about [passing arrays to a function](https://www.programiz.com/c-programming/c-arrays-functions).

### Example 3: Passing string to a Function

#include <stdio.h>

void displayString(char str[]);

int main()

{

char str[50];

printf("Enter string: ");

fgets(str, sizeof(str), stdin);

displayString(str); // Passing string to a function.

return 0;

}

void displayString(char str[])

{

printf("String Output: ");

puts(str);

}

1. **Why the length of string is +1 in fgets.**

**fgets**() reads in at most **one** less than **size** characters from stream and stores them into the buffer pointed to by s. Reading stops after an EOF or a newline. ... A terminating null byte (aq\0aq) is stored after the last character in the buffer. So it adds '\n' after your 4 letters, returning string\_length+**1**

**eg**

* 1. #include <stdio.h>
  2. #include <stdlib.h>
  3. #include <string.h>
  4. int main(void)
  5. {
  6. char name[10] = { 0 };
  7. printf("enter your name\n");
  8. fgets(name, 10, stdin);
  9. printf("your name is %s and it is %d letters\n", name, strlen(name)); // length problem
  10. name[strcspn(name, "\n")] = 0;
  11. printf("NEW - your name is %s and it is %d letters\n", name, strlen(name));
  12. return 0;
  13. }

1. **List string functions available in C language. Give examples.**
2. **strlen( )**function is used to find the length of a character string.

**Example:**      **int  n;**

**char st[20] = “Bangalore”;**

**n = strlen(st);**

**2) strcpy( ) Function :**

**strcpy( )** function copies contents of one string into another string. Syntax for strcpy function is given below.

**Syntax:  char \* strcpy (char \* destination, const char \* source);**

**Example:**

**strcpy ( str1, str2) – It copies contents of str2 into str1.**

**Strcpy ( str2, str1) – It copies contents of str1 into str2.**

**3) strcat( ) Function :**

**strcat( )**function in C language concatenates two given strings. It concatenates source string at the end of destination string. Syntax for **strcat( )**function is given below.

**Syntax :  char \* strcat ( char \* destination, const char \* source );**

#### 4) Strncat() function :

**strncat( )** function in C language concatenates (appends) portion of one string at the end of another string.

**Syntax : char \* strncat ( char \* destination, const char \* source, size\_t num );**

**Example:  
strncat ( str2, str1, 3 ); – First 3 characters of str1 is concatenated at the end of str2.**

#### 5) strcmp( ) Function :

**strcmp( )** function in C compares two given strings and returns zero if they are same. If length of string1 < string2, it returns < 0 value. If length of string1 > string2, it returns > 0 value.

**Syntax : int strcmp ( const char \* str1, const char \* str2 );**

#### 7) strlwr() function :

**strlwr()** function converts a given string into lowercase.

**Syntax : char \*strlwr(char \*string);**

**strlwr()** function is non standard function which may not available in standard library in C.

#### 8) strupr() function :

**strupr()** function converts a given string into uppercase.

**Syntax : char \*strupr(char \*string);**

**strupr()** function is non standard function which may not available in standard library in C.

#### 9) strrev() function :

**strrev()** function reverses a given string in C language.

**Syntax : char \*strrev(char \*string);**

**strrev()** function is non standard function which may not available in standard library in C.

**Example:  
char name[20]=”ftl”; then  
strrev(name)= ltf**

#### 12) atoi() function :

It converts string-value to numeric-value and it converts a numeric-string value to equivalent integer-value.

**Syntax : int atoi(string);**

**Example :  
printf(“output=%d”, atoi(“123”)+atoi(“234”));  
This printf() will print 357**

#### 13) toll() function :

converts a long int string value to equivalent long integer value.

**Syntax : long int toll(string);**

**Example :  
printf(“output=%d”, toll(“486384”)-atol(“112233”));  
This statement will print 374151**

#### 14) atof() function :

converts a floating point text format value to double value.

**Syntax : int atoi(string);**

**Example :  
printf(“%1f”,atof(“3.1412”)\*5\*5);**

#### 10) strchr() function :

**strchr()** function returns pointer to the first occurrence of the character in a given string.

**Syntax : char \*strchr(const char \*str, int character);**

**Example :  
char city[20] = “Madras”;  
char town[20] = “Mangalore”;  
strcmp(city, town);**