**UNIT-1**

1. **What is an Operating System?**

**Ans:-** An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks.

The OS helps you to communicate with the computer without knowing how to speak the computer’s language. It is not possible for the user to use any computer or mobile device without having an operating system.

### Features of Operating SystemFeatures of Operating system:-

* + Provides a platform for running applications
  + Handles Memory management and CPU scheduling
  + Provides File system abstraction and Networking Support
  + Provides security features and user interface
  + Provides utilities and system services
  + Handling I/O operations
  + Supports application development
  + Error Detection and handling
  + Resource allocation

### What is Operating System ? Explain Advantages and Dis-Aadvantages?

**Ans:-** An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks.

### Advantages :-

* + Allows you to hide details of hardware by creating an abstraction
  + Easy to use with a GUI
  + It Offers an environment in which a user may execute programs/applications
  + Operating System acts as an intermediator among applications and the hardware components
  + It provides the computer system resources with easy to use format
  + Acts as an intermediator between all hardware’s and software’s of the system

### Dis - Advantages:-

* If any issue occurs in OS, you may lose all the contents which have been stored in your system
* Operating system’s software is quite expensive for small size organization which adds burden on them. **Ex:-** Windows
* It is never entirely secure as a threat can occur at any time.

### Explain History and Evolution of OS ?

**Ans :-** An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs.

### History of OS:-

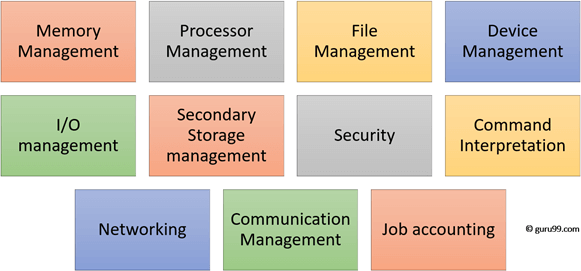
* Operating systems were first developed in the late 1950s to manage tape storage
* The General Motors Research Lab implemented the first OS in the early 1950s(1956) for their IBM 701
* In the mid-1960s, operating systems started to use disks
* In the late 1960s, the first version of the Unix OS was developed
* The first OS built by Microsoft was DOS. It was built in 1981 by purchasing the 86-DOS software from a Seattle company
* The present-day popular OS Windows first came to existence in 1985 when a GUI was created and paired with MS-DOS.

**Ex:-** Latest Operating systems are :

Windows, Android , iOS , Mac OS, Linux , Chrome OS , Windows Phone OS.

### Explain about Functions of Operating System ?

Ans :- Operating system functions may include managing memory, files, processes, I/O system & devices, security. In an Operating system software performs each of the function.



1. **Process management**:- Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.
2. **Memory management:-** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.
3. **File management**:- It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.
4. **Device Management**:- Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.
5. **I/O System Management:-** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.
6. **Secondary-Storage Management**:- Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.
7. **Security**:- Security module protects the [data and information](https://www.guru99.com/difference-information-data.html) of a computer system against malware threat and authorized access.
8. **Command interpretation**:- This module is interpreting commands given by the and acting system resources to process that commands.
9. **Networking:-** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.
10. **Job accounting**:- Keeping track of time & resource used by various job and users.
11. **Communication management**:- Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

### What is Operating System ? Explain types of Operating systems ?

**Ans:-** An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks.

### Types of Operating System (OS):-

Following are the popular types of OS (Operating System):

* Batch Operating System
* Multitasking/Time Sharing OS
* Multiprocessing OS
* Real Time OS
* Distributed OS
* Network OS
* Mobile OS
* **Batch Operating System :-** Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group.

The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer operator.

* **Multi-Tasking/Time-sharing Operating systems**:- Time-sharing operating system enables people located at a different terminal(shell) to use a single computer system at the same time. The processor time (CPU) which is shared among multiple users is termed as time sharing.
* **Real time OS:-** A real time operating system time interval to process and respond to inputs is very small.

Ex:- Military Software Systems, Space Software Systems are the Real time OS example.

* **Distributed Operating System:-** Distributed systems use many processors located in different machines to provide very fast computation to its users.
* **Network Operating System:-**Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions.
* **Mobile OS:-** Mobile operating systems are those OS which is especially that are designed to power smart phones, tablets devices.

Some most famous mobile operating systems are Android and iOS, but others include BlackBerry, Web, and watch OS.

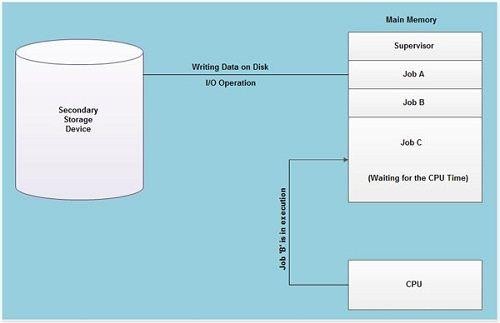
### Discuss about Multi Programming Systems?

**Ans:-** An operating system that is capable of running multiple programs on a single processor is known as a multiprogramming operating system. If a program has to

wait for an I/O transfer in a multiprogramming operating system, other programs utilize the CPU and other resources meanwhile.

One of the aims of multiprogramming is to manage the various resources of the entire system. Multiprogramming is the principle concept used by multiprogramming systems. These resources include, but are not restricted to, the file system, memory, processors, input and output, etc.

Multiprogramming operating systems, therefore, are designed for storing and processing several programs simultaneously. The routines responsible for managing resources are made accessible to the core functions of the operating system.



**Advantages :-**

1. Processor is utilized most of the time and rarely becomes idle unless there are no jobs to execute.
2. The system is fast because all the jobs run parallel amongst themselves.
3. Multiprogramming operating systems support multiple users on the computer system.
4. Total time required to execute a job reduced.
5. Multiprogramming operating systems are good at dealing with applications in which multiple jobs are to be executed.

### Disadvantages:-

1. Sometimes, processes requiring long CPU times have to wait for other jobs to finish.
2. It's not easy keeping track of a large number of processes in multiprogramming.
3. Multiprogramming operating systems have to use CPU scheduling.
4. Memory management should be very efficient.
5. While a program executes, there cannot be any interaction between it and the user.

### Explain about Time Sharing Systems ?

**Ans:-** A time sharing system allows many users to share the [computer](https://ecomputernotes.com/fundamental/introduction-to-computer/what-is-computer) resources simultaneously. Time sharing refers to the allocation of [computer](https://ecomputernotes.com/fundamental/introduction-to-computer/what-is-computer) resources in time slots to several programs simultaneously. For example a [mainframe](https://ecomputernotes.com/fundamental/introduction-to-computer/mainframe) computer that has many users logged on to it. Each user uses the resources of the [mainframe](https://ecomputernotes.com/fundamental/introduction-to-computer/mainframe) -

i.e. [memory](https://ecomputernotes.com/fundamental/input-output-and-memory/memory), [CPU](https://ecomputernotes.com/fundamental/introduction-to-computer/what-is-cpu) .

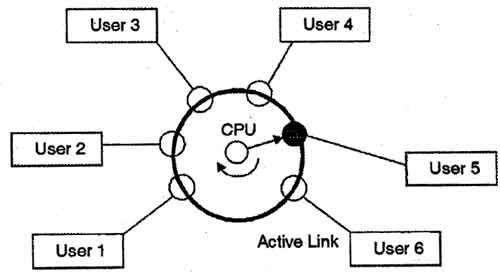
A time shared system uses CPU scheduling and multi-programming to provide each user with a small portion of a time-shared computer. It allows many users to share the computer resources simultaneously. As the system switches rapidly from one user to the other, a short time slot is given to each user for their executions.

The time sharing system provides the direct access to a large number of users where CPU time is divided among all the users on scheduled basis. The OS allocates a set of time to each user. When this time is expired, it passes control to the next user on the system. The time allowed is extremely small and the users are given the impression that they each have their own CPU and they are the sole owner of the CPU.

The concept of time sharing system is shown in figure. In above figure the user 5 is active but user 1, user 2, user 3, and user 4 are in waiting state whereas

user 6 is in ready status.

As soon as the time slice of user 5 is completed, the control moves on to the next ready user i.e. user 6. In this state user 2, user 3, user 4, and user 5 are in waiting state and user 1 is in ready state. The process continues in the same way and so on.



The time-shared systems are more complex than the multi-programming systems. In time-shared systems multiple processes are managed simultaneously which requires an adequate management of main [memory](https://ecomputernotes.com/fundamental/input-output-and-memory/memory) so that the processes can be swapped in or swapped out within a short time.

### Discuss about Batch Systems ?

**Ans:-** To avoid the drawbacks of early systems the batch processing systems were introduced. The early systems required more setup time.

To overcome this drawback, batch processing is used where the more set up time was reduced by processing the jobs in batches in this approach similar jobs were sent to the CPU for processing and were run together.

The important task of the batch processing system is to automatically keep executing the job in batches to reduce the setup time.

In the early job processing systems, the jobs were placed in a job queue and the memory allocated or managed by the primary memory, when space was

available in the main memory, a job was selected from the job queue and was loaded into memory.

Once the job is loaded into primary memory, it completes for the processor. When the processor becomes available, the processor scheduler selects a job that was loaded in the memory and executes it.

The batch strategy is implemented to provide a batch file processing. It follows that files of the similar batch are processed to speed up the task.

**Steps in Batch Operating System:-**

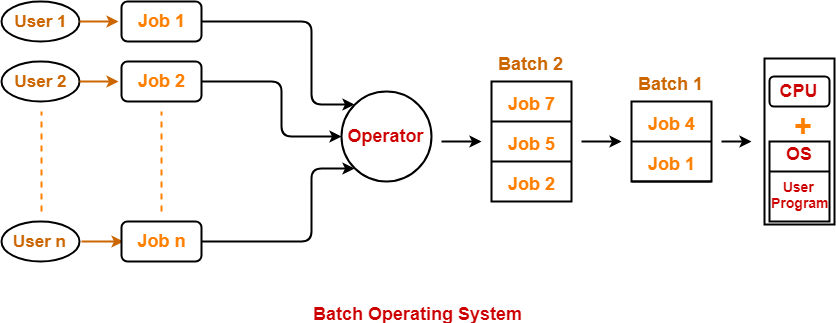
**Step 1** − Using punch cards the user prepares his job.

**Step 2** − After that the user submits the job to the programmer.

**Step 3** − The programmer collects the jobs from different users and sorts the jobs into batches with similar needs.

**Step 4** − Finally, the programmer submits the batches to the processor one by one.

**Step 5** − All the jobs of a single batch are executed together.



### Advantages

The advantages of batch operating system are as follows −

* The time taken by the system to execute all the programs will be reduced.
* It can be shared between multiple users.

### Disadvantages

The disadvantages of batch operating system are as follows −

* Manual interrupts are required between two batches.
* Priority of jobs is not set, they are executed sequentially.
* It may lead to starvation.
* The CPU utilization is low and it has to remain ideal for a long time because the time taken in loading and unloading of batches is very high as compared to execution time.

### What is PC Operating System? Explain Operating Systems for Personal Computers ?

**Ans:-** Personal computer operating system provides a good interface to a single user. Personal computer operating systems are widely used for word processing, spreadsheets and Internet access.

Personal computer operating system are made only for personal. You can say that your laptops, computer systems, tablets... are your personal computers and the operating system such as windows 7, windows 10, android, are your personal computer operating system and you can use your personal computer operating system for your personal purposes.

**Ex:-**To chatting with your friends using some social media sites, reading some articles from internet, making some projects through Microsoft power point or any other, designing your website, programming something, watching some videos and movies, listening to some songs and many more.

### Examples of Operating Systems:-

**Microsoft windows:-** It is a most popular of the OS, it really is a set of distributions built to provide older Operating Systems (MS-DOS) with a supporting graphical interface and a set of software tools. It’s first version appeared in 1985 and since

then it has not stopped updating in more powerful and diverse versions, as Microsoft, its mother company , prevails in the market of digital technologies.

**GNU / Linux :-** This term refers to the combined use of the kernel free from the Unix family called “Linux”, along with the GNU distribution. The result is one of the main is protagonists in the development of free software, whose source code can be freely used, modified and redistributed.

**UNIX:-** This portable, multi- tasking, multi – user operating system was developed early in 1969, and over the years its rights to copyright they have passed from one company to another. In reality they have passed from one company to another. In reality it is a family of similar OS, many of which have become commercial and others are free format, all from the Linux kernel.

**Ubuntu :-** Based on GNU / Linux, this free and open source Operating System takes its name from the South African philosophy focused on the loyality of man to the rest of the species. The British company owns its rights, subsists on the basis of technical services linked to the program.

**Mac OS:-**The Machintosh operating system, also known as **OSX** or **Mac OS X,** whose environment is based on Unix and has been developed and sold as part of Apple – brand computers since 2002. Part of this family of software was released by Apple as an open and free source operating system called Darwin.

To which they later added components such as Aqua and the Finder, to obtain the interface on which Mac OS X, its most recent versions , is based.

**Solaris :-** Another Unix – like operating System, created I 1992 by Sun Microsystems and used today for SPARC system architectures and x86, common on servers and workstation. It is an officially certified versions of Unix whose released versions is called Open Solaris.

**Android**:- Based on the Linux kernel, this OS for touch screen mobile devices was developed by Android Inc. and later purchased by Google . It is so popular today that sales of Android systems exceed IOS and windows Phone together.

**Chrome OS:-** Currently in the project stage, the Operating system of the Google company is assumed , based on the web and on an open source Linux kernel, initially oriented to mini laptops with ARM or x86 technology processors. This project was announced in 2009, after the explorer Google chrome and your open source project Chromium OS they will show very positive market results.

### What is Work station? Explain Operating Systems for Workstations ?

**Ans:- Workstation**, a high-performance [computer](https://www.britannica.com/topic/computer-graphics) system that is basically designed for a single user and has advanced graphics [capabilities](https://www.britannica.com/dictionary/capabilities), large storage capacity, and a powerful [central processing unit](https://www.britannica.com/technology/central-processing-unit). A workstation is more capable than a [personal](https://www.britannica.com/technology/personal-computer) [computer](https://www.britannica.com/technology/personal-computer) (PC) but is less advanced than a [server](https://www.britannica.com/technology/server) .

Workstation operating system are Windows XP , Windows Vista, Windows 7

, Windows 8 and similar Workstations operating system is primarily. Designed to run applications. Those applications can be text processor, a spread sheet application, presentation software , video or audio editors, games. Workstations operating systems can run services, but are not really designed for it. By services we mean on services that other users can use on the network.

**Ex:--** Services like DHCP , DNS, FTP, Mail, Web servers. Some of that services actually are available on workstation operating systems, but they are not optimized for them. Almost all workstation operating systems support multiple user accounts on the same workstation, but the thing is they are not designed to concurrent multi- user. Workstation OS are not designed to support multiple users at the same time, meaning they don’t do it very well. Most Windows operating systems have a limit of 10 concurrent users at the time. This limit is applied when we share something on our workstation computer.

**Ex:-** Printer or some folder Only 10 users maximum will be able to utilize our shared resources on the workstation OS. Also, workstation operating systems are designed to run on lower end hardware. Suppose workstation operating systems include Windows 95, Windows 98, Windows ME , windows 2000 , Windows XP , Windows Vista, Windows 7, Windows 8 and various Macintosh operating systems as well.

### Discuss about Hand-Held devices Operating Systems ?

**Ans:-** A handheld is any portable device that can be carried and can be any computing or electronic that is compact and portable enough to be held and used in one or both hands.

A mobile operating system is an operating system that helps to run other application software on mobile devices. It is the same kind of software as the famous computer operating systems like Linux and window, but now they aare light and simple to some extent.

The operating systems found on smart phones include Symbian Os, iphone Os, Rim’s BlackBerry, windows Mobile, Palm Web OS, android and Maemo. Android, Web OS, and Maemo are all derived from Linux. The iphone OS originated from BSD and NeXT STEP which are related to Unix.

It combines the beauty of computer and hand use devices. It typically contains a cellular built – in modem and SIM tray for telephony and internet connections. If you buy a mobile, the manufacturer company chooses the OS for that specific device.

### Popular platforms of the Mobile OS :-

1. **Android OS :-** It is a very popular operating system today. It is a mobile OS based on the Linux Kernel and open – source software. The android operating system was developed by Google. The first Android device was launched in 2008.
2. **Bada ( Samsung Electronics ):**- Bada is a Samsung operating system that was launched in 2010. The Samsung wave was the first mobile to use the bada operating system. The Bada operating system offers many mobile features, such as 3-d graphics, application installation and multipoint touch
3. **Blackberry OS:-** It is a mobile operating system developed by research in Motion (RIM) . This operating system was designed specifically for BlackBerry handheld devices. The operating system is beneficial for the corporate users because it provides synchronization with Microsoft Exchange , Novell Group wise email, Lotus Domino, and other business software when used with the Blackberry Enterprise Server.
4. **iPhone Os / iOS:-** iOS was developed by the “Apple inc” for the use on its device. The iOS operating system is the most popular operating system today. It is

a very secure operating system. The iOS operating system is not available for any other mobiles.

1. **Symbian OS:-** it is a mobile operating system that provides a high levl of integration with communication. The Symbian operating system is based on the java language. It combines middleware of wireless communications and personal information management (pim)functionality. The Symbian operating system was developed by Symbian Ltd in 1998 for the use of mobile phones. Nokia was the first company to release Symbian OS on its mobile phone at that time.
2. **Windows Mobile OS :-** This Operating system was developed by Microsoft . It was designed for the pocket PPCs and smart mobiles.
3. **Harmony OS :-** It is the latest mobile operating system that was developed by Huawei for the use of its devices. It is designed primarily for IOT devices.
4. **Palm OS :-** It is a mobile operating system that was developed by Palm ltd for use on personal digital assistants 9PPADs0. It was introduced in 1996. Palm OS is also known as thee Garnet OOs.
5. **Web OOS ( Palm / HP):-** The Web OS is a mobile operating system that was developed by palm. It based on the Linux Kernel. The HP uses this operating system in its mobile and touchpads.

### Explain about Process Control ?

**Ans :-** In an Operating System , Each process has some information that is needed by CPU for the execution of the process . So, we need some kind of data structure to store information about a particular process.

**process control block** (**PCB**) is a data structure used by computer operating systems to store all the information about a process. It is also known as a process descriptor. When a process is created (initialized or installed), the operating system creates a corresponding process control block.

### Attributes of a Process Control Block :-

In [multitasking](https://en.wikipedia.org/wiki/Computer_multitasking) operating systems, the PCB stores data needed for correct and efficient process management.[[1]](https://en.wikipedia.org/wiki/Process_control_block#cite_note-OSConcepts-1) Though the details of these structures are system- dependent, common elements fall in three main categories:

* Process id
* Process state
* Process control

**Process id:-** A Process id is a unique identity of process. Each process is identified with the help of the process id. In a multiuser-multitasking system, data such as the identifier of the parent process, user identifier, user group identifier, etc.

**Process State :-** Process state can be define the status of a process when it is suspended, allowing the OS to restart it later. This always includes the content of general-purpose CPU registers, the CPU process status word, stack and frame pointers, etc.

**Process Control :-**Process control information is used by the OS to manage the process itself. This includes:

* **Process scheduling state** :– The state of the process in terms of "ready", "suspended", etc., and other scheduling information as well, such as priority value, the amount of time elapsed since the process gained control of the CPU or since it was suspended. Also, in case of a suspended process, event identification data must be recorded for the event the process is waiting for;
* **Process structuring information:**– the process's children id's, or the id's of other processes related to the current one in some functional way, which may be represented as a queue, a ring or other data structures;
* **Inter process communication information:**– flags, signals and messages associated with the communication among independent processes;
* **Process Privileges:** – allowed/disallowed access to system resources;
* **Process State:** – new, ready, running, waiting, dead;
* **Process Number (PID):** – unique identification number for each process (also known as [Process ID](https://en.wikipedia.org/wiki/Process_identifier));
* **Program Counter (PC):** – a pointer to the address of the next instruction to be executed for this process;
* **CPU Registers:** – register set where process needs to be stored for execution for running state;
* **CPU Scheduling Information:** – information scheduling CPU time;
* **Memory Management Information:** – page table, memory limits, segment table;
* **Accounting Information:** – amount of [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) used for process execution, time limits, execution ID etc.;
* **I/O Status Information** :– list of I/O devices allocated to the process.

### 13). Explain about Real time Systems ?

Ans :- A **real-time operating system** (**RTOS**) is an [operating system](https://en.wikipedia.org/wiki/Operating_system) (OS) for [real-](https://en.wikipedia.org/wiki/Real-time_computing) [time](https://en.wikipedia.org/wiki/Real-time_computing) applications that processes data and events that have critically defined time constraints.

In a Real time Operating Systems , processing time requirement are calculated in tenths of seconds increments of time. It is time bound system that can be defined as fixed time constraints. In this type of system, processing must be done inside the specified constraints. Otherwise, the system will fail.

### Important reasons for using real time systems :-

1. It offers priority based scheduling, which allows you to separate analytical processing from non – critical processing .
2. The Real time Os provides APII functions that allow cleaner and smaller application code.
3. RTOS offers modular task based development, which allows modular task based testing.
4. The task based API encourages modular development as a task, will typically have a clearly defined role. IT allows designers / teams to work independently on their parts of the project.
5. An RTOS is event drivn with no time wastage on processing time for the event which is not occur.

Ex:-

1. Airlines reservation system
2. air traffic control system
3. Systems that provide immediate updating

### Advantages: -

* 1. Easy to layout , develop and execute real time applications under the real time operating systems.
  2. In a Real time operating systems , the maximum utilization of devices and systems.
  3. Focus on running applications and less importance to applications that are in the queue.
  4. These types of systems are error free
  5. Memory allocation is best managed in these types of systems.

### Dis – Advantages :-

1. Real time operating systems have complicated layout principles and are very costly to develop.
2. Real time operating systems are very complex and can consume critical CPU cycles.

END

UNIT -2

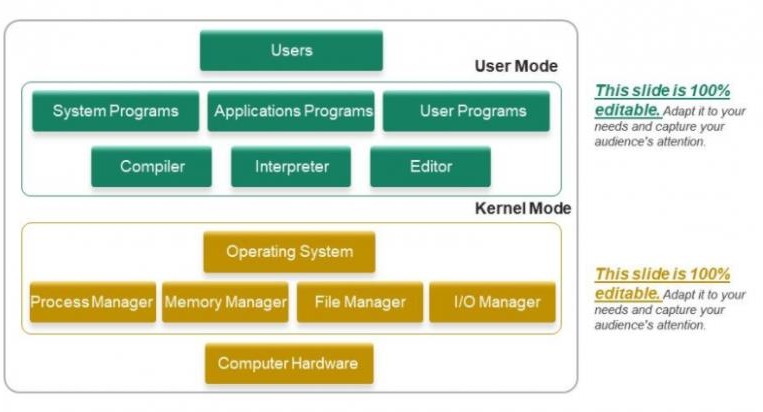
* 1. **Discuss about Processor and user modes ?**

**Ans:-** A processor in a computer running Windows has two different modes: user mode and kernel mode. The processor switches between the two modes depending on what type of code is running on the processor. Applications run in user mode, and core operating system components run in kernel mode.

**User Mode :-** When you start a user-mode application, Windows creates a *process* for the application. The process provides the application with a **private** [***virtual address space***](https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/virtual-address-spaces) and a **private *handle table*.** Because an application's virtual address space is private, one application cannot alter data that belongs to another application. Each application runs in isolation, and if an application crashes, the crash is limited to that one application. Other applications and the operating system are not affected by the crash.

The virtual address space of a user-mode application is limited. A processor running in user mode cannot access virtual addresses that are reserved for the operating system. Limiting the virtual address space of a user-mode application prevents the application from altering, and possibly damaging, critical operating system data.

This diagram illustrates communication between user-mode and kernel-mode components.



**Kernel Mode:-** All code that runs in kernel mode shares a single [virtual address space](https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/virtual-address-spaces). This means that a kernel-mode driver is not isolated from other drivers and the operating system

itself. If a kernel-mode driver accidentally writes to the wrong virtual address, data that belongs to the operating system or another driver could be compromised. If a kernel-mode driver crashes, the entire operating system crashes.

1. **What is Kernels ? Explain it’s Functions ?**

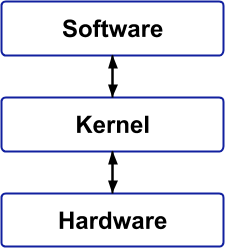
Ans:- A Kernel is a computer program that is the heart and core of an Operating System . It is the primary interface between the hardware and the processes of a computer. It is the most important part of an Operating System.

When ever a system starts, the Kernel is the first program that is loaded after the bootloader because the Kernel has to handle the rest of the thing of the system for the Operating System. The Kernel remains in the memory until the Operating System is shut- down.

The Kernel is responsible for low-level tasks such as disk management, memory management, task management, etc. It provides an interface between the user and the hardware components of the system. When a process makes a request to the Kernel, then it is called “System Call”.

**Functions of a Kernel:-**

* + **Access Computer resource :-** A Kernel can access various computer resources like the CPU, I/O devices and other resources. It acts as a bridge between the user and the resources of the system.
  + **Resource Management :-** It is the duty of a Kernel to share the resources between various process in such a way that there is uniform access to the resources by every process.
  + **Memory Management :-** Every process needs some memory space. So, memory must be allocated and de-allocated for its execution. All these memory management is done by a Kernel.
  + **Device Management :-** The peripheral devices connected in the system are used by the processes. So, the allocation of these devices is managed by the Kernel.

1. **What is Kernels ? What are different types of Kernels ?**

Ans:- A Kernel is a computer program that is the heart and core of an Operating System . It is the primary interface between the hardware and the processes of a computer. It is the most important part of an Operating System.

Whenever a system starts, the Kernel is the first program that is loaded after the bootloader because the Kernel has to handle the rest of the thing of the system for the Operating System. The Kernel remains in the memory until the Operating System is shut- down.

**Types of Kernel:-**

In general, there are five types of Kernel. They are:

1. **Monolithic Kernels :-** Monolithic Kernels are those Kernels where the user services and the kernel services are implemented in the same memory space i.e. different memory for user services and kernel services are not used in this case. The size of the Kernel is increased and this , increases the size of the Operating System. As there is no separate User Space and Kernel Space, so the execution of the process will be faster in Monolithic Kernels.

**Advantages:-**

* + It provides CPU scheduling, memory scheduling, file management through System calls only.
  + Execution of the process is fast because there is no separate memory space for user and kernel.

**Disadvantages:-**

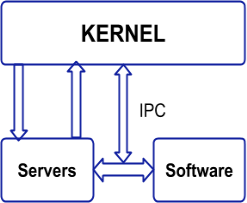
* + If any service fails, then it leads to system failure.
  + If new services are to be added then the entire Operating System needs to be modified.

1. **Microkernel :-** A Microkernel is different from Monolithic kernel because in a Microkernel, the user services and kernel services are implemented into different spaces, we use User Space and Kernel Space in case of Microkernels. As we are using User Space and Kernel Space separately, so it reduces the size of the Kernel and this reduces the size of Operating System.

we are using different spaces for user services and kernel service, so the communication between application and services is done with the help of message parsing and this, in turn, reduces the speed of execution.

**Advantages :-** If new services are to be added then it can be easily added.

**Disadvantages :-** Since we are using User Space and Kernel Space separately, so the communication between these can reduce the overall execution time.

1. **Hybrid Kernel :-** A Hybrid Kernel is a combination of both Monolithic Kernel and Microkernel. It makes the use of the speed of Monolithic Kernel and the modularity of Microkernel.

Hybrid kernels are micro kernels that have some "non-essential" code in kernel-space in order for the code to run more quickly than it would be in user-space

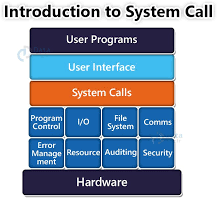
1. **Nanokernel :-** In a Nanokrnel , as the name suggests, the whole code of the kernel is very small, the code executing in the privileged mode of the hardware is very small. The term nanokernel is used to describe a kernel that supports a nanosecond clock resolution.
2. **Exokernel :-** Exokernel is an Operating System kernel that is developed by the MIT parallel and the Distributed Operating Systems group. Here in this type of kernel, the resource protection is separated from the management and this, in turn, results in allowing us to perform application-specific customization.

In the Exokernel, the idea is not to implement all the abstractions. But the idea is to impose as few abstractions as possible and by doing so the abstraction should be used only when needed. So, no force abstraction will be there in Exokernel and this is the feature that makes it different from a Monolithic Kernel and Microkernel. But the drawback of this is the complex design. The design of the Exokernel is very complex.

1. **Discuss about Systemcalls ?**

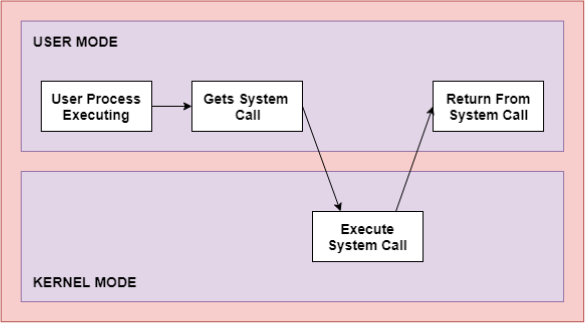
Ans:- A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System call provides the services of the operating system to the user programs via Application Program Interface(API).

It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.



System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.

A figure representing the execution of the system call is given as follows −



1.The processes execute normally in the user mode until a system call interrupts it. 2.Then the system call is executed on a priority basis in the kernel mode.

1. After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed.

**In general, system calls are required in the following situations −**

* + If a file system requires the creation or deletion of files. Reading and writing from files also require a system call.
  + Creation and management of new processes.
  + Network connections also require system calls. This includes sending and receiving packets.
  + Access to a hardware devices such as a printer, scanner etc. requires a system call.

**Types of System Calls :-**

There are mainly five types of system calls.

1. **Process Control:-**

These system calls deal with processes such as process creation, process termination

etc.

1. **File Management:-**

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

1. **Device Management :-**

These system calls are responsible for device manipulation such as reading from device ` buffers, writing into device buffers etc.

1. **Information Maintenance : -**

These system calls handle information and its transfer between the operating system and the user program.

1. **Communication :-**

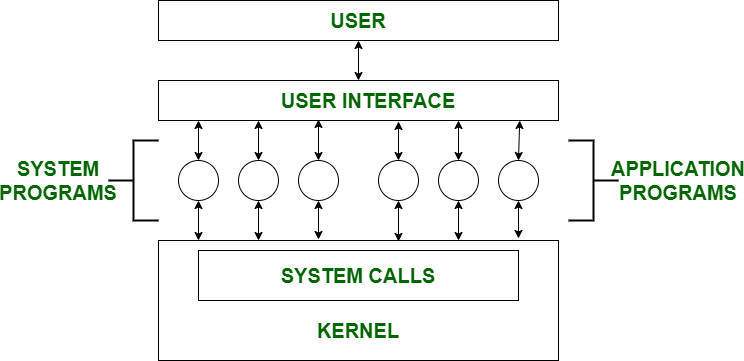
These system calls are useful for inter process communication. They also deal with creating and deleting a communication connection.

Some of the examples of all the above types of system calls in Windows and Unix are given as follows –

|  |  |  |
| --- | --- | --- |
| **Types of System Calls** | **Windows** | **Linux** |
| Process Control | CreateProcess()ExitProcess()WaitForSingleObject() | fork()exit()wait() |
| File Management | CreateFile()ReadFile()WriteFile()CloseHandle() | open()read()  write()close() |
| Device Management | SetConsoleMode()ReadConsole()WriteConsole() | ioctl()read()  write() |
| Information Maintenance | GetCurrentProcessID()SetTimer()Sleep() | getpid()alarm()  sleep() |
| Communication | CreatePipe()CreateFileMapping()MapViewOfFile() | pipe()shmget()  mmap() |

1. **Discuss about System Programs ?**

Ans:- System Programming can be defined as the act of building Systems Software using System Programming Languages. According to Computer Hierarchy, one which comes at last is Hardware. Then it is Operating System, System Programs, and finally Application Programs. Program Development and Execution can be done conveniently in System Programs.



So here, the user can only view up-to-the System Programs he can’t see System Calls. System Programs can be divided into these categories :

1. **File Management :–** A file is a collection of specific information stored in the memory of a computer system. File management is defined as the process of manipulating files in the computer system, its management includes the process of creating, modifying and deleting files.
   * It helps to create new files in the computer system and placing them at specific locations.
   * It helps in easily and quickly locating these files in the computer system.
   * It makes the process of sharing files among different users very easy and user- friendly.
   * It helps to store files in separate folders known as directories.
   * It helps users to modify the data of files or to modify the name of files in directories.
2. **Status Information:–** Information like date, time amount of available memory, or disk space is asked by some users. Others providing detailed performance, logging, and debugging information which is more complex. All this information is formatted and displayed on output devices or printed. Terminal or other output devices or files or a window of GUI is used for showing the output of programs.
3. **File Modification :–** For modifying the contents of files we use this. For Files stored on disks or other storage devices, we used different types of editors. For searching contents of files or perform transformations of files we use special commands.
4. **Programming-Language support:–** For common programming languages, we use Compilers, Assemblers, Debuggers, and interpreters which are already provided to users. It provides all support to users. We can run any programming language. All languages of importance are already provided.
5. **Program Loading and Execution:–** When the program is ready after Assembling and compilation, it must be loaded into memory for execution. A loader is part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages for starting a program
6. **Communications:–** Virtual connections among processes, users, and computer systems are provided by programs. Users can send messages to another user on their screen, User can send e-mail, browsing on web pages, remote login, the transformation of files from one user to another. Some examples of system program in O.S. are –

* Windows 10
* Mac OS X
* Ubuntu
* Linux
* Unix
* Android
* Anti-virus
* Disk formatting
* Computer language translators

1. **Explain in detail System View of the Process and Resources?**

Ans :- The system views the operating system as a resource allocator. There are many resources such as CPU time, memory space, file storage space, I/O devices etc. that are required by processes for execution.

The operating system manages the resources, decides between competing demands, controls the program execution, etc. According to this point of view, the operating system's purpose is to maximize performance.

Operating system (OS), program that manages a computer's resources, especially the allocation of those resources among other programs. Typical resources include the central processing unit (CPU), computer memory, file storage, input/output (I/O) devices, and network connections.

1. **Explain about Process Abstraction and Process Hierarchy ?**

Ans:- **Abstraction :-** The process of establishing the decomposition of a problem into simpler and more understood primitives is basic to science and software engineering. This process has many underlying techniques of abstraction.

An abstraction is a model. The process of transforming one abstraction into a more detailed abstraction is called refinement. The new abstraction can be referred to as a refinement of the original one. Abstractions and their refinements typically do not coexist in the same system description. Composition occurs when two abstractions are used to define another higher abstraction. Decomposition occurs when an abstraction is split into smaller abstractions.

Information management is one of the goals of abstraction. Complex features of one abstraction are simplified into another abstraction. Good abstractions can be very useful while bad abstractions can be very harmful. A good abstraction leads to reusable components.

Information hiding distinguishes between public and private information. Only the essential information is made public while internal details are kept private. This simplifies interactions and localizes details and their operations into well defined units.

Abstraction, in traditional systems, naturally forms layers representing different levels of complexity. Each layer describes a solution. These layers are then mapped onto each other. In this way, high level abstractions are materialized by lower level abstractions until a simple realization can take place.

**Process hierarchy:-** Now-a-days all general purpose operating systems permit a user to create and destroy processes. A process can create several new processes during its time of execution.

The creating process is called the Parent Process and the new process is called Child Process. There are different ways for creating a new process. These are as follows −

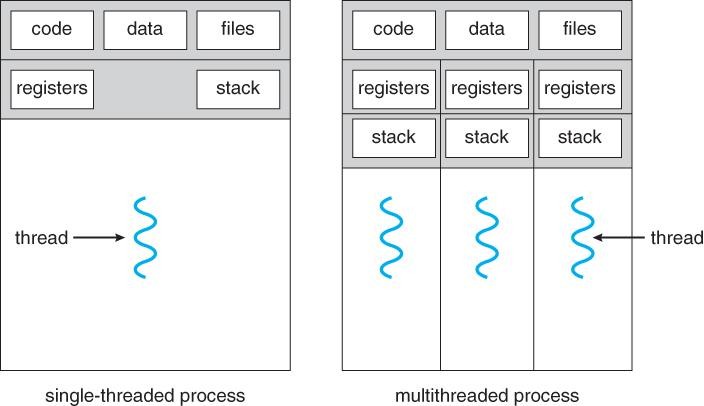
* **Execution** − The child process is executed by the parent process concurrently or it waits till all children get terminated.
* **Sharing** − The parent or child process shares all resources like memory or files or children process shares a subset of parent’s resources or parent and children process share no resource in common.

The reasons that parent process terminates the execution of one of its children are as follows

−

* The child process has exceeded its usage of resources that have been allocated. Because of this there should be some mechanism which allows the parent process to inspect the state of its children process.
* The task that is assigned to the child process is no longer required.

1. **Discuss about Threads ?**



**Ans:-** Thread is a sequential flow of tasks within a process. Threads in OS can be of the same or different types. Threads are used to increase the performance of the applications.

Each thread has its own program counter, stack, and set of registers. But the threads of a single process might share the same code and data/file. Threads are also termed as lightweight processes as they share common resources.

**Eg:-** While playing a movie on a device the audio and video are controlled by different threads in the background.

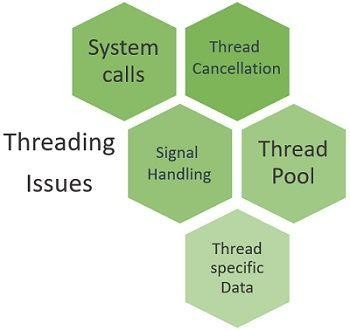
**Advantages of Thread:-**

1. Responsiveness
2. Resource sharing, hence allowing better utilization of resources 3.**Economy :-** Creating and managing threads becomes easier
3. **Scalability :-** One thread runs on one CPU.In Multithreaded processes, threads can be distributed over a series of processsors to scale
4. **Context Switching is smooth :-** Context switching refers to the procedure followed by the CPU to change from one task to another
5. **Enhanced Throughput of the system :-** Suppose a process is divided into multiple threads, and the function of each thread is considered as one job, then the number of jobs completed per unit of time increases which then leads to an increase in the throughput of the system.

**9) Discuss about Threading Issues ?**

Ans:- There are several threading issues when we are in a multithreading environment. In this

section, we will discuss the threading issues with system calls, cancellation of thread, signal handling, thread pool and thread-specific data.

Threading Issues in OS :-

1. [System Calls](https://binaryterms.com/threading-issues-in-os.html#SystemCalls)
2. [Thread Cancellation](https://binaryterms.com/threading-issues-in-os.html#ThreadCancellation)
3. [Signal Handling](https://binaryterms.com/threading-issues-in-os.html#SignalHandling)
4. [Thread Pool](https://binaryterms.com/threading-issues-in-os.html#ThreadPool)
5. [Thread Specific Data](https://binaryterms.com/threading-issues-in-os.html#ThreadSpecificData)
6. **fork() and exec() System Calls :-**The fork() and exec() are the system calls. The fork() call creates a duplicate process of the process that invokes fork(). The new duplicate process is called child process and process invoking the fork() is called the parent process. Both the parent process and the child process continue their execution from the instruction that is just after the fork().

Next system call i.e. exec() system call when invoked replaces the program along with all its threads with the program that is specified in the parameter to exec(). Typically the exec() system call is lined up after the fork() system call.

1. **Thread cancellation:-** Termination of the thread in the middle of its execution it is termed as ‘thread cancellation’. Let us understand this with the help of an example. Consider that there is a multithreaded program which has let its multiple threads to search through a database for some information. However, if one of the thread returns with the desired result the remaining threads will be cancelled**.**

Now a thread which we want to cancel is termed as target thread. Thread cancellation can be performed in two ways:

**Asynchronous Cancellation:-** In asynchronous cancellation, a thread is employed to terminate the target thread instantly.

**Deferred Cancellation:-** In deferred cancellation, the target thread is scheduled to check itself at regular interval whether it can terminate itself or not.

1. **Signal Handling:-** Signal handling is more convenient in the single-threaded program as the signal would be directly forwarded to the process. But when it comes to multithreaded program, the issue arrives to which thread of the program the signal should be delivered.
2. **Thread Pool:-** When a user requests for a webpage to the server, the server creates a separate thread to service the request. Although the server also has some potential issues. Consider if we do not have a bound on the number of actives thread in a system and would create a new thread for every new request then it would finally result in exhaustion of system resources.

We are also concerned about the time it will take to create a new thread. It must not be that case that the time require to create a new thread is more than the time required by the thread to service the request and then getting discarded as it would result in wastage of CPU time.

The solution to this issue is the thread pool. The idea is to create a finite amount of threads when the process starts. This collection of threads is referred to as the thread pool. The threads stay in the thread pool and wait till they are assigned any request to be serviced.

Whenever the request arrives at the server, it invokes a thread from the pool and assigns it the request to be serviced. The thread completes its service and return back to the pool and wait for the next request.

1. **Thread Specific data:-** We all are aware of the fact that the threads belonging to the same process share the data of that process. Here the issue is what if each particular thread of the process needs its own copy of data. So the specific data associated with the specific thread is referred to as **thread-specific data.**

Consider a transaction processing system, here we can process each transaction in a different thread. To determine each transaction uniquely we will associate a unique identifier with it. Which will help the system to identify each transaction uniquely.

**10) Discuss about Thread Libraries ?**

Ans:- Thread Libraries has a collection of functions that useful in creating and controlling threads. Programmers can access these thread libraries using an application programming interface (API). Thread libraries can be the *user level library* or *kernel level library*.

If the thread library is implemented at the userspace then code and data of the thread library would reside in user space. In this case, invoking any function from thread library would be a simple function call and it won’t be a system call.

If the thread library is implemented at the kernel space then code and data of the library would reside in the kernel space and would be supported by the operating system. In this case, invoking a function from thread library would be a system call to the kernel. In the section further, we would be discussing three kinds of thread libraries.

Thread Libraries in Operating System

1. [Pthreads Library](https://binaryterms.com/thread-libraries-in-os.html#PthreadsLibrary)
2. [Win32 Library](https://binaryterms.com/thread-libraries-in-os.html#Win32Library)
3. [Java Library](https://binaryterms.com/thread-libraries-in-os.html#JavaLibrary)

**Pthreads Library:-** Pthreads are also termed as **POSIX** thread library. This can be implemented either at the *userspace* or at the *kernel space*. Pthreads library is often implemented at LINUX, UNIX, Solaris, Mac OSX. The Pthread program must always have a **pthread.h** header file.

**Win32 Library :-** Creation of thread in Win2 library is similar to pthread library. To create a thread using the Win32 library always include **windows.h** header file in the program. The Win32 thread library is a *kernel-level* library which means invoking the Win32 library function results in a system call.

**Java Thread Library :-** You must have seen that mostly the java virtual machine JVM runs on the top of the host operating system. That’s why java threads are created and controlled by using the available library at the host operating system.

In the Windows operating system, the java threads are implemented using Win32 API and in operating systems such as Linux and UNIX, the java thread is implemented using Pthread library. In a Java program, there is at least one thread of control. Well, there are twomethods to creating a thread in Java program first you can derive a new class from a Thread class and override it’s run() method. The second method is to define a class and implement a Runnable interface to it, the second method is the most common method to be used.

1. **Write Short notes on Process Scheduling ?**

Ans:- Process Scheduling is an OS task that schedules processes of different states like ready, waiting, and running.

Process scheduling allows OS to allocate a time interval of CPU execution for each process. Another important reason for using a process scheduling system is that it keeps the CPU busy all the time. This allows you to get the minimum response time for programs.

**Scheduling Objectives:-**

Here, are important objectives of Process scheduling

* + Maximize the number of interactive users within acceptable response times.
  + Achieve a balance between response and utilization.
  + It also should give reference to the processes holding the key resources.

**Two State Process Model :-** 1) **Running State 2) Not Running State**

**Running State:-**

In the [Operating system,](https://www.guru99.com/operating-system-tutorial.html) whenever a new process is built, it is entered into the system, which should be running.

**Not Running:-**

The process that are not running are kept in a queue, which is waiting for their turn to execute. Each entry in the queue is a point to a specific process.

**Type of Process Schedulers:-**

A scheduler is a type of system software that allows you to handle process scheduling. There are mainly three types of Process Schedulers:

1) Long Term Scheduler 2) Short Term Scheduler 3) Medium Term Scheduler

**Long Term Scheduler :-** Long term scheduler is also known as a job scheduler. This scheduler regulates the program and select process from the queue and loads them into memory for execution. It also regulates the degree of multi-programing.

The main goal of this type of scheduler is to offer a balanced mix of jobs, like Processor, I/O jobs., that allows managing multiprogramming.

**Medium Term Scheduler:-** Medium-term scheduling is an important part of swapping. It enables you to handle the swapped out-processes. In this scheduler, a running process can become suspended, which makes an I/O request.

A running process can become suspended if it makes an I/O request. A suspended processes can’t make any progress towards completion. In order to remove the process from memory and make space for other processes, the suspended process should be moved to secondary storage.

**Short Term Scheduler:-** Short term scheduling is also known as CPU scheduler. The main goal of this scheduler is to boost the system performance according to set criteria. This helps you to select from a group of processes that are ready to execute and allocates CPU to one of them. The dispatcher gives control of the CPU to the process selected by the short term scheduler.

**12). Discuss about Non- Premptive Scheduling Algorithm ?**

**Ans :- Non Preemptive Priority Scheduling** :- In the Non Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them. Once the process gets scheduled, it will run till the completion. Generally, the lower the priority number, the higher is the priority of the process. The people might get confused with the priority numbers, hence in the GATE, there clearly mention which one is the highest priority and which one is the lowest one.

**Advantages:-** 1.Offers low scheduling overhead

* 1. It is conceptually very simple method

**Dis-Advantages:-** 1. It can lead to starvation especially for those real time tasks

* + 1. It can make real time and priority Scheduling difficult
    2. Poor response time for processes.

**Example:-**

In the Example, there are 7 processes P1, P2, P3, P4, P5, P6 and P7. Their priorities, Arrival Time and burst time are given in the table.

|  |  |  |  |
| --- | --- | --- | --- |
| Process ID | Priority | Arrival Time | Burst Time |
| 1 | 2 | 0 | 3 |
| 2 | 6 | 2 | 5 |
| 3 | 3 | 1 | 4 |
| 4 | 5 | 4 | 2 |
| 5 | 7 | 6 | 9 |
| 6 | 4 | 5 | 4 |
| 7 | 10 | 7 | 10 |

We can prepare the Gantt chart according to the Non Preemptive priority scheduling.

The Process P1 arrives at time 0 with the burst time of 3 units and the priority number

2. Since No other process has arrived till now hence the OS will schedule it immediately.

Meanwhile the execution of P1, two more Processes P2 and P3 are arrived. Since the priority of P3 is 3 hence the CPU will execute P3 over P2.

Meanwhile the execution of P3, All the processes get available in the ready queue. The Process with the lowest priority number will be given the priority. Since P6 has priority number assigned as 4 hence it will be executed just after P3.

After P6, P4 has the least priority number among the available processes; it will get executed for the whole burst time.

Since all the jobs are available in the ready queue hence All the Jobs will get executed according to their priorities. If two jobs have similar priority number assigned to them, the one with the least arrival time will be executed.

os Non Preemptive Priority SchedulingFrom the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, waiting time and response time will be determined.

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Process Id | Priori ty | Arriv al Time | Burst Time | Completi on Time | Turnaround Time | Waitin g Time | Respons e Time |
| 1 | 2 | 0 | 3 | 3 | 3 | 0 | 0 |
| 2 | 6 | 2 | 5 | 18 | 16 | 11 | 13 |
| 3 | 3 | 1 | 4 | 7 | 6 | 2 | 3 |
| 4 | 5 | 4 | 2 | 13 | 9 | 7 | 11 |
| 5 | 7 | 6 | 9 | 27 | 21 | 12 | 18 |
| 6 | 4 | 5 | 4 | 11 | 6 | 2 | 7 |
| 7 | 10 | 7 | 10 | 37 | 30 | 18 | 27 |

**Avg Waiting Time = (0+11+2+7+12+2+18)/7 = 52/7 units**

**13) Discuss about Preemptive Scheduling algorithm ?**

Ans:- **Preemptive Priority Scheduling:-** In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the preemptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.

Once all the jobs get available in the ready queue, the algorithm will behave as non- preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

**Example:-**

There are 7 processes P1, P2, P3, P4, P5, P6 and P7 given. Their respective priorities, Arrival Times and Burst times are given in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Process Id | Priority | Arrival Time | Burst Time |
| 1 | 2(L) | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 6 | 1 | 7 |
| 3 | 3 | 2 | 3 |
| 4 | 5 | 3 | 6 |
| 5 | 4 | 4 | 5 |
| 6 | 10(H) | 5 | 15 |
| 7 | 9 | 15 | 8 |

GANTT chart Preparation

At time 0, P1 arrives with the burst time of 1 units and priority 2. Since no other process is available hence this will be scheduled till next job arrives or its completion (whichever is lesser).

os Preemptive Priority Scheduling GANTT chart PreparationAt time 1, P2 arrives. P1 has completed its execution and no other process is available at this time hence the Operating system has to schedule it regardless of the priority assigned to it.

os Preemptive Priority Scheduling GANTT chart Preparation 1

The Next process P3 arrives at time unit 2, the priority of P3 is higher to P2. Hence the execution of P2 will be stopped and P3 will be scheduled on the CPU.

os Preemptive Priority Scheduling GANTT chart Preparation 2

During the execution of P3, three more processes P4, P5 and P6 becomes available. Since, all these three have the priority lower to the process in execution so PS can't preempt the process. P3 will complete its execution and then P5 will be scheduled with the priority highest among the available processes.

os Preemptive Priority Scheduling GANTT chart Preparation 3

Meanwhile the execution of P5, all the processes got available in the ready queue. At this point, the algorithm will start behaving as Non Preemptive Priority Scheduling. Hence now, once all the processes get available in the ready queue, the OS just took the process with the highest priority and execute that process till completion. In this case, P4 will be scheduled and will be executed till the completion.

os Preemptive Priority Scheduling GANTT chart Preparation 4

Since P4 is completed, the other process with the highest priority available in the ready queue is P2. Hence P2 will be scheduled next.

os Preemptive Priority Scheduling GANTT chart Preparation 5

P2 is given the CPU till the completion. Since its remaining burst time is 6 units hence P7 will be scheduled after this.

os Preemptive Priority Scheduling GANTT chart Preparation 6

The only remaining process is P6 with the least priority, the Operating System has no choice unless of executing it. This will be executed at the last.

os Preemptive Priority Scheduling GANTT chart Preparation 7

The Completion Time of each process is determined with the help of GANTT chart. The turnaround time and the waiting time can be calculated by the following formula.

1. Turnaround Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process Id | | Priority | | Arrival Time | | Burst Time | | Completion Time | | Turn around Time | | Waiting Time | |
| 1 | | 2 | | 0 | | 1 | | 1 | | 1 | | 0 | |
| 2 | | 6 | | 1 | | 7 | | 22 | | 21 | | 14 | |
| 3 | | 3 | | 2 | | 3 | | 5 | | 3 | | 0 | |
| 4 | | 5 | | 3 | | 6 | | 16 | | 13 | | 7 | |
| 5 | | 4 | | 4 | | 5 | | 10 | | 6 | | 1 | |
| 6 | | 10 | | 5 | | 15 | | 45 | | 40 | | 25 | |
| 7 | | 9 | | 6 | | 8 | | 30 | | 24 | | 16 | |

**Avg Waiting Time = (0+14+0+7+1+25+16)/7 = 63/7 = 9 units**

END

## UNIT - 3

### What is Process Management of OS?

**Ans:-** Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems. The OS must allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other methods and allows synchronization among processes.

### Process Attributes (or) Process Concepts :-

1. Process State
2. Process Control Block (PCB)
3. Process Operations
4. Process Scheduling
5. Process Synchronization
6. Inter Process Communication (IPC)
7. Deadlock

**Process States :-** process state is a condition of the process at a specific instant of time. It also defines the current position of the process. There are mainly seven stages of a process . They are “ New, Ready, Waiting , Executing , Blocked , suspended , Terminated “. After completing every step, all the resources are used by a process, and memory becomes free.

**Process Control Block (PCB) :-** PCB stands for Process Control Block. It is a data structure that is maintained by the Operating System for every process. The PCB should be identified by an integer Process ID (PID). It helps you to store all the information required to keep track of all the running processes.

It is also accountable for storing the contents of processor registers. These are saved when the process moves from the running state and then returns back to it. The information is quickly updated in the PCB by the OS as soon as the process makes the state transition. Now, important components of PCB

* + **Process state:** A process can be new, ready, running, waiting, etc.
  + **Program counter:** The program counter lets you know the address of the next instruction, which should be executed for that process.
  + **CPU registers:** This component includes accumulators, index and general- purpose registers, and information of condition code.
  + **CPU scheduling information:** This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.
  + **Accounting and business information:** It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.
  + **Memory-management information:** This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.
  + **I/O status information:** This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

### Process operation:-

1. **Creation:-** This is the initial step of process execution activity. Process creation means the construction of a new process for the execution. There are several events that leads to the process creation. Some of the such events are following:

* When we start the computer, system creates several background processes.
* A user may request to create a new process.
* A process can create a new process itself while executing.

1. **Termination:-** Process termination is an operation in which a Process is terminated after the execution of its last instruction.

**Process Scheduling :-** The process scheduling is the activity of the process manager, that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing. There are 2 types of scheduling. They are:

### 1.Non-Preemptive 2.Preemptive

**Process synchronization :-** It is the task of synchronizing the execution of processes in such a manner that no 2 processes have access to the same shared data and resource.

**IPC:- I**PC is the method of Communication which is used to make communication between process through which processes interact with each other for gaining access of shared data and resources.

**Deadlock:-** Deadlock is a condition that occurs when multiple processes wait for each other to free up resources and as a result, all processes remains halted. In other words, deadlocks are the set of processes which are blocked each process is holding a resource and waiting to acquire a resource held by another process.

### What is deadlock?

**Ans :-** Deadlock is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software.

The earliest computer [operating systems](https://www.techtarget.com/whatis/definition/operating-system-OS) ran only one program at a time. All of the resources of the system were available to this one program. Operating systems ran multiple programs at once, interleaving them. Programs were required to specify in advance what resources they needed so that they could avoid conflicts with other programs running at the same time. Eventually some operating systems offered dynamic allocation of resources. Programs could request further allocations of resources after they had begun running. This led to the problem of the deadlock. Here is the simplest example:

Program 1 requests resource A and receives it.

Program 2 requests resource B and receives it.

Program 1 requests resource B and is queued up, pending the release of B. Program 2 requests resource A and is queued up, pending the release of A.

Now neither program can proceed until the other program releases a resource. The operating system cannot know what action to take. At this point the only alternative is to abort (stop) one of the programs.

Learning to deal with deadlocks had a major impact on the development of operating systems and the structure of [databases](https://www.techtarget.com/searchdatamanagement/definition/database). Data was structured and the order of requests was constrained in order to avoid creating deadlocks. The 3 necessary conditions for a deadlock to arise are as follows.

* **Mutual Exclusion**: Only one process can use a resource at any given time

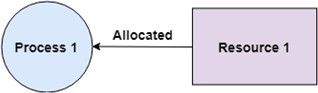
i.e. the resources are non-sharable.

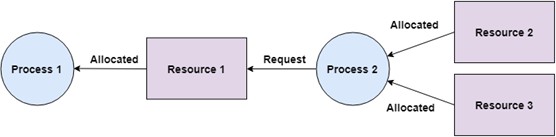
* **Hold and wait**: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.
* **No preemption**: The resource can be released by a process voluntarily i.e. after execution of the process.

### Write a Short notes on Deadlock Characteristics ?

**Ans:-** A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process. A deadlock occurs if the 4 conditions hold true. But these conditions are not mutually exclusive. They are given as follows −

**Mutual Exclusion:** There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.

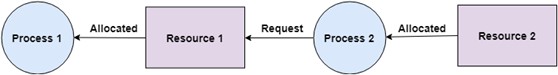
Hold and Wait:-



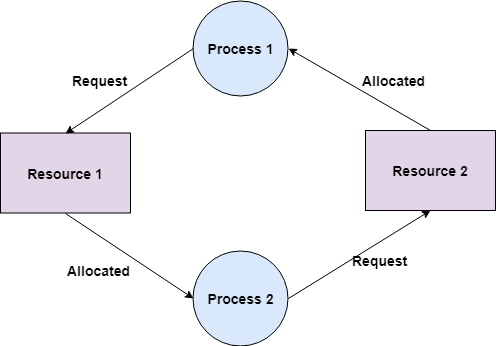
A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds

Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.

**No Preemption:-** A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



**Circular Wait:-** A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



### Explain about Deadlock handling Approaches ?

**Ans:-** The first two methods are used to ensure the system never enters a deadlock.

**Deadlock Ignorance :-** The system assumes that deadlock never occurs. Since the problem of deadlock situation is not frequent, some systems simply ignore it. Operating systems such as UNIX and Windows follow this approach. if a deadlock occurs we can reboot our system and the deadlock is resolved automatically.

**Deadlock Prevention:-** This is done by restraining the ways a request can be made. Since deadlock occurs when all the above four conditions are met, we try to prevent any one of them, thus preventing a deadlock.

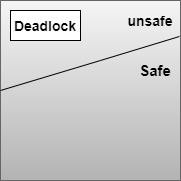
**Deadlock Avoidance :-** When a process requests a resource, the deadlock avoidance algorithm examines the resource-allocation state. If allocating that resource sends the system into an unsafe state, the request is got granted.

It requires additional information such as how many resources of each type is required by a process. If the system enters into an unsafe state, it has to take a step back to avoid deadlock.

**Deadlock Detection and Recovery:-**We let the system fail into a deadlock and if it happens, we detect it using a detection algorithm and try to recover.

* Aborting all the deadlocked processes.
* Abort one process at a time until the system recovers from the deadlock.
* Resource Preemption: Resources are taken one by one from a process and assigned to higher priority processes until the deadlock is resolved.
  1. **Explain about Deadlock Avoidance ?**

**Ans:-** The deadlock Avoidance method is used by the operating system in order to check whether the system is in a safe state or in an unsafe state and in order to avoid the deadlocks, the process must need to tell the operating system about the maximum number of resources a process can request in order to complete its execution.

**Deadlock Avoidance work :-** The request for any resource will be granted only if the resulting state of the system doesn't cause any deadlock in the system. This method checks every step performed by the operating system. Any process continues its execution until the system is in a safe state. Once the system enters into an unsafe state, the operating system has to take a step back.

With the help of a deadlock-avoidance

algorithm, you can dynamically assess the resource-allocation state so that there can never be a circular-wait situation.

The simplest and useful approach is any process should declare the maximum number of resources of each type it will need. The algorithms of deadlock avoidance mainly examine the resource allocations so that there can never be an occurrence of circular wait conditions. Deadlock avoidance can mainly be done with the help of Banker's Algorithm.

Let us first understand the concept of Safe and Unsafe states

**Safe State and Unsafe State :-** A state is safe if the system can allocate resources to each process( up to its maximum requirement) in some order and still avoid a deadlock. Formally, a system is in a safe state only, if there exists a safe sequence.

So a safe state is not a deadlocked state and conversely a deadlocked state is an unsafe state.

In an Unsafe state, the operating system cannot prevent processes from requesting resources in such a way that any deadlock occurs. It is not necessary that all unsafe states are deadlocks; an unsafe state may lead to a deadlock.

The above Figure shows the Safe, unsafe, and deadlocked state spaces

### Deadlock Avoidance Example :-

Let us consider a system having 12 magnetic tapes and three processes P1, P2, P3. Process P1 requires 10 magnetic tapes, process P2 may need as many as 4 tapes, process P3 may need up to 9 tapes. Suppose at a time to, process P1 is holding 5 tapes, process P2 is holding 2 tapes and process P3 is holding 2 tapes. (There are 3 free magnetic tapes)

So at time t0, the system is in a safe state. The sequence is <P2,P1,P3> satisfies the safety condition. Process P2 can immediately be allocated all its tape drives and then return them. After the return the system will have 5 available tapes, then process P1 can get all its tapes and return them ( the system will then have 10 tapes); finally, process P3 can get all its tapes and return them (The system will then have 12 available tapes).

A system can go from a safe state to an unsafe state. Suppose at time t1, process P3 requests and is allocated one

|  |  |  |
| --- | --- | --- |
| **Processes** | **Maximum Needs** | **Current Needs** |
| **P1** | 10 | 5 |
| **P2** | 4 | 2 |
| **P3** | 9 | 2 |

more tape. The system is no longer in a safe state. At this point, only process P2 can be allocated all its tapes. When it returns them the system will then have only 4 available tapes. Since P1 is allocated five tapes but has a maximum of ten so it may request 5 more tapes. If it does so, it will have to wait because they are unavailable. Similarly, process P3 may request its additional 6 tapes and have to wait which then results in a deadlock.

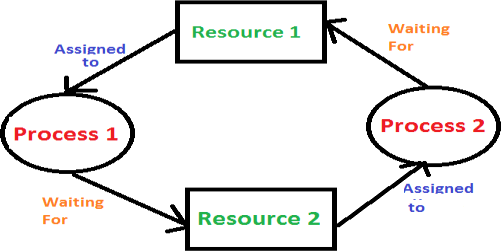
The mistake was granting the request from P3 for one more tape. If we made P3 wait until either of the other processes had finished and released its resources, then we could have avoided the deadlock

The main key of the deadlock avoidance method is whenever the request is made for resources then the request must only be approved only in the case if the **resulting state** is a **safe state**.

### Explain about Deadlock Detection and Recovery ?

**Ans:- Deadlock Detection :-** T**he OS periodically checks the system for any deadlock**. In case, it finds any of the deadlock then the OS will recover the system using some recovery techniques. The main task of the OS is detecting the deadlocks. This Purpose follows 2 methods.

1. **Single instance resources type :–** In this case for Deadlock detection, we can run an algorithm to check for the cycle in the Resource Allocation Graph. The presence of a cycle in the graph is a sufficient condition for deadlock.



1. In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.
2. **Multiple instance of resources type :–** Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

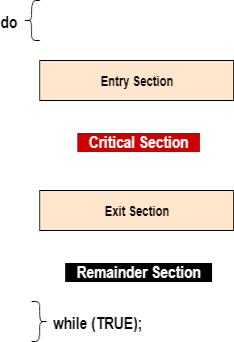
**Deadlock Recovery :-** A traditional operating system such as Windows doesn’t deal with deadlock recovery as it is a time and space-consuming process. Real-time operating systems use Deadlock recovery.

1. **Killing the process :–** Killing all the processes involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till the system recovers from deadlock. Killing all the processes one by one helps a system to break circular wait condition.
2. **Resource Preemption :–** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

### Explain Concurrent and Dependent Processes ?

**Ans:-** Concurrency is the execution of the multiple instruction sequences at the same time. It happens in the operating system when there are several process threads running in parallel. The running process threads always communicate with each other through shared memory or message passing. Concurrency results in sharing of resources result in problems like deadlocks and resources starvation.

It helps in techniques like coordinating execution of processes, memory allocation and execution scheduling for maximizing throughput.

**Principles of Concurrency :-** Both interleaved and overlapped processes can be viewed as examples of concurrent processes, they both present the same problems. The relative speed of execution cannot be predicted. It depends on the following:

* The activities of other processes
* The way operating system handles interrupts
* The scheduling policies of the operating system

### Problems in Concurrency :-

* **Sharing global resources :–** Sharing of global resources safely is difficult. If two processes both make use of a global variable and both perform read and write on that variable, then the order in which various read and write are executed is critical.
* **Optimal allocation of resources :–** It is difficult for the operating system to manage the allocation of resources optimally.
* **Locating programming errors :–** It is very difficult to locate a programming error because reports are usually not reproducible.
* **Locking the channel :–** It may be inefficient for the operating system to simply lock the channel and prevents its use by other processes.

### Advantages of Concurrency :-

* **Running of multiple applications :–** It enable to run multiple applications at the same time.
* **Better resource utilization:–** It enables that the resources that are unused by one application can be used for other applications.
* **Better average response time :–**Without concurrency, each application has to be run to completion before the next one can be run.
* **Better performance :–** It enables the better performance by the operating system. When one application uses only the processor and another application uses only the disk drive then the time to run both applications concurrently to completion will be shorter than the time to run each application consecutively.

### Drawbacks of Concurrency :-

* It is required to protect multiple applications from one another.
* It is required to coordinate multiple applications through additional mechanisms.
* Additional performance overheads and complexities in operating systems are required for switching among applications.
* Sometimes running too many applications concurrently leads to severely degraded performance.

### -------------------------------------------------------------------------------------------------

* 1. **What is Critical section ?**

The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

A diagram that demonstrates the critical section is as follows −

In the above diagram, the entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

### Solution to the Critical Section Problem:-

The critical section problem needs a solution to synchronize the different processes. The solution to the critical section problem must satisfy the following conditions −

* + - **Mutual Exclusion:-** Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.
    - **Progress:-** Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.
    - **Bounded Waiting:-** Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

### What is Semaphores ? Explain Types of Semaphores and Advantages , Dis- advantages of semaphores ?

**Ans:-** Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

The definitions of wait and signal are as follows −

### Wait

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

wait(S)

{

while (S<=0); S--;

}

### Signal

The signal operation increments the value of its argument S.

signal(S)

{

S++;

}

**Types of Semaphores:-** There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows −

* + **Counting Semaphores:-** These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources. If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.
  + **Binary Semaphores:-**The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

### Advantages of Semaphores:-

* + Semaphores allow only one process into the critical section. They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.
  + There is no resource wastage because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
  + Semaphores are implemented in the machine independent code of the microkernel. So they are machine independent.

### Disadvantages of Semaphores:-

* + Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
  + Semaphores are impractical for last scale use as their use leads to loss of modularity. This happens because the wait and signal operations prevent the creation of a structured layout for the system.
  + Semaphores may lead to a priority inversion where low priority processes may access the critical section first and high priority processes later.

### What is Inter Process Communication?

**Ans:-** Inter process communication (IPC) is used for exchanging data between multiple threads in one or more processes or programs. The Processes may be running on single or multiple computers connected by a network.

It is a set of programming interface which allow a programmer to coordinate activities among various program processes which can run concurrently in an operating system. This allows a specific program to handle many user requests at the same time.

Advantages of interprocess communication:-

* Inter process communication allows one application to manage another .
* Inter process communication helps send messages efficiently between processes.
* The program is easy to maintain and debug because it is divided into different sections of code that work separately.
* Programmers can perform a variety of other tasks at the same time, including Editing, listening to music, compiling, etc.
* Data can be shared between different programs at the same time.
* Tasks can be subdivided and run on special types of processors. You can then exchange data via IPC.

Disadvantages of inter process communication:-

* The program cannot write to similar locations.
* Processes or programs that use the shared memory model must make sure that they are not writing to similar memory locations.
* It’s slower than a direct function call.

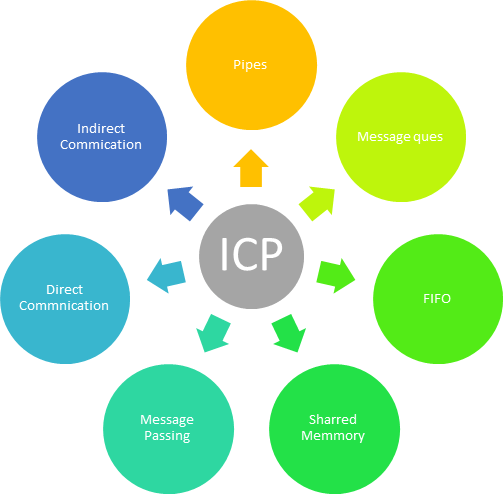
### Applications:-

* Inter-process communication is mainly used to exchange data in between several threads within a single or multiple programs/processes.
* Inter process communication is one type of mechanism that allows programs to interact with each other & synchronize their actions
* IPC is essential in the design process of micro kernels & nano kernels because Micro kernels decrease different functionalities which are provided through the kernel. After that, these functionalities are attained by interacting with servers through IPC.

### Write about Methods (or)Approaches for IPC (Inter Process Communication)?

**Ans:-** Here, are few important methods for inter process communication:

**Pipes:**- Pipe is widely used for communication between two related processes. This is a half-duplex method, so the first process communicates with the second process. However, in order to achieve a full-duplex, another pipe is needed.



**Message Passing:-**It is a mechanism for a process to communicate and synchronize. Using message passing, the process communicates with each other without resorting to shared variables. IPC mechanism provides two operations:

* + Send (message)- message size fixed or variable
  + Received (message)

**Message Queues:-** A message queue is a linked list of messages stored within the kernel. It is identified by a message queue identifier. This method offers communication between single or multiple processes with full-duplex capacity.

**Direct Communication:-** In this type of inter-process communication process, should name each other explicitly. In this method, a link is established between one pair of communicating processes, and between each pair, only one link exists.

**Indirect Communication:-** Indirect communication establishes like only when processes share a common mailbox each pair of processes sharing several communication links. A link can communicate with many processes. The link may be bi-directional or unidirectional.

**Shared Memory:-** Shared memory is a memory shared between two or more processes that are established using shared memory between all the processes. This type of memory requires to protected from each other by synchronizing access across all the processes.

**FIFO:-** Communication between two unrelated processes. It is a full-duplex method, which means that the first process can communicate with the second process, and the opposite can also happen.

### Explain Process Synchronization?

Ans:- An operating system is a software that manages all applications on a device and basically helps in the smooth functioning of our computer. Because of this reason, the operating system has to perform many tasks, and sometimes simultaneously. This isn't usually a problem unless these simultaneously occurring processes use a common resource.

**Solutions To The Critical Section Problem:-**

**Peterson's solution** :- Peterson's approach to critical section problems is extensively utilized. It is a classical software-based solution.

The solution is based on the idea that when a process is executing in a critical section, then the other process executes the rest of the code and vice-versa is also possible, i.e., this solution makes sure that only one process executes the critical section at any point in time.

**Lock Variable Mechanism:-** A lock variable provides the simplest synchronization

mechanism for processes. Some noteworthy points regarding Lock Variables are-

1. Its a **software mechanism** implemented in user mode, i.e. no support required from the Operating System.
2. Its a busy waiting solution (keeps the CPU busy even when its technically waiting).
3. It can be used for more than two processes.

When Lock = 0 implies critical section is vacant (initial value ) and Lock = 1 implies critical section occupied.

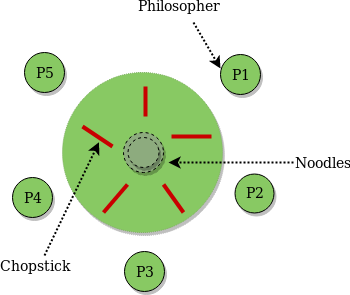
**Semaphores:-** A semaphore is a signaling mechanism, and a process can signal a process that is waiting on a semaphore. This differs from a mutex in that the mutex can only be notified by the process that sets the shared lock. Semaphores make use of the wait() and signal() functions for synchronization among the processes.

### Explain about Classical Process Synchronization Problems?

**Ans:-** These problems are used for testing nearly every newly proposed synchronization scheme. Classical problems are 4 types . They are :

1. Bounded-buffer (or Producer-Consumer) Problem,
2. Dining-Philosophers Problem,
3. Readers and Writers Problem,
4. Sleeping Barber Problem

These are summarized, for detailed explanation, you can view the linked articles for each.

* [**Bounded-buffer (or Producer-Consumer) Problem**](https://www.geeksforgeeks.org/producer-consumer-solution-using-semaphores-java/)**:**- Bounded Buffer problem is also called producer consumer problem. This problem is generalized in terms of the Producer-Consumer problem. Solution to this problem is, creating two counting semaphores “full” and “empty” to keep track of the current number of full and empty buffers respectively. Producers produce a product and consumers consume the product, but both use of one of the containers each time. [**Dining-Philosophers Problem**](https://www.geeksforgeeks.org/operating-system-dining-philosopher-problem-using-semaphores/)**:-**The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pickup the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.
* This problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation- free manner.
* [**Readers and Writers Problem**](https://www.geeksforgeeks.org/readers-writers-problem-set-1-introduction-and-readers-preference-solution/)**:**- Suppose that a database is to be shared among several concurrent processes. Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database. We distinguish between these two types of processes by referring to the former as readers and to the latter as writers. Precisely in OS we call this situation as the readers-writers problem. Problem parameters:
  + One set of data is shared among a number of processes.
  + Once a writer is ready, it performs its write. Only one writer may write at a time.
  + If a process is writing, no other process can read it.
  + If at least one reader is reading, no other process can write.
  + Readers may not write and only read.
* [**Sleeping Barber Problem**](https://www.geeksforgeeks.org/operating-system-sleeping-barber-problem/)**:-** Barber shop with one barber, one barber chair and N chairs to wait in. When no customers the barber goes to sleep in barber chair and must be woken when a customer comes in. When barber is cutting hair new customers take empty seats to wait, or leave if no vacancy.

### Explain about Producer and Consumer Problem ?

**Ans:-** The Producer-Consumer problem is a classic problem this is used for multi- process synchronization i.e. synchronization between more than one processes.

In the producer-consumer problem, there is one Producer that is producing something and there is one Consumer that is consuming the products produced by the Producer. The producers and consumers share the same memory buffer that is of fixed-size.

The job of the Producer is to generate the data, put it into the buffer, and again start generating data. While the job of the Consumer is to consume the data from the buffer.

### Problem :-

* + The producer should produce data only when the buffer is not full. If the buffer is full, then the producer shouldn't be allowed to put any data into the buffer.
  + The consumer should consume data only when the buffer is not empty. If the buffer is empty, then the consumer shouldn't be allowed to take any data from the buffer.
  + The producer and consumer should not access the buffer at the same time.

### Solution:-

The above three problems can be solved with the help of semaphores(learn more about semaphores from [here](https://afteracademy.com/blog/what-is-semaphore-and-what-are-its-types)).

In the producer-consumer problem, we use three semaphore variables:

1. **Semaphore S:** This semaphore variable is used to achieve mutual exclusion between processes. By using this variable, either Producer or Consumer will be allowed to use or access the shared buffer at a particular time. This variable is set to 1 initially.
2. **Semaphore E:** This semaphore variable is used to define the empty space in the buffer. Initially, it is set to the whole space of the buffer i.e. "n" because the buffer is initially empty.
3. **Semaphore F:** This semaphore variable is used to define the space that is filled by the producer. Initially, it is set to "0" because there is no space filled by the producer initially.

By using the above three semaphore variables and by using the *wait()* and *signal()* function, we can solve our problem(the *wait()* function decreases the semaphore variable by 1 and the *signal()* function increases the semaphore variable by 1). So. let's see how.

The following is the pseudo-code for the producer:

void producer() {

while(T) {

produce()

wait(E)

wait(S)

append()

signal(S)

signal(F)

}

}

The above code can be summarized as:

* + ***while()*** is used to produce data, again and again, if it wishes to produce, again and again.
  + ***produce()*** function is called to produce data by the producer.
  + ***wait(E)*** will reduce the value of the semaphore variable "E" by one i.e. when the producer produces something then there is a decrease in the value of the empty space in the buffer. If the buffer is full i.e. the vale of the semaphore variable "E" is "0", then the program will stop its execution and no production will be done.
  + ***wait(S)*** is used to set the semaphore variable "S" to "0" so that no other process can enter into the critical section.
  + ***append()*** function is used to append the newly produced data in the buffer.
  + ***signal(s****)* is used to set the semaphore variable "S" to "1" so that other processes can come into the critical section now because the production is done and the append operation is also done.
  + ***signal(F****)* is used to increase the semaphore variable "F" by one because after adding the data into the buffer, one space is filled in the buffer and the variable "F" must be updated.

This is how we solve the produce part of the producer-consumer problem. Now, let's see the consumer solution. The following is the code for the consumer:

void consumer() {

while(T) {

wait(F)

wait(S)

take()

signal(S)

signal(E)

use()

} }

The above code can be summarized as:

* + *while()* is used to consume data, again and again, if it wishes to consume, again and again.
  + *wait(F)* is used to decrease the semaphore variable "F" by one because if some data is consumed by the consumer then the variable "F" must be decreased by one.
  + *wait(S)* is used to set the semaphore variable "S" to "0" so that no other process can enter into the critical section.
  + *take()* function is used to take the data from the buffer by the consumer.
  + *signal(S)* is used to set the semaphore variable "S" to "1" so that other processes can come into the critical section now because the consumption is done and the take operation is also done.
  + *signal(E)* is used to increase the semaphore variable "E" by one because after taking the data from the buffer, one space is freed from the buffer and the variable "E" must be increased.
  + use() is a function that is used to use the data taken from the buffer by the process to do some operation.

### Discuss about Reader and Writer Problem ?

Ans:- The readers-writers problem relates to an object such as a file that is shared between multiple processes. Some of these processes are readers i.e. they only want to read the data from the object and some of the processes are writers i.e. they want to write into the object.

The readers-writers problem is used to manage synchronization.

**Example:-** If two readers access the object at the same time there is no problem. However if two writers or a reader and writer access the object at the same time, there may be problems. To solve this situation, a writer should get exclusive access to an object i.e. when a writer is accessing the object, no reader or writer may access it. However, multiple readers can access the object at the same time.

This can be implemented using semaphores. The codes for the reader and writer process in the reader-writer problem are given as follows −

**Reader Process :-** The code that defines the reader process is given below –

wait (mutex); rc ++;

if (rc == 1) wait (wrt); signal(mutex);

.**READ THE OBJECT**

.wait(mutex);

rc --;

if (rc == 0)

signal (wrt); signal(mutex);

In the above code, mutex and wrt are semaphores that are initialized to 1. Also, rc is a variable that is initialized to 0. The mutex semaphore ensures mutual exclusion and wrt handles the writing mechanism and is common to the reader and writer process code.

The variable rc denotes the number of readers accessing the object. As soon as rc becomes 1, wait operation is used on wrt. This means that a writer cannot

access the object anymore. After the read operation is done, rc is decremented. When re becomes 0, signal operation is used on wrt. So a writer can access the object now.

**Writer Process:-** The code that defines the writer process is given below:

wait(wrt);

.. WRITE INTO THE OBJECT

.signal(wrt);

If a writer wants to access the object, wait operation is performed on wrt. After that no other writer can access the object. When a writer is done writing into the object, signal operation is performed on wrt.

END

### Unit- 4

1. **What is Memory Management in OS?**

**Ans:- Memory Management** is the process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of the system.

It is the most important function of an operating system that manages primary memory. It helps processes to move back and forward between the main memory and execution disk. It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.

### Uses:-

* + It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.
  + Tracks whenever inventory gets freed or unallocated. According to it will update the status.
  + It allocates the space to application routines.
  + It also make sure that these applications do not interfere with each other.
  + Helps protect different processes from each other
  + It places the programs in memory so that memory is utilized to its full extent.

### Memory Management Techniques:-

Here, are some most crucial memory management techniques :-

### Single Contiguous Allocation:-

It is the easiest memory management technique. In this method, all types of computer’s memory except a small portion which is reserved for the OS is available for one application.

**Example:-** MS-DOS operating system allocates memory in this way. An embedded system also runs on a single application.

### Partitioned Allocation:-

It divides primary memory into various memory partitions, which is mostly contiguous areas of memory. Every partition stores all the information for a specific task or job. This method consists of allotting a partition to a job when it starts & unallocated when it ends.

### Paged Memory Management:-

This method divides the computer’s main memory into fixed-size units known as page frames. This hardware memory management unit maps pages into frames which should be allocated on a page basis.

### Segmented Memory Management:-

Segmented memory is the only memory management method that does not provide the user’s program with a linear and contiguous address space.

Segments need hardware support in the form of a segment table. It contains the physical address of the section in memory, size, and other data like access protection bits and status.

### Swapping :-

Swapping is a method in which the process should be swapped temporarily from the main memory to the backing store. It will be later brought back into the memory for continue execution.

1. Memory allocation is a process by which computer programs are assigned memory or space.
2. Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages.
3. Processes are stored and removed from memory, which creates free memory space, which are too small to use by other processes.
4. Segmentation method works almost similarly to paging. The only difference between the two is that segments are of variable-length, whereas, in the paging method, pages are always of fixed size.
5. Dynamic loading is a routine of a program which is not loaded until the program calls it.
6. Linking is a method that helps OS to collect and merge various modules of code and data into a single executable file

### Write about Physical and Virtual Address Space ? Ans:- Logical Address:-

It is a virtual address generated by the CPU while a program is running. It is referred to as a virtual address because it does not exist physically. Using this address, the CPU access the actual address or physical address inside the memory, and data is fetched from there.

The hardware device called Memory Management Unit (MMU) is used for mapping this logical address to the physical address. The set of all logical addresses generated by the CPU for a program is called the logical address space.

#### Physical Address

Physical Address is the actual address of the data inside the memory. The logical address is a virtual address and the program needs physical memory for its execution. The user never deals with the Physical Address. The user program generates the logical address and is mapped to the physical address by the Memory Management Unit(MMU).

The set of all physical addresses corresponding to the logical addresses in the logical address space is called the physical address space.

### Differences between Logical Address and Physical Address :

1. The Logical Address is generated by the CPU while the program is running and the Physical Address is the location inside the main memory.
2. The Logical Address is Virtual and the Physical Address is the actual address of the memory.
3. The Logical Address is generated by the CPU and the Physical Address is calculated by MMU.
4. Users can't view the Logical Address of a program whereas the Physical Address is visible to the user.
5. The set of all logical addresses generated by the CPU in reference to a program is called the Logical Address Space and the set of all physical addresses corresponding to the logical addresses in the Logical Address Space is called the Physical Address Space.

***Logical Address***

It is a virtual address generated by the CPU while a program is running. It is referred to as a virtual address because it does not exist physically. Using this

address, the CPU access the actual address or physical address inside the memory, and data is fetched from there.

The hardware device called Memory Management Unit (MMU) is used for mapping this logical address to the physical address. The set of all logical addresses generated by the CPU for a program is called the logical address space.

**3. Write a Short notes on Memory Allocation strategies ?**

**Ans:-** Memory allocation is an action of assigning the physical or the virtual memory address space to a process (its instructions and data). The two fundamental methods of memory allocation are static and dynamic memory allocation.

The static memory allocation method assigns the memory to a process, before its execution. But, the dynamic memory allocation method assigns the memory to a process, during its execution.

### Types of Memory Allocations :-

1. **Static Memory Allocation:-**

Static memory allocation is performed when the compiler compiles the program and generates object files. The linker merges all these object files and creates a single executable file. The loader loads this single executable file in the main memory, for execution. In this process, the size of the data required by the process must be known before the execution of the process initiates.

If the data sizes are not known before the execution of the process, then they have to be guessed. If the data size guessed is larger than the required, then it leads to wastage of memory. If the guessed size is smaller, then it leads to inappropriate execution of the process.

The static memory allocation method does not need any memory allocation operation during the execution of the process. All the memory allocation operation required for the process is done before the execution of the process has started. So, it leads to faster execution of a process.

Static memory allocation provides more efficiency when compared to dynamic memory allocation.

### Dynamic Memory Allocation:-

Dynamic memory allocation is performed while the program is in execution. Here, the memory is allocated to the entities of the program when they are to be used for the first time while the program is running.

The actual size, of the data required, is known at the run time so, it allocates the exact memory space to the program there by, reducing the memory wastage.

Dynamic memory allocation provides flexibility to the execution of the program. As it can decide what amount of memory space will be required by the program. If the program is large enough then a dynamic memory allocation is performed on the different parts of the program, which is to be used currently. This reduces memory wastage and improves the performance of the system.

Allocating memory dynamically creates an overhead over the system. Some allocation operations are performed repeatedly during the program execution creating more overheads, leading in slow execution of the program.

Dynamic memory allocation does not require special support from the operating system. It is the responsibility of the programmer to design the program in a way to take advantage of dynamic memory allocation method.

Thus the dynamic memory allocation is flexible but slower than static memory allocation

### Define Static Memory Allocation ? Explain it’s Advantages & Dis- Advantages?

**Ans:-** Static memory allocation is performed when the compiler compiles the program and generates object files. The linker merges all these object files and creates a single executable file. The loader loads this single executable file in the main memory, for execution. In static memory allocation, the size of the data required by the process must be known **before** the execution of the process initiates.

### Advantages :-

1. Static memory allocation provides an efficient way of assigning the memory to a process.
2. All the memory assigning operations are done before the execution starts. So, there are *no* overheads of memory allocation operations at the time of execution of the program.
3. Static memory allocation provides faster execution, as at the time of execution it doesn’t have to waste time in allocation memory to the program.

### Dis- Advantages:-

1. In static memory allocation, the system is unaware of the memory requirement of the program.
2. Static memory allocation leads to memory wastage**.** As it estimates the size of memory required by the program. So, if the estimated size is larger, it will lead to memory wastage else if the estimated size is smaller, then the program will execute inappropriately**.**

### Define Dynamic Memory Allocation ? Explain it’s Advantages and Dis- Advantages ?

**Ans:-** Dynamic memory allocation is performed while the program is in execution. Here, the memory is allocated to the entities of the program when they are to be used for the first time while the program is running.

### Advantages :-

1. Dynamic memory allocation provides a flexible way of assigning the memory to a process.
2. Dynamic memory allocation reduces the memory wastage as it assigns memory to a process during the execution of that program. So, it is aware of the exact memory size required by the program.
3. If the program is large then the dynamic memory allocation is performed on the different parts of the program. Memory is assigned to the part of a program that is currently in use. This also reduces memory wastage and indeed improves system performance.

### Dis- Advantages:-

1. Dynamic memory allocation method has an overhead of assigning the memory to a process during the time of its execution.
2. Sometimes the memory allocation actions are repeated several times during the execution of the program which leads to more overheads.
3. The overheads of memory allocation at the time of its execution slowdowns **t**he execution to some extent.

### Explain Variable Partitions?

**Ans:-** In operating systems, Memory Management is the function, responsible for allocating and managing computer’s main memory. Memory Management

function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.

There are 2 Memory Management Techniques:

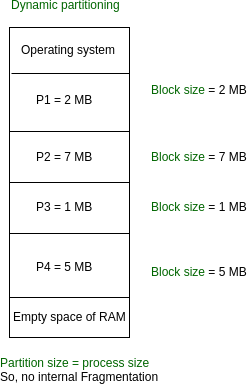
1) **Contiguous** 2) **Non-Contiguous**.

In Contiguous Technique, executing process must be loaded entirely in main- memory. Contiguous Technique can be divided into:

1. [Fixed (or static) partitioning](https://www.geeksforgeeks.org/fixed-or-static-partitioning-in-operating-system/)
2. Variable (or dynamic) partitioning

### Variable Partitioning: –

It is a part of Contiguous allocation technique. It is used to all deviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configure. Various **features** associated with variable Partitioning-

1. Initially RAM is empty and partitions are made during the run-time according to process’s need instead of partitioning during system configure.
2. The size of partition will be equal to incoming process.
3. The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.
4. Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory’s size.

There are some advantages and disadvantages of variable partitioning over fixed partitioning as given below.

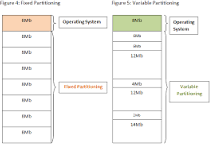
### Advantages of Variable Partitioning –

1. **No Internal Fragmentation:-** In variable Partitioning, space in main memory is allocated strictly according to the need of process, hence there is no

case of internal fragmentation. There will be no unused space left in the partition.

1. **No restriction on Degree of Multiprogramming:-**More number of processes can be accommodated due to absence of internal fragmentation. A process can be loaded until the memory is empty.
2. **No Limitation on the size of the process:-**In Fixed partitioning, the process with the size greater than the size of the largest partition could not be loaded and process can not be divided as it is invalid in contiguous allocation technique. Here, In variable partitioning, the process size can’t be restricted since the partition size is decided according to the process size.

**2. fixed size partition:-** Fixed partitioning creates memory partitions of identical sizes. Variable partitioning offers more flexibility because it creates partitions of varying sizes but still results in memory fragmentation. Dynamic partitioning allocates partitions of the required sizes requested by the programs respectively.25-Jan-2022



### Explain about Paging?

**Ans:-** Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

### Advantages of Paging :-

Here, are advantages of using Paging method:

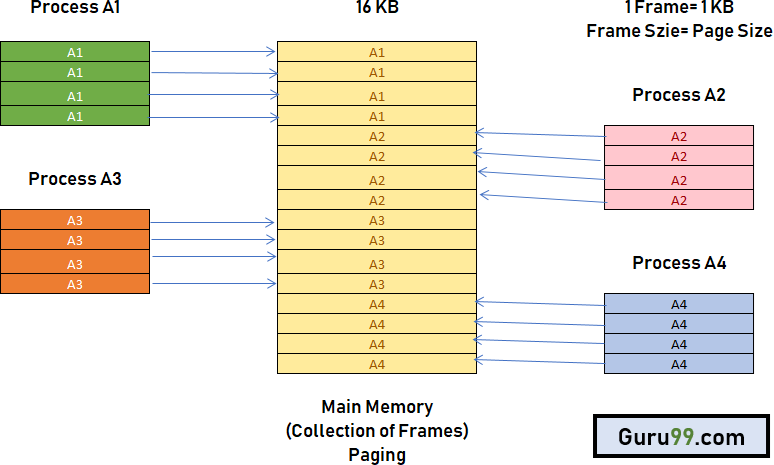
* + Easy to use [memory management](https://www.guru99.com/os-memory-management.html) algorithm
  + No need for external Fragmentation
  + Swapping is easy between equal-sized pages and page frames.

### Disadvantages of Paging:-

* + May cause Internal fragmentation
  + Page tables consume additonal memory.
  + Multi-level paging may lead to memory reference overhead.

### Example of paging in OS:-

**Ex:-** For example, if the main memory size is 16 KB and Frame size is 1 KB. Here, the main memory will be divided into the collection of 16 frames of 1 KB each.

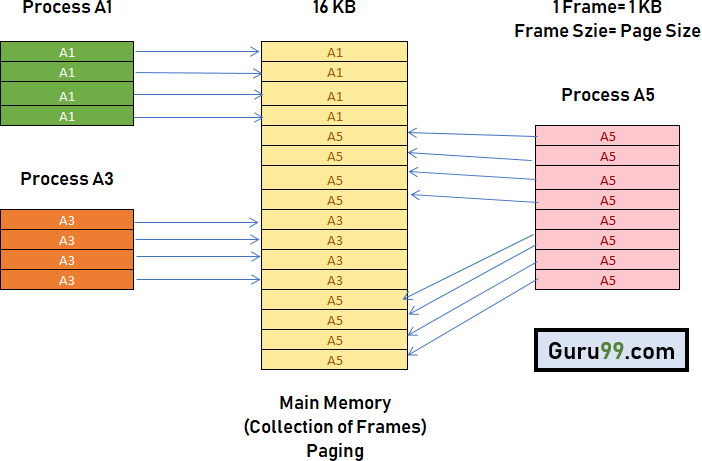


There are 4 separate processes in the system that is A1, A2, A3, and A4 of 4 KB each. Here, all the processes are divided into pages of 1 KB each so that operating system can store one page in one frame.

At the beginning of the process, all the frames remain empty so that all the pages of the processes will get stored in a contiguous way.

In this example you can see that A2 and A4 are moved to the waiting state after some time. Therefore, eight frames become empty, and so other pages can be loaded in that empty blocks. The process A5 of size 8 pages (8 KB) are waiting in the ready queue.

In this example you can see that A2 and A4 are moved to the waiting state after some time. Therefore, eight frames become empty, and so other pages can be loaded in that empty blocks. The process A5 of size 8 pages (8 KB) are waiting in the ready queue.



In this example, you can see that there are eight non-contiguous frames which is available in the memory, and paging offers the flexibility of storing the process at the different places. This allows us to load the pages of process A5 instead of A2 and A4.

### Explain about Segmentation?

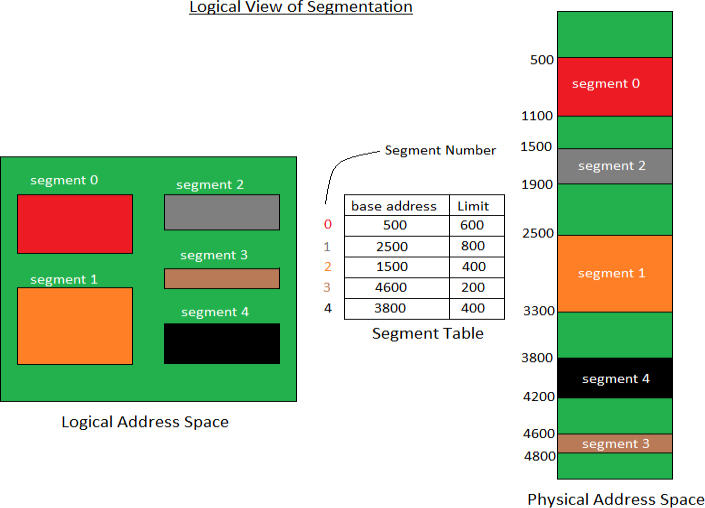
**Ans:-** A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the same sizes are called segments. Segmentation gives user’s view of the process which paging does not give. Here the user’s view is mapped to physical memory.

### Types of segmentation:-

1. **Virtual memory segmentation :–** Each process is divided into a number of segments, not all of which are resident at any one point in time.
2. **Simple segmentation: –**Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

**Segment Table:-–** It maps two-dimensional Logical address into one- dimensional Physical address. It’s each table entry has:

* **Base Address:-** It contains the starting physical address where the segments reside in memory.
* **Limit:-** It specifies the length of the segment.

### Advantages of Segmentation: –

* No Internal fragmentation.
* Segment Table consumes less space in comparison to Page table in paging.

### Disadvantage of Segmentation: –

* As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.
* segmentation technique is expensive

In Segmentation, it is tough to allocate memory in a contiguous manner to a variable sized partition.

### Explain about virtual memory ?

**Ans:-** Virtual Memory is a storage mechanism which offers user an illusion of having a very big main memory. It is done by treating a part of secondary memory as the main memory. In Virtual memory, the user can store processes with a bigger size than the available main memory.

Therefore, instead of loading one long process in the main memory, the OS loads the various parts of more than one process in the main memory. Virtual memory is mostly implemented with demand paging and demand segmentation.

### Reasons for using virtual memory:-

* + Whenever your computer doesn’t have space in the physical memory it writes what it needs to remember to the hard disk in a swap file as virtual memory.
  + If a computer running Windows needs more memory/RAM, then installed in the system, it uses a small portion of the hard drive for this purpose.

### Virtual Memory Working Process :-

* + In the modern world, virtual memory has become quite common these days. It is used whenever some pages require to be loaded in the main memory for the execution, and the memory is not available for those many pages.
  + So, in that case, instead of preventing pages from entering in the main memory, the OS searches for the RAM space that are minimum used in the recent times or that are not referenced into the secondary memory to make the space for the new pages in the main memory.
  + Let’s understand virtual memory management with the help of one example.

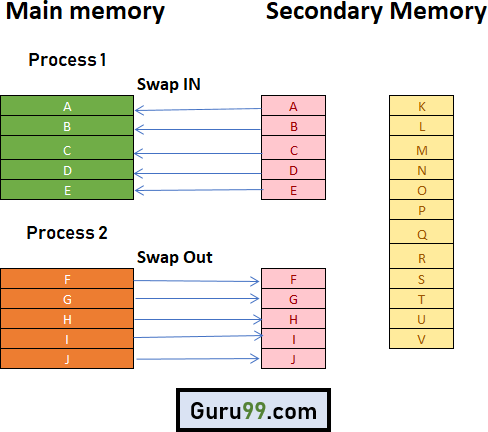
### For example:-

Let’s assume that an OS requires 300 MB of memory to store all the running programs. However, there’s currently only 50 MB of available physical memory stored on the RAM.

* + The OS will then set up 250 MB of virtual memory and use a program called the Virtual Memory Manager(VMM) to manage that 250 MB.
  + So, in this case, the VMM will create a file on the hard disk that is 250 MB in size to store extra memory that is required.
  + The OS will now proceed to address memory as it considers 300 MB of real memory stored in the RAM, even if only 50 MB space is available.
  + It is the job of the VMM to manage 300 MB memory even if just 50 MB of real memory space is available.

### Demand Paging:-

A demand paging mechanism is very much similar to a paging system with swapping where processes stored in the secondary memory and pages are loaded only on demand, not in advance. So, the OS never copy any of the old program’s pages from the disk or any of the new program’s pages into the main memory. During the program execution, if the program references as it will start



page that may not be available in the main memory because it was swapped, then the processor considers it as an invalid memory reference. That’s because the page fault and transfers send control back from the program to the OS, which demands to store page back into the memory.

END

# Unit -5

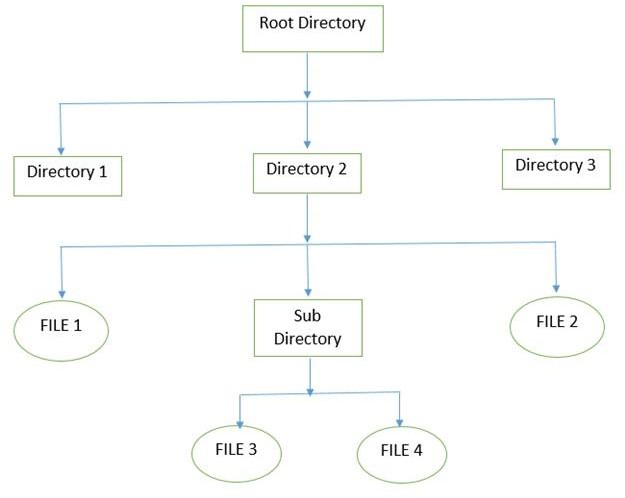
### Explain File Management in Operating System?

**Ans:-** File management is one of the basic and important feature of operating system. Operating system is used to manage files of computer system. All the files with different extensions are managed by operating system.

A file is collection of specific information stored in the memory of computer system. File management is defined as the process of manipulating files in computer system, it management includes the process of creating, modifying and deleting the files.

The following are some of the tasks performed by file management of operating system of any computer system:

* 1. It helps to create new files in computer system and placing them at the specific locations.
  2. It helps in easily and quickly locating these files in computer system.



* 1. It makes the process of sharing of the files among different users very easy and user friendly.
  2. It helps to stores the files in separate folders known as directories. These directories help users to search file quickly or to manage the files according to their types or uses.
  3. It helps the user to modify the data of files or to modify the name of the file in the directories.

The above figure shows the general hierarchy of the storage in an operating system. In this figure the root directory is present at the highest level in the hierarchical structure. It includes all the subdirectories in which the files are stored. Subdirectory is a directory present inside another directory in the file storage system.

The directory base storage system ensures better organization of files in the memory of the computer system.

### The file management of functions based on the following concepts:-

1. **File Attributes:-** It specifies the characteristics of the files such as type, date of last modification, size, location on disk etc. file attributes help the user to understand the value and location of files. File attributes is one most important feature. It is uses to describe all the information regarding particular file.
2. **File Operations :-** It specifies the task that can be performed on a file such as opening and closing of file.
3. **File Access permission:-** It specifies the access permissions related to a file such as read and write.
4. **File Systems:-** It specifies the logical method of file storage in a computer system. Some of the commonly used files systems include FAT and NTFS.

### Explain in detail I/O Management ?

**Ans:-**The Primary role of the operating System in computer Input /Output is to manage and Organize I/O operations and all I/O devices.

### Over view of Input /Output System:-

These methods form the I/O sub-system of the kernel of OS that separates the rest of the kernel from the complications of managing I / O devices.

### I/O Hardware :-

Computers operate many huge kinds of devices. The general categories of storage devices are like disks, tapes, transmission devices (like network interface cards, modems) and human interface devices (like screen, keyboard, etc.).

A device communicates with the operating system of a computer by transferring signals over cable. The peripheral devices communicate with the machine through a connection point also called ports— (one example is a serial port). When devices use a set of wires or cables, that connecting cables are called a "bus." A bus is a collection of wires and a firmly defined protocol which specifies a set of messages that can be sent on the wires.

### Operating System Using I/O Port:-

An I/O port usually consists of four different registers. These are (1) status, (2) control, (3) data-in, and (4) data-out registers.

* The data-in register is read by the host for getting input.
* The data-out register is written by the host for sending output.
* The status register holds bits which can be read by the host.
* The control register is written by the host for starting a command or for changing the mode of any device.
* The data registers are usually 1 to 4 bytes in size. Some of the controllers have FIFO chips which hold several bytes of input or output data for expanding the capacity of the controller beyond the size of the data register.

### Explain Operating System Security ?

**Ans:-** Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause several damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. These methods are 2.

1) Authentication 2) One Time passwords

### Authentication:-

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −

**Username / Password** :− User need to enter a registered username and password with Operating system to login into the system.

**User card/key** :− User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.

**User attribute - fingerprint/ eye retina pattern/ signature:**− User need to pass his/her attribute via designated input device used by operating system to login into the system.

### One Time passwords:-

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.

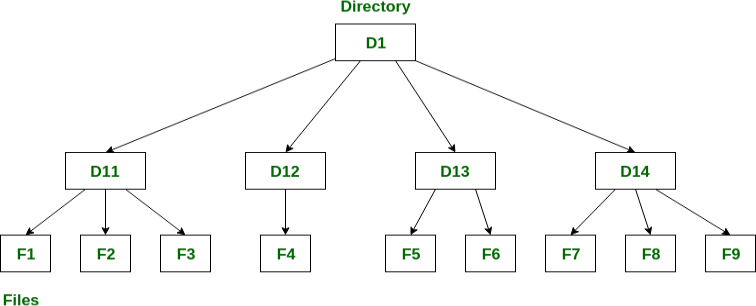
**Random numbers: −** Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.

**Secret key** :− User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.

**Network password** :− Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

### 4) Write about Directory Structure ?

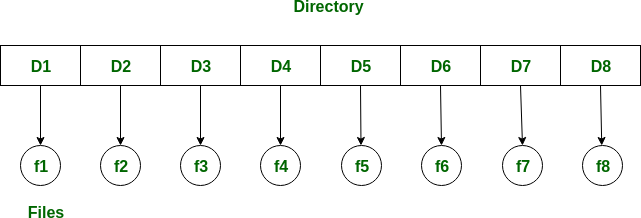
**Ans:-** Directory is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner.



There are several logical structures of a directory, these are given below.

1. **Single-level directory:-** The single-level directory is the simplest directory structure. In it, all files are contained in the same directory which makes it easy to support and understand.

A single level directory has a significant limitation, When the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a unique name. if two users call their dataset test, then the unique name rule violated.



### Advantages:-

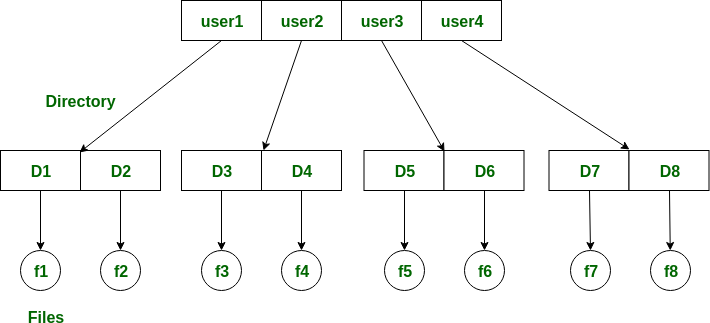
* It is a single directory, so its implementation is very easy.
* If the files are smaller in size, searching will become faster.
* The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

### Disadvantages:-

* Searching will become time taking if the directory is large.
* This can not group the same type of files together.

1. **Two-level directory:–** A single level directory often leads to confusion of files names among different users. The solution to this problem is to create a separate directory for each user.

In the two-level directory structure, each user has their own user files directory (UFD). The UFDs have similar structures, but each lists only the files of a single user. system’s master file directory (MFD) is searches whenever a new user id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.



### Advantages: -

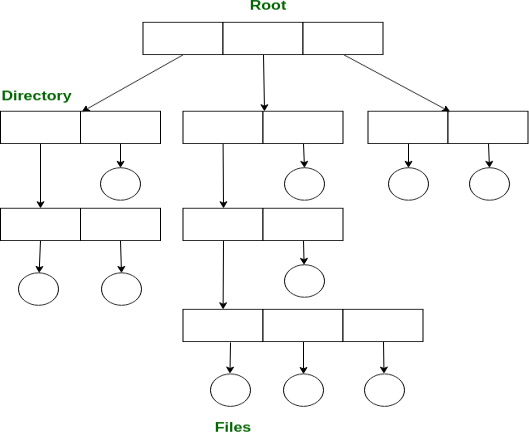
* We can give full path like /User-name/directory-name/.
* Different users can have the same directory as well as the file name.
* Searching of files becomes easier due to pathname and user-grouping.

### Disadvantages:-

* A user is not allowed to share files with other users.
* Two files of the same type cannot be grouped together in the same user.

**Tree-structured directory:–** Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.

This generalization allows the user to create their own subdirectories and to organize their files accordingly.



A tree structure is the most common directory structure. The tree has a root directory, and every file in the system has a unique path.

### Advantages:-

* Very general, since full pathname can be given.
* Very scalable, the probability of name collision is less.
* Searching becomes very easy, we can use both absolute paths as well as relative.

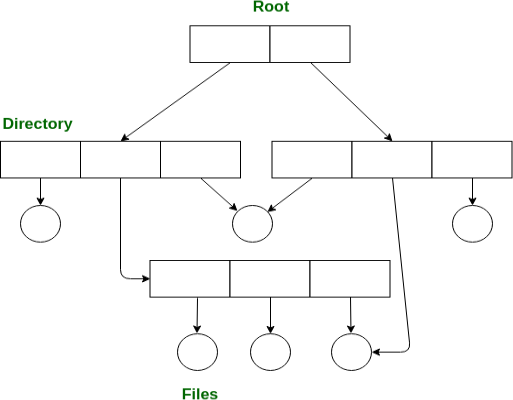
### Disadvantages:-

* Every file does not fit into the hierarchical model, files may be saved into multiple directories.
* We can not share files.
* It is inefficient, because accessing a file may go under multiple directories.

**Acyclic graph directory:–** An acyclic graph is a graph with no cycle and allows us to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.

It is used in the situation like when two programmers are working on a joint project and they need to access files. The associated files are stored in a subdirectory, separating them from other projects and files of other programmers since they are working on a joint project so they want the subdirectories to be into their own directories. The common subdirectories should be shared. So here we use Acyclic directories.

It is the point to note that the shared file is not the same as the copy file. If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.



### Advantages:-

* We can share files.
* Searching is easy due to different-different paths.

### Disadvantages:-

* We share the files via linking, in case deleting it may create the problem,
* If the link is a soft link then after deleting the file we left with a dangling pointer.
* In the case of a hard link, to delete a file we have to delete all the references associated with it.

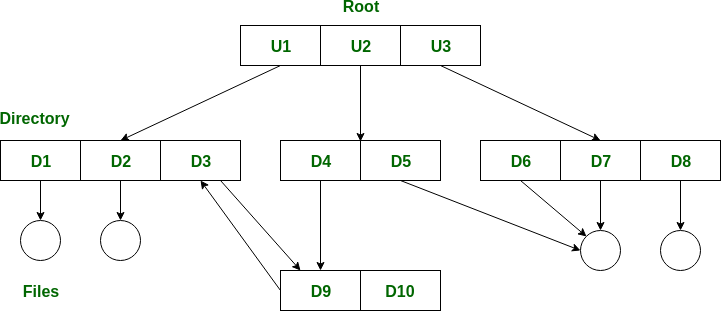
**General graph directory structure:–** In this graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.

The main problem with this kind of directory structure is to calculate the total size or space that has been taken by the files and directories. **Advantages:**-

* It allows cycles.
* It is more flexible than other directories structure.

### Disadvantages:-

* It is more costly than others.
* It needs garbage collection.



### Write A Short note on File Operations ?

**Ans:-** File Operations Files exist to store information and allow it to be retrieved later. Different systems provide different operations to allow storage and retrieval.

1. **Create.** The file is created with no data. The purpose of the call is to announce that the file is coming and to set some of the attributes.
2. **Delete**. When the file is no longer needed, it has to be deleted to free up disk space.
3. **Open**. The purpose of the open call is to allow the system to fetch the attributes and list of disk addresses into main memory for rapid access on later calls.
4. **Close**. When all the accesses are finished, the attributes and disk addresses are no longer needed, so the file should be closed to free up some internal table space. 5. **Read**. Data are read from file. Usually, the bytes come from the current position. The caller must specify how much data are needed and must also provide a buffer to put them in.
5. **Write.** Data are written to the file, again, usually at the current position. If the current position is the end of the file, the file's size increases. If the current position is in the middle of the file, existing data are overwritten and lost forever.
6. **Append**. It can only add data to the end of the file. Systems that provide a minimal set of system calls do not generally have append, but many systems provide multiple ways of doing the same thing, and these systems sometimes have append.
7. **Seek.** For random access files, a method is needed to specify from where to take the data. One common approach is a system call, seek, that repositions the file pointer to a specific place in the file. After this call has completed, data can be read from, or written to, that position.
8. **Get** attributes Processes often need to read file attributes to do their work. For example, the UNIX make program is commonly used to manage software development projects consisting of many source files. When make is called, it examines the modification times of all the source and object files and arranges for the minimum number of compilations required to bring everything up to date. To do its job, it must look at the attributes, namely, the modification times.7
9. **Set** attributes. Some of the attributes are user settable and can be changed after the file has been created. This system call makes that possible.
10. **Rename.** It frequently happens that a user needs to change the name of an existing file. This system call makes that possible. It is not always strictly necessary, because the file can usually be copied to a new file with the new name, and the old file then deleted.
11. **Lock.** Locking a file or a part of a file prevents multiple simultaneous access by different process. For an airline reservation system, for instance, locking the database while making a reservation prevents reservation of a seat for two different travelers.

### Explain in detail File Allocation Methods ?

**Ans:-** The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

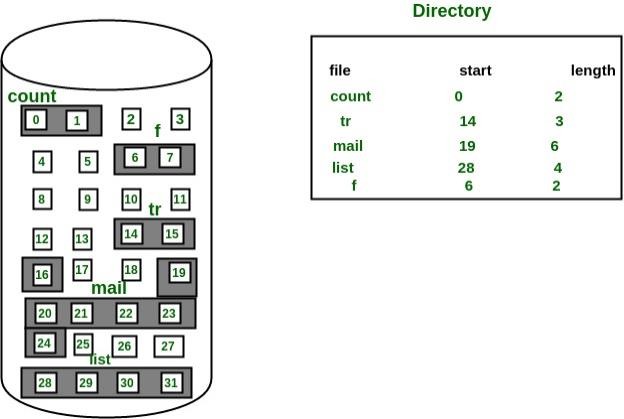
* Contiguous Allocation
* Linked Allocation
* Indexed Allocation

The main idea behind these methods is to provide:

* Efficient disk space utilization.
* Fast access to the file blocks.

1. **Contiguous Allocation:-** Each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting

location, then the blocks assigned to the file will be: *b, b+1, b+2,……b+n-1.* This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file. The directory entry for a file with contiguous allocation contains

* Address of starting block
* Length of the allocated portion.

The *file ‘mail’* in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies *19, 20, 21, 22, 23, 24* blocks.

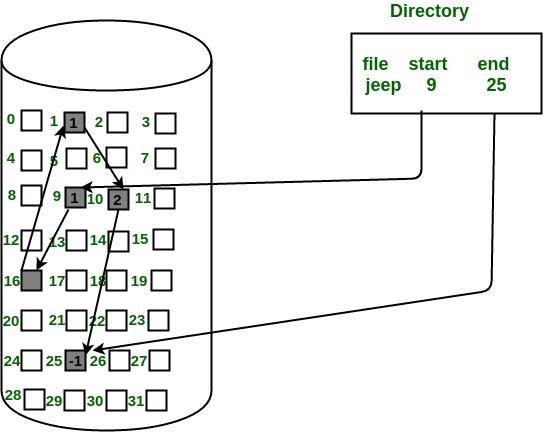
### Advantages:-

* Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
* This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

### Disadvantages:-

* This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
* Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

### Linked List Allocation:-



In this scheme, each file is a linked list of disk blocks which **need not be** contiguous. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

The file ‘jeep’ in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.

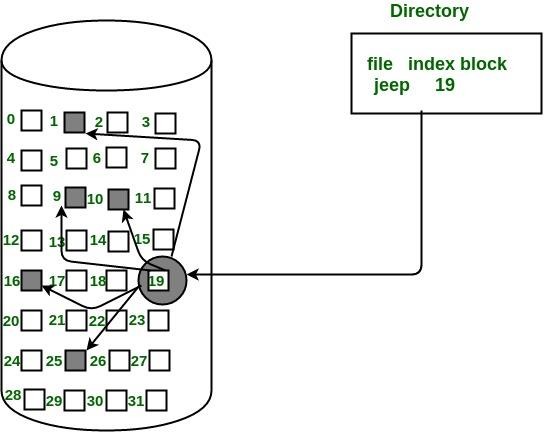
### Advantages:-

* This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
* This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

### Disadvantages:-

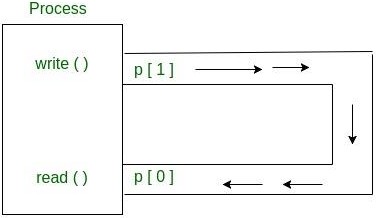
* Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
* It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access ) from the starting block of the file via block pointers.
* Pointers required in the linked allocation incur some extra overhead.

1. **Indexed Allocation:-** In this Process, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file. Each file has its own index block. The i th entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block as shown in the image:



### Explain about the Pipes along with Program ?

**Ans:-** Conceptually, a pipe is a connection between two processes, such that the standard output from one process becomes the standard input of the other process. In UNIX Operating System, Pipes are useful for communication between related processes(inter-process communication).

* Pipe is one-way communication only i.e we can use a pipe such that One process write to the pipe, and the other process reads from the pipe. It opens a pipe, which is an area of main memory that is treated as a ***“virtual file”***.
* The pipe can be used by the creating process, as well as all its child processes, for reading and writing. One process can write to this “virtual file” or pipe and another related process can read from it.
* If a process tries to read before something is written to the pipe, the process is suspended until something is written.
* The pipe system call finds the first two available positions in the process’s open file table and allocates them for the read and write ends of the pipe.

### Syntax in C language:

**int pipe(int fds[2]);**

### Parameters :

**fd[0]** will be the fd(file descriptor) for the read end of pipe.

**fd[1]** will be the fd for the write end of pipe.

**Returns :** 0 on Success.

**-1** on error.

Pipes behave **FIFO**(First in First out), Pipe behave like a **queue** data structure. Size of read and write don’t have to match here. We can write **512** bytes at a time but we can read only 1 byte at a time in a pipe.

// C program to illustrate

// pipe system call in C #include <stdio.h> #include <unistd.h> #define MSGSIZE 16

char\* msg1 = "hello, world #1"; char\* msg2 = "hello, world #2"; char\* msg3 = "hello, world #3"; int main()

{

char inbuf[MSGSIZE]; int p[2], i;

if (pipe(p) < 0)

exit(1);

/\* continued \*/

/\* write pipe \*/

write(p[1], msg1, MSGSIZE); write(p[1], msg2, MSGSIZE); write(p[1], msg3, MSGSIZE); for (i = 0; i < 3; i++) {

/\* read pipe \*/

read(p[0], inbuf, MSGSIZE); printf("% s\n", inbuf);

}

return 0;

}

**Output:**

hello, world #1

hello, world #2

hello, world #3

### Explain about the uses and types of Buffer ?

**Ans:-** The ***buffer*** is an area in the **main memory** used to store or hold the data **temporarily**. In other words, buffer temporarily stores data transmitted from one place to another, either between two devices or an application. The act of storing data temporarily in the buffer is called ***buffering***.

A buffer may be used when moving data between processes within a computer. Buffers can be implemented in a fixed memory location in hardware or by using a virtual data buffer in software, pointing at a location in the physical memory. In all cases, the data in a data buffer are stored on a physical storage medium.

Most buffers are implemented in software, which typically uses the faster RAM to store temporary data due to the much faster access time than hard disk

drives. Buffers are typically used when there is a difference between the rate of received data and the rate of processed data, for example, in a printer spooler ,video streaming.

A buffer often adjusts timing by implementing a queue or FIFO algorithm in memory, simultaneously writing data into the queue at one rate and reading it at another rate.

### Purpose of Buffering:-

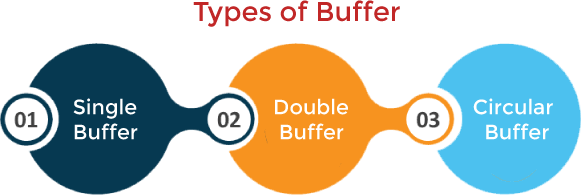
You face buffer during watching videos on YouTube or live streams. In a video stream, a buffer represents the amount of data required to be downloaded before the video can play to the viewer in real-time. A buffer in a computer environment means that a set amount of data will be stored to preload the required data before it gets used by the CPU.

There are three reasons behind buffering of data,

1. It helps in ***matching speed*** between two devices in which the data is transmitted. **Example :-** A hard disk has to store the file received from the modem. The transmission speed of a modem is slow compared to the hard disk. So bytes coming from the modem is accumulated in the buffer space, and when all the bytes of a file has arrived at the buffer, the entire data is written to the hard disk in a single operation.
2. It helps the devices with different ***sizes of data transfer*** to get adapted to each other. It helps devices to manipulate data before sending or receiving it. In computer networking, the large message is fragmented into small fragments and sent over the network. The fragments are accumulated in the buffer at the receiving end and reassembled to form a complete large message.
3. It also supports ***copy semantics***. With copy semantics, the version of data in the buffer is guaranteed to be the version of data at the time of system call, irrespective of any subsequent change to data in the buffer. Buffering increases the performance of the device. It overlaps the I/O of one job with the computation of the same job.

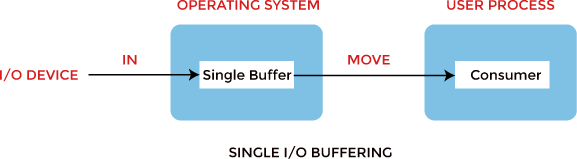
### Types of Buffering:-

There are three main types of buffering in the operating system, such as:



1. **Single Buffer :-**In Single Buffering, only one buffer is used to transfer the data between two devices. The producer produces one block of data into the buffer. After

that, the consumer consumes the buffer. Only when the buffer is empty, the processor again produces the data.



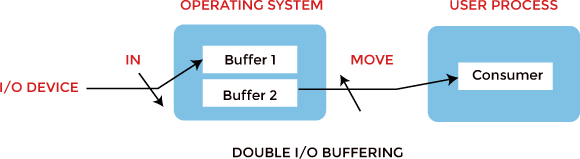
**Block oriented device:-** The following operations are performed in the block- oriented device,

* + System buffer takes the input.
  + After taking the input, the block gets transferred to the user space and then requests another block.
  + Two blocks work simultaneously. When the user processes one block of data, the next block is being read in.
  + OS can swap the processes.
  + OS can record the data of the system buffer to user processes.

**Stream oriented device:-** It performed the following operations, such as:

* + **Line** *-* at a time operation is used for scroll made terminals. The user inputs one line at a time, with a carriage return waving at the end of a line.
  + **Byte** *-* at a time operation is used on forms mode, terminals when each keystroke is significant.

1. **Double Buffer:-** In Double Buffering, two buffers are used in the place.In this buffering, the producer produces one buffer while the consumer consumes another buffer simultaneously. So, the producer not needs to wait for filling the buffer. Double buffering is also known as buffer swapping.



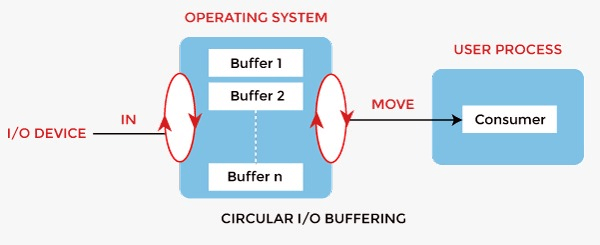
**Block oriented:-** This is how a double buffer works. There are two buffers in the system.

* + The driver or controller uses one buffer to store data while waiting for it to be taken by a higher hierarchy level.
  + Another buffer is used to store data from the lower-level module.
  + A major disadvantage of double buffering is that the complexity of the process gets increased.
  + If the process performs rapid bursts of I/O, then using double buffering may be deficient.

**Stream oriented:-** It performs these operations, such as:

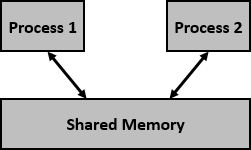
* + **Line***-* at a time I/O, the user process does not need to be suspended for input or output unless the process runs ahead of the double buffer.
  + **Byte***-* at time operations, double buffer offers no advantage over a single buffer of twice the length.

1. **Circular Buffer :-** When more than two buffers are used, the buffers' collection is called a **circular buffer**. Each buffer is being one unit in the circular buffer. The data transfer rate will increase using the circular buffer rather than the double buffering.



* + In this, the data do not directly pass from the producer to the consumer because the data would change due to overwriting of buffers before consumed.
  + The producer can only fill up to buffer x-1 while data in buffer x is waiting to be consumed.

### Write about Shared Memory ?

**Ans**:- A shared Memory is an extra piece of memory, that is attached to some address spaces for their owners to use. All of these processes share the same memory segment and have access to it.

Conditions may occur, if memory accesses are not handled properly.

In this figure, Rectangle is a shared memory attached to both address spaces and both process 1 and process2 can have access to

this shared memory as if the shared memory s part of its own address space. The Original address spaces is extended by attaching this shared memory. Shared memory is a feature supported by UNIX System V , including Linux, SunOS and Solaris.

A Shared memory segment is identified by a unique integer, the shared memory ID. The Shared memory itself is described by a structure of type shmid

\_ds in header file sys/ shm .h . To use this file, files sys/types. h and sys / ipc. h must be included. Every program must be starts with the following lines :

# include < sys / types. h >

#include <sys / ipc. h> #include < sys / shm. h> Uses of Shared memory :-

1. Ask for a shared memory with a memory key and memorize the returned shared memory ID. This is performed by system call shmget ( ).
2. Attach this shared memory to the server’s address space with system call shmat ( )
3. Initialize the shared memory , if necessary .
4. Do something and wait for all client’s completion.
5. Detach the shared memory with system call shmdt( ) 6.Remove the shared memory with system call shmctl ( )

7. For the Client part, the procedure is almost the same.

### Explain about Security Policy Mechanism and Protection ?

**Ans:-** A Security policy I a statement of what is allowed , and what is not allowed. A Security mechanism is a method for enforcing a security policy. Mechanisms can be non-technical , requiring proof of identity before changing a password.

The commonly accepted aspects of security are as follows:

* + [Identification and authentication](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009740_.html)
  + [Authorization](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009750_.html)
  + [Auditing](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009760_.html)
  + [Confidentiality](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009770_.html)
  + [Data integrity](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009780_.html)

*Security mechanisms* are technical tools and techniques that are used to implement security services. A mechanism might operate by itself, or with others, to provide a particular service. Examples of common security mechanisms are as follows:

* + [Cryptography](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009800_.html)
  + [Message digests and digital signatures](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009810_.html)
  + [Digital certificates](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009820_.html)
  + [Public Key Infrastructure (PKI)](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009900_.html)
  + [**Identification and authentication**:](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009740_.html)- Identification is the ability to identify uniquely a user of a system or an application that is running in the system. Authentication is the ability to prove that a user or application is genuinely who that person or what that application claims to be.
  + [**Authorization**](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009750_.html) :- Authorization protects critical resources in a system by limiting access only to authorized users and their applications. It prevents the unauthorized use of a resource or the use of a resource in an unauthorized manner.
  + [**Auditing**](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009760_.html) :- Auditing is the process of recording and checking events to detect whether any unexpected or unauthorized activity has taken place, or whether any attempt has been made to perform such activity.
  + [**Confidentiality**:](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009770_.html)- The confidentiality service protects sensitive information from unauthorized disclosure.
  + [**Data integrity**](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009780_.html) :- The *data integrity* service detects whether there has been unauthorized modification of data.
  + [**Cryptographic concepts**:](https://www.ibm.com/docs/en/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009790_.html)- Cryptographic protocols provide secure connections, enabling two parties to communicate with privacy and data integrity. The Transport Layer Security (TLS) protocol evolved from that of the Secure Sockets Layer (SSL).

### Explain about Android Application Architecture ? (OR) Write about Android Development Framework ?

**Ans:-** Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services.

The components Linux Kernel provides main functionality of operating system functions to smart phones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

The main components of android architecture are following:-

* Applications
* Application Framework
* Android Runtime
* Platform Libraries
* Linux Kernel

### Applications :–

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.

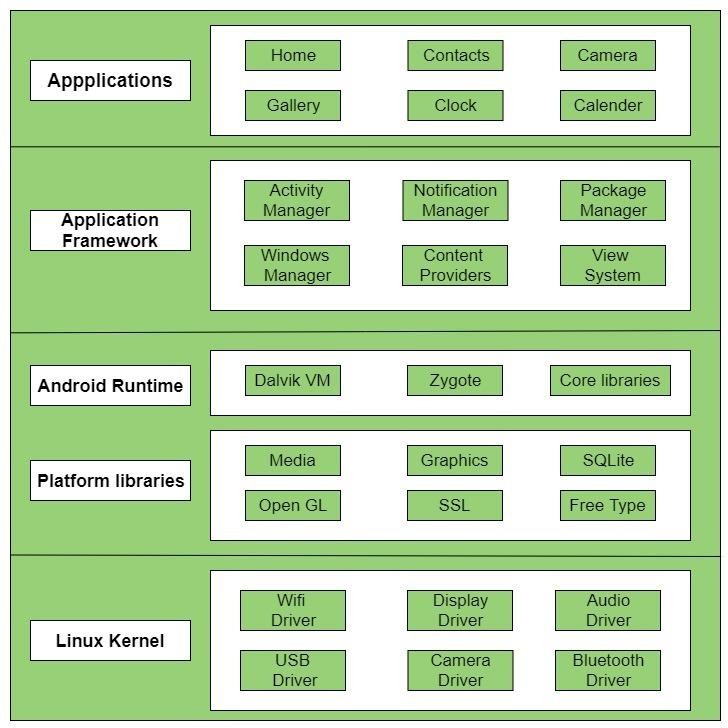
It runs within the Android run time with the help of the classes and services provided by the application framework.

### Application framework :–

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite .

Pictorial representation of android architecture with several main components and their sub components



### Application runtime : –

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

### Platform libraries : –

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

* **Media** library provides support to play and record an audio and video formats.

**Surface manager** responsible for managing access to the display subsystem.

* **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
* **SQLite** provides database support and **FreeType** provides font support.
* **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
* **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

### Linux Kernel :–

Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

* **Security:** The Linux kernel handles the security between the application and the system.
* **Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.
* **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
* **Network Stack:** It effectively handles the network communication.
* **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

### 11.) Explain Android process management ?

**Ans:-** Windows allows desktop apps to remain running whether they’re visible or not, while Apple’s iOS only allows apps to perform a few limited tasks in the background. Android sits somewhere in between — apps running in the foreground are prioritized, but apps have much more freedom to run in the background than they do on iOS.

Take a look at exactly how Android is managing the apps and processes running on your phone or tablet, demystifying what’s going on in the background.

### The Process Lifecycle Hierarchy :-

A process on Android can be in one of five different states at any given time, from most important to least important:

* **1. Foreground process**:- The app you’re using is considered the foreground process. Other processes can also be considered foreground processes — for example, if they’re interacting with the process that’s currently in the foreground. There are only a few foreground processes at any given time.
* **2. Visible process**:- A visible process isn’t in the foreground, but is still affecting what you see on your screen. For example, the foreground process may be a dialog that allows you to see an app behind it — the app visible in the background would be a visible process.
* **3. Service process**: A service process isn’t tied to any app that’s visible on your screen. However, it’s doing something in the background, such as playing music or downloading data in the background. For example, if you start playing music and switch to another app, the music-playing is in the background is being handled by a service process.
* **4. Background process**: Background processes are not currently visible to the user. They have no impact on the experience of using the phone. At any given time, many background processes are currently running. You can think of these background processes as “paused” apps. They’re kept in memory so you can quickly resume using them when you go back to them, but they aren’t using valuable CPU time or other non-memory resources.
* **5. Empty process**: An empty process doesn’t contain any app data anymore. It may be kept around for caching purposes to speed up app launches later, or the system may kill it as necessary.

### Android Automatically Manages Processes

Android does a good job of automatically managing these processes, which is why [you don’t need a task killer on Android](https://www.howtogeek.com/127388/htg-explains-why-you-shouldnt-use-a-task-killer-on-android/).

When Android needs more system resources, it will start killing the least important processes first. Android will start to kill empty and background processes to free up memory if you’re running low. If you need more memory — for example, if you’re playing a particularly demanding game on a device without much RAM, Android will then start to kill service processes, so your streaming music and file downloads may stop.

Android does this all without you needing to worry about it. Android intelligently uses your device’s RAM for caching apps and other data. Because there is no point in leaving your RAM empty..

One interesting thing you’ll notice in antivirus apps like Avast! for Android is that the antivirus app uses a notification icon. if you try to disable the notification icon, Avast! will recommend against it. By having a visible notification icon, Avast! makes itself higher-priority app, preventing Android from considering it a background app and killing it.

### Explain Android file system?

**Ans** :- File System types are : “ /boot , /system , /recovery , /data , /cache , /misc “. Also for SD Card File System Partitions are “ /sdcard , / sd-ext “.

**/boot :-**This is the boot partition of your Android device, as the name suggests. It includes the android kernel and the ram disk. The device will not boot without this partition. Wiping this partition from recovery should only be done if absolutely required and once done, the device must NOT be rebooted before installing a new one, which can be done by installing a ROM that includes a /boot partition.

**/system:-** This partition contains the entire Android OS. This includes the Android GUI and all the system applications that come pre-installed on the device. Wiping this partition will remove Android from the device without rendering it unbootable, and you will still be able to put the phone into recovery or boot loader mode to install a new ROM.

**/recovery**:- This is specially designed for backup. The recovery partition can be considered as an alternative boot partition, that lets the device boot into a recovery console for performing advanced recovery and maintenance operations on it.

**/data:-** It is called user data partition. This partition contains the user’s data like your contacts, sms, settings and all android applications that you have installed. While you are doing factory reset on your device, this partition will wipe out.

**/cache:-** This partition stores the frequently accessed data and app components. Wiping the cache doesn’t effect your personal data but simply gets rid of the existing data there, which gets automatically rebuilt as you continue using the device.

**/misc:-** This partition contains miscellaneous system settings in form of on/off switches. These settings may include Carrier or Region ID, USB configuration and certain hardware settings. This is an important partition and if it is corrupt or missing, several of the device’s features will not function normally.

**/sdcard:-** This is not a partition on the internal memory of the device but rather the SD card. In terms of usage, this is your storage space to use as you see fit, to store your media, documents, ROMs etc. on it. Wiping it is perfectly safe as long as you backup all the data you require from it, to your computer first. Though several user- installed apps save their data and settings on the SD card and wiping this partition will make you lose all that data.

**/sd-ext :-** This is not a standard Android partition, but has become popular in the custom ROM scene. It is basically an additional partition on your SD card that acts as the /data partition. It is especially useful on devices with little internal memory allotted to the /data partition. Thus, users who want to install more programs than the internal memory allows can make this partition and use it for installing their apps.

### Write about Small application development using android development frame work ?

**Ans:-** To develop small application by using android development framework need to follow some steps. Those steps are as follows :-

1. First, you learn how to create a “Hello , World !” project with android Studio and run it. Then, you create a new interface for the app that takes user input and switches to a new screen in the app to display it.

Before you start, there are 2 fundamental concepts that you need to understand about Android apps : how they provide multiple entry points, and how they adapt to different provide multi entry points, and how they adapt to different devices.

Apps Provide multiple entry points :- Android apps are built as a combination of components that can be invoked individually.

**Ex:-** An activity is a type of app component that provides a user interface(UI)

The main activity starts when the user taps your app’s icon. You can also direct the user to an activity from elsewhere, such as from a notification or even from a different app.

Other components, such as work manager, allow your app to perform background tasks without a UI .

After you build your first app, you can learn more about the other app components at Application fundamentals.

**Apps adapt to different devices:-** Android allows you to provide different resources for different devices.

**Ex:-**You can create different layouts for different screen sizes. The system determines which layer to use based on the screen size of the current device.

You can specify that your app requires certain hardware so that Google play wont allow the app to be installed on devices without them.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*