OPERATING SYSTEMS

UNIT- I

What is Operating System? History and Evolution of OS, Basic OS functions, Resource Abstraction, Types of Operating Systems– Multiprogramming Systems, Batch Systems, Time Sharing Systems; Operating Systems for Personal Computers, Workstations and Hand-held Devices, Process Control & Real time Systems.

UNIT- II

Processor and User Modes, Kernels, System Calls and System Programs, System View of the Process and Resources, Process Abstraction, Process Hierarchy, Threads, Threading Issues, Thread Libraries; Process Scheduling, Non-Preemptive and Preemptive Scheduling Algorithms.

UNIT III

Process Management: Deadlock, Deadlock Characterization, Necessary and Sufficient Conditions for Deadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery.

Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Inter-process Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

UNIT IV

Memory Management: Physical and Virtual Address Space; Memory Allocation Strategies–Fixed and -Variable Partitions, Paging, Segmentation, Virtual Memory.

UNIT V

File and I/O Management, OS security : Directory Structure, File Operations, File Allocation Methods, Device Management, Pipes, Buffer, Shared Memory, Security Policy Mechanism, Protection, Authentication and Internal Access Authorization

REFERENCE BOOKS:

1. Operating System Principles by Abraham Silberschatz, Peter Baer Galvin and Greg Gagne (7thEdition) Wiley India Edition.
2. Operating Systems: Internals and Design Principles by Stallings (Pearson)
3. Operating Systems by J. Archer Harris (Author), Jyoti Singh (Author) (TMH)

UNIT – I

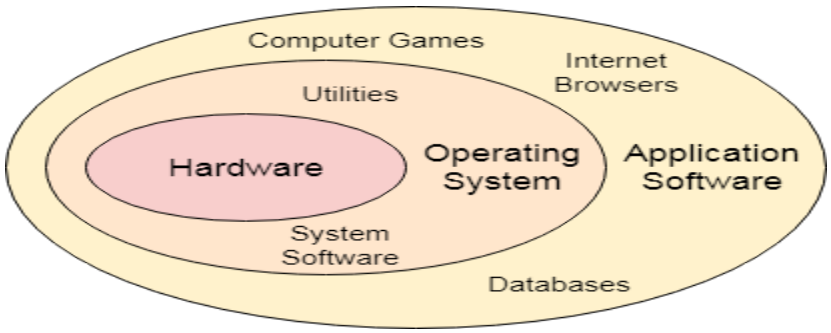
1Q) What is Operating System?

Ans:

An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.

The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

The main task an operating system carries out is the allocation of resources and services, such as the allocation of memory, devices, processors, and information. The operating system also includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.



2Q) Write the functions of Operating System?

Ans:

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 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**:
    Operating system Keeps track of time and resources used by various tasks and users, this information can be used to track resource usage for a particular user or group of users.
11. **Communication management**:
    Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems

3Q) History of Operating System

Ans:

The operating system is a system program that serves as an interface between the computing system and the end-user. Operating systems create an environment where the user can run any programs or communicate with software or applications in a comfortable and well-organized way.

Furthermore, an operating is a software program that manages and controls the execution of application programs, software resources and computer hardware. It also helps manage the software/hardware resource, such as file management, memory management, input/ output and many peripheral devices like a disk drive, printers, etc. These are the popular operating system: Linux OS, Windows OS, Mac OS, VMS, OS/400 etc.

Evolution of OS is divided in 5 phases:

Phase 0: (1940-1955):

- Computers are exotic experimental equipment.
- Program in machine language.
- Use plug boards to direct computer.
- No overlap between computation, I/O, think time, and response time.
- Programs manually loaded via card decks.

Phase 1 (1955-1970):

- User at console: one user at a time
- OS becomes a batch monitor.
- OS were written in Assembly language.
- No structured programming.
- More efficient use of hardware.
- No protection
- difficult to debug!

Phase 2 (1970-1980):

- Interactive timesharing.
- One of the first timesharing systems.

- To let multiple users interact with the system at the same time.
- Users do debugging, editing, and email online.
- More than one user executes their tasks simultaneously.

Phase 3 (1980-1990):

- Created MS- DOS.
- GUI operating systems was developed first time.
- Microsoft Windows: Win 1.0 (1985)
- Phase 4 (1990-2000):
- Networked Systems: (LAN).
- Different machines share resources, printers, File Servers, Web Servers.
- Internet service providers (service between OS and apps).

Phase 5 (2000-present):

- Mobile and computer operating systems have been developed in different ways and for different uses.
- Computer OS products are older and more familiar to larger groups of users.
- Through this time, Microsoft Windows and Apple's Mac OS have emerged as the two dominant operating system designs.
- So many types of GUI operating systems are develop in phase 5 major types are: OS system of mobiles. window 95, window 98, window XP, window crystal vista window 8, window 10.

4Q) Resource Abstraction

Ans:

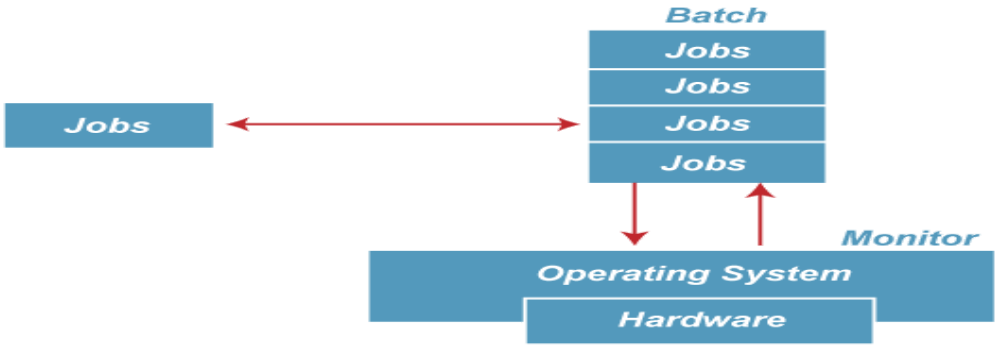
Resource abstraction is the process of "hiding the details of how the hardware operates, thereby making computer hardware relatively easy for an application programmer to use" [given by Nutt in 1997]. One way in which the operating system might implement resource abstraction is to provide a single abstract disk interface which will be the same for both the hard disk and floppy disk. Such an abstraction saves the programmer from needing to learn the details of both hardware interfaces. Instead, the programmer only needs to learn the disk abstraction provided by the operating system.

5Q) Write about types of Operating Systems

Ans:

1) Batch Operating System

- In the 1970s, Batch processing was very popular.
- In this technique, similar types of jobs were batched together and executed in time. People were used to having a single computer which was called a mainframe.
- In Batch operating system, access is given to more than one person; they submit their respective jobs to the system for the execution.
- The system put all of the jobs in a queue on the basis of first come first serve and then executes the jobs one by one. The users collect their respective output when all the jobs get executed.



- The purpose of this operating system was mainly to transfer control from one job to another as soon as the job was completed. It contained a small set of programs called the resident monitor that always resided in one part of the main memory. The remaining part is used for servicing jobs.

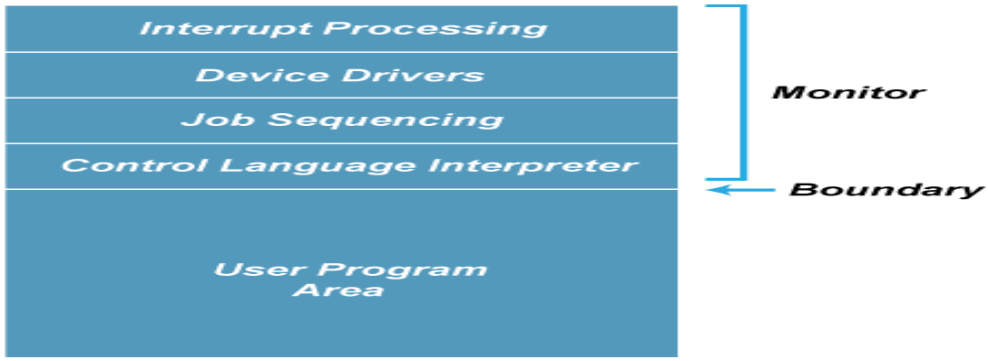


Figure: Memory Layout of the resident monitor

Advantages of Batch OS

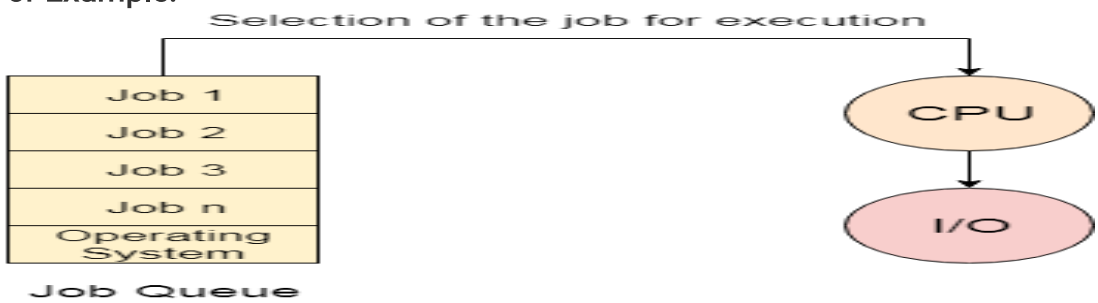
- The use of a resident monitor improves computer efficiency as it eliminates CPU time between two jobs.

Disadvantages of Batch OS

1. Starvation

   Batch processing suffers from starvation.

   For Example:



   There are five jobs J1, J2, J3, J4, and J5, present in the batch. If the execution time of J1 is very high, then the other four jobs will never be executed, or they will have to wait for a very long time. Hence the other processes get starved.

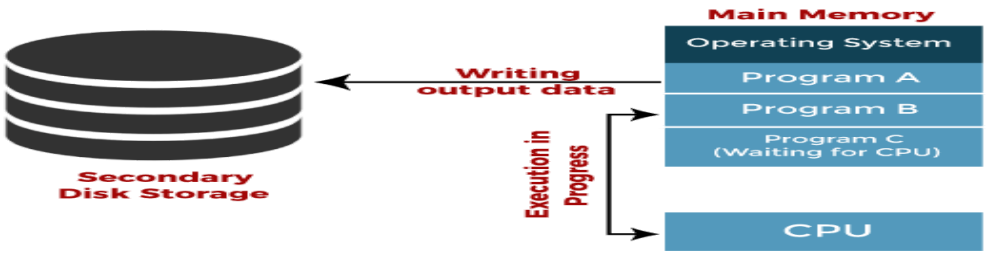
2. Not Interactive

   Batch Processing is not suitable for jobs that are dependent on the user's input. If a job requires the input of two numbers from the console, then it will never get it in the batch processing scenario since the user is not present at

the time of execution.

2) Multiprogramming Operating System

- Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.
- In a multiprogramming environment, when a process does its I/O. The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.



Jobs in multiprogramming system

Advantages of Multiprogramming OS

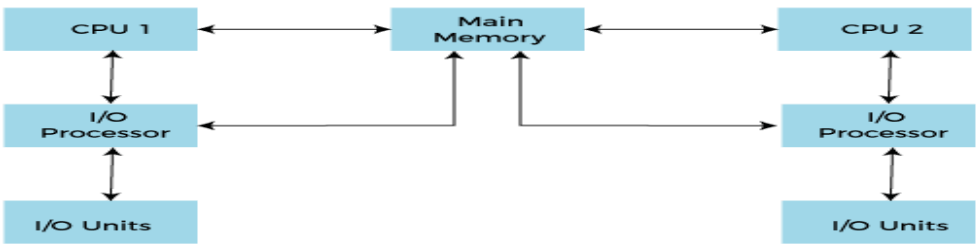
- Throughout the system, it increased as the CPU always had one program to execute.
- Response time can also be reduced.

Disadvantages of Multiprogramming OS

- Multiprogramming systems provide an environment in which various systems resources are used efficiently, but they do not provide any user interaction with the computer system.

3) Multiprocessing Operating System

- In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.



Working of Multiprocessor System

- In Multiprocessing, Parallel computing is achieved. More than one processor present in the system can execute more than one process simultaneously, which will increase the throughput of the system.

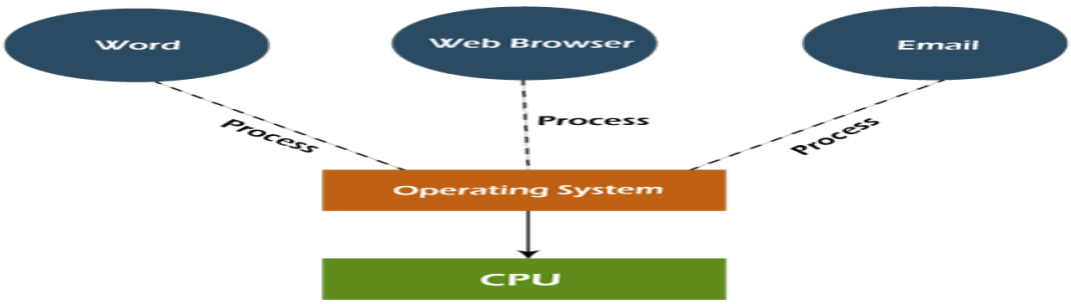
Advantages of Multiprocessing operating system:

- **Increased reliability:** Due to the multiprocessing system, processing tasks can be distributed among several processors. This increases reliability as if one processor fails, the task can be given to another processor for completion.
- **Increased throughout:** As several processors increase, more work can be done in less.

- Multiprocessing operating system is more complex and sophisticated as it takes care of multiple CPUs simultaneously.

4) Multitasking Operating System

- The multitasking operating system is a logical extension of a multiprogramming system that enables **multiple** programs simultaneously. It allows a user to perform more than one computer task at the same time.



Advantages of Multitasking operating system

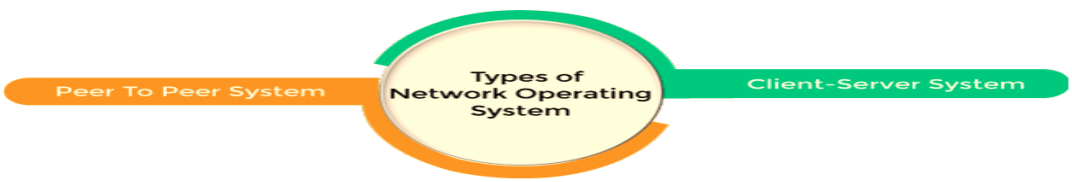
- This operating system is more suited to supporting multiple users simultaneously.
- The multitasking operating systems have well-defined memory management.

Disadvantages of Multitasking operating system

- The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.

5) Network Operating System

An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.



Advantages of Network Operating System

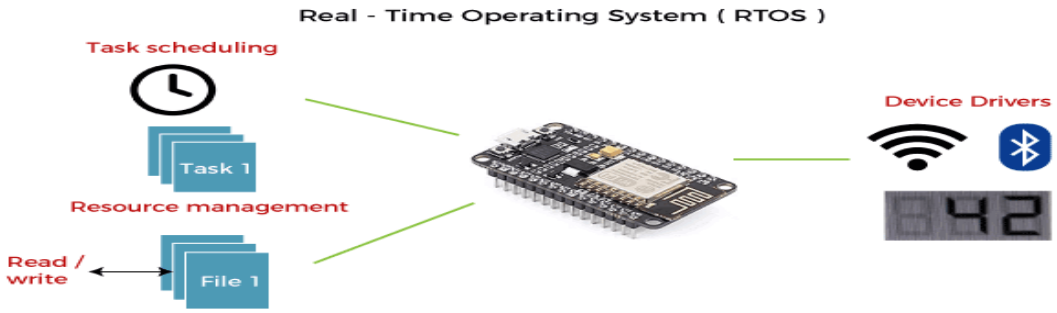
- In this type of operating system, network traffic reduces due to the division between clients and the server.
- This type of system is less expensive to set up and maintain.

Disadvantages of Network Operating System

- In this type of operating system, the failure of any node in a system affects the whole system.
- Security and performance are important issues. So trained network administrators are required for network administration.

6) Real Time Operating System

- In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed, otherwise, the huge loss will be there, or even if the result is produced, it will be completely useless.



- The Application of a Real-Time system exists in the case of military applications, if you want to drop a missile, then the missile is supposed to be dropped with a certain precision.

Advantages of Real-time operating system:

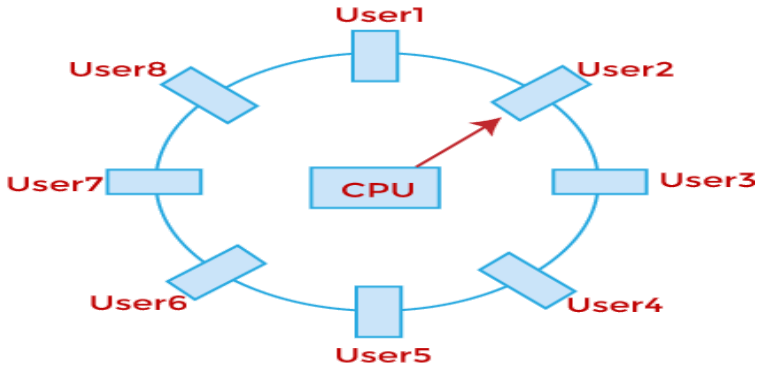
- Easy to layout, develop and execute real-time applications under the real-time operating system.
- In a Real-time operating system, the maximum utilization of devices and systems.

Disadvantages of Real-time operating system:

- Real-time operating systems are very costly to develop.
- Real-time operating systems are very complex and can consume critical CPU cycles.

7) Time-Sharing Operating System

- In the Time Sharing operating system, computer resources are allocated in a time-dependent fashion to several programs simultaneously. Thus it helps to provide a large number of user's direct access to the main computer.
- It is a logical extension of multiprogramming. In time-sharing, the CPU is switched among multiple programs given by different users on a scheduled basis.
- A time-sharing operating system allows many users to be served simultaneously, so sophisticated CPU scheduling schemes and Input/output management are required.
- Time-sharing operating systems are very difficult and expensive to build.



Timesharing in case of 8 users

Advantages of Time Sharing Operating System

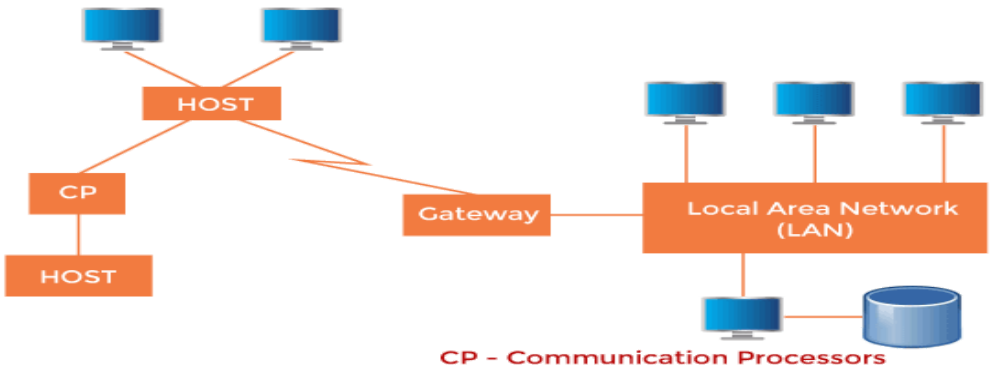
- The time-sharing operating system provides effective utilization and sharing of resources.
- This system reduces CPU idle and response time.

Disadvantages of Time Sharing Operating System

- Data transmission rates are very high in comparison to other methods.
- Security and integrity of user programs loaded in memory and data need to be maintained as many users access the system at the same time.

8) Distributed Operating System

- The Distributed Operating system is not installed on a single machine, it is divided into parts, and these parts are loaded on different machines.
- A part of the distributed Operating system is installed on each machine to make their communication possible.
- Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.



A Typical View of a Distributed System

Advantages of Distributed Operating System

- The distributed operating system provides sharing of resources.
- This type of system is fault-tolerant.

Disadvantages of Distributed Operating System

- Protocol overhead can dominate computation cost.

6Q) Operating Systems for Personal Computers

Ans:

1) Disk Operating System (DOS)

- DOS was the first widely-installed operating system for personal computers.
- It is a master control program that is automatically run when you start your personal computer (PC).
- DOS stays in the computer all the time letting you run a program and manage files.
- It is a single-user operating system from Microsoft for the PC.
- It was the first OS for the PC and is the underlying control program for Windows 3.1, 95, 98 and ME. Windows NT, 2000 and XP emulate DOS in order to support existing DOS applications.

2) UNIX

- UNIX operating systems are used in widely-sold workstation products from Sun Microsystems, Silicon Graphics, IBM, and a number of other companies.
- The UNIX environment and the client/server program model were important elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers.
- Linux, a UNIX derivative available in both “free software” and commercial versions, is increasing in popularity as an alternative to proprietary operating systems.
- UNIX is written in C. Both UNIX and C were developed by AT&T and freely distributed to government and academic institutions, causing it to be ported to a wider variety of machine families than any other operating system. As a result, UNIX became synonymous with “open systems”.
- UNIX is made up of the kernel, file system and shell (command line interface). The major shells are the Bourne shell (original), C shell and Korn shell.
- The UNIX vocabulary is exhaustive with more than 600 commands that manipulate data and text in every way conceivable. Many commands are cryptic, but just as Windows hid the DOS prompt, the Motif GUI presents a friendlier image to UNIX users. Even with its many versions, UNIX is widely used in mission critical applications for client/server and transaction

processing systems. The UNIX versions that are widely used are Sun's Solaris, Digital's UNIX, HP's HP-UX, IBM's AIX and SCO's UnixWare. A large number of IBM mainframes also run UNIX applications, because the UNIX interfaces were added to MVS and OS/390, which have obtained UNIX branding. Linux, another variant of UNIX, is also gaining enormous popularity.

3) Windows

- Windows is a personal computer operating system from Microsoft that, together with some commonly used business applications such as Microsoft Word and Excel, has become a de facto "standard" for individual users in most corporations as well as in most homes.
- Windows contains built-in networking, which allows users to share files and applications with each other if their PC's are connected to a network.
- In large enterprises, Windows clients are often connected to a network of UNIX and NetWare servers. The server versions of Windows NT and 2000 are gaining market share, providing a Windows-only solution for both the client and server.
- Windows is supported by Microsoft, the largest software company in the world, as well as the Windows industry at large, which includes tens of thousands of software developers.
- This networking support is the reason why Windows became successful in the first place. However, Windows 95, 98, ME, NT, 2000 and XP are complicated operating environments. Certain combinations of hardware and software running together can cause problems, and troubleshooting can be daunting. Each new version of Windows has interface changes that constantly confuse users and keep support people busy, and Installing Windows applications is problematic too.
- Microsoft has worked hard to make Windows 2000 and Windows XP more resilient to installation of problems and crashes in general.

4) Macintosh

- The Macintosh (often called "the Mac"), introduced in 1984 by Apple Computer, was the first widely-sold personal computer with a Graphical User Interface (GUI).
- The Mac was designed to provide users with a natural, intuitively understandable, and, in general, "user-friendly" computer interface. This includes the mouse, the use of icons or small visual images to represent objects or actions, the point-and-click and click-and-drag actions, and a

number of window operation ideas.

- Microsoft was successful in adapting user interface concepts first made popular by the Mac in its first Windows operating system.
- The primary disadvantage of the Mac is that there are fewer Mac applications on the market than for Windows. However, all the fundamental applications are available, and the Macintosh is a perfectly useful machine for almost everybody.
- Data compatibility between Windows and Mac is an issue, although it is often overblown and readily solved.
- The Macintosh has its own operating system, Mac OS which, in its latest version is called Mac OS X. Originally built on Motorola's 68000 series microprocessors, Mac versions today are powered by the PowerPC microprocessor, which was developed jointly by Apple, Motorola, and IBM.

7Q) Workstations

Ans:

- Workstation, a high performance computer system that is basically designed for a single user and has advanced graphics capabilities, large storage capacity, and a powerful central processing unit.
- A workstation is more capable than a personal computer but is less advanced than a server.
- Workstation operating systems are windows xp, windows vista, windows 7, windows 8 and similar workstation operating system is primarily designed to run applications. Those applications can be text processor, a spreadsheet application, presentation software, video or audio editors, games, etc.
- Workstation 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. **For example**, services like DHCP, DNS, FTP, Mail, Webservers, etc. Well, some of that services actually are available on workstation operating systems, but they are not optimized for them.
- As we know, almost all workstation operating systems support multiple user accounts on the same workstation, but the thing is they are not designed to be concurrent multi-user.
- Workstation OS are not designed to support multiple users at the same time, meaning they don't do it very well.

8Q) Hand Held Devices Operating Systems

Ans:

- A handheld is any portable device that can be carried and held in one's palm. A handheld can be any computing or electronic device that is compact and portable enough to be held and used on 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 Windows, but now they are light and simple to some extent

- The operating systems found on smartphones include Symbian OS, iPhone OS, RIM's Blacks Berry, Windows Mobile, Palm Web OS, Android, and Maemo, Android, webOS and Maemo are all derviced from Linux.
- The iPhone OS originated from BSD and NeXTStep, which are related to Unix.
- It combines the beauty of computer and hadn 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**
   - The Android OS is the most popular operating system today.
   - It is a mobile 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 Electorincs):**
   - Bada is a Samsung mobile 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 3D graphics, application installation and multipoint touch
3. **Black Berry OS**
   - The Black Berry operating system is a mobile operating system developed by Research in Motion (RIM).
   - This operating system was designed specifically for BlackBerry handheld devices.
   - This 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 Black Berry Enterprise server.
4. **iPhone os/IOS**
   - When the iPhone was introduced in 2007, its operating system was originally called "iPhone OS." Despite the name, the iPod Touch also ran iPhone OS.
   - In 2010, Apple introduced the iPad, which ran the same OS. When the fourth version of the mobile OS launched later that year, Apple decided to rebrand the operating system's name as "iOS," since it wasn't just the iPhone that used it anymore.
   - iOS is Apple's mobile operating system that powers the iPhone and iPod Touch. Until 2019, it was also the operating system used by the iPad

5. Symbian OS
   - Symbian OS is an operating system designed for mobile devices.
   - Symbian was the leading smartphone platform up from 2003 up until 2010. After that Google's Android OS took the lead.
   - The core Symbian OS originally provided no user interface. Instead, it was used as the underlying base for two major smartphone UI platforms: S60 and UIQ. These can be regarded as development branches, each backed by different companies. Unlike Android OS with its different cosmetic UIs, Symbian UIs ran deeper in the code and apps written for one of these platforms were not compatible with the other directly.
   - Visually, the S60 and the UIQ had nothing in common and UIQ was created with touchscreens in mind.
6. Windows Mobile OS
   - Windows Mobile was a Microsoft operating system that targeted smartphones and Pocket PCs.
   - It was first released in the Pocket PC 2000 operating system and was based on the Windows CE kernel.
   - Windows Mobile included basic applications developed with the Microsoft Windows API and options for customization and software development with no restrictions by Microsoft.

9Q) Process Control

Ans: In an operating system, we have a number of processes present in it. Each process has some information that is needed by the CPU for the execution of the process. So, we need some kind of data structure to store information about a particular process.

A **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.

There are various attributes of a PCB that helps the CPU to execute a particular process. These attributes are

1) **Process State**
   This specifies the process state i.e. new, ready, running, waiting or terminated.
2) **Process Number**
   This shows the number of the particular process.
3) **Program Counter**
   This contains the address of the next instruction that needs to be executed in the process
4) **Registers**
   This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.
5) **List of Open Files**
   These are the different files that are associated with the process
6) **CPU Scheduling Information**

   The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

7) **Memory Management Information**

   The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

8) **I/O Status Information**

   This information includes the list of I/O devices used by the process, the list of files etc.

9) **Accounting information**

   The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

10) **Location of the Process Control Block**

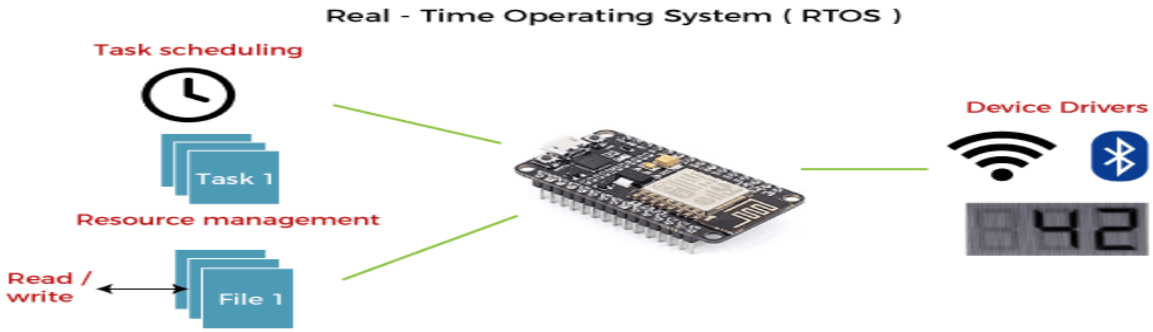
   The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

10Q) Real time Systems

Ans:

A real-time operating system (RTOS) is a special-purpose operating system used in computers that has strict time constraints for any job to be performed. It is employed mostly in those systems in which the results of the computations are used to influence a process while it is executing. Whenever an event external to the computer occurs, it is communicated to the computer with the help of some sensor used to monitor the event. The sensor produces the signal that is interpreted by the operating system as an interrupt. On receiving an interrupt, the operating system

invokes a specific process or a set of processes to serve the interrupt.



This process is completely uninterrupted unless a higher priority interrupt occurs during its execution. Therefore, there must be a strict hierarchy of priority among the interrupts. The interrupt with the highest priority must be allowed to initiate the process, while lower priority interrupts should be kept in a buffer that will be handled later. Interrupt management is important in such an operating system.

The various examples of Real-time operating systems are:

- MTS
- Lynx
- QNX
- VxWorks etc.

Types of Real-time operating system

1. **Hard Real-Time operating system:**
   In Hard RTOS, all critical tasks must be completed within the specified time duration, i.e., within the given deadline. Not meeting the deadline would result in critical failures such as damage to equipment or even loss of human life.
2. **Soft Real-Time operating system:**
   Soft RTOS accepts a few delays via the means of the Operating system. In this kind of RTOS, there may be a closing date assigned for a particular job, but a delay for a small amount of time is acceptable. So, cut off dates are treated softly via means of this kind of RTOS.
3. **Firm Real-Time operating system:**
   In Firm RTOS additionally want to observe the deadlines. However, lacking a closing date might not have a massive effect, however may want to purposely undesired effects, like a massive discount within the fine of a product.

Advantages of Real-time operating system:

- Easy to layout, develop and execute real-time applications under the real-time operating system.
- The real-time working structures are extra compact, so those structures require much less memory space.
- In a Real-time operating system, the maximum utilization of devices and systems.
- These types of systems are error-free.
- Memory allocation is best managed in these types of systems.

Disadvantages of Real-time operating system:

- Real-time operating systems have complicated layout principles and are very

costly to develop.
- Real-time operating systems are very complex and can consume critical CPU cycles.

UNIT- II

1Q) Processor and User Modes

Ans:

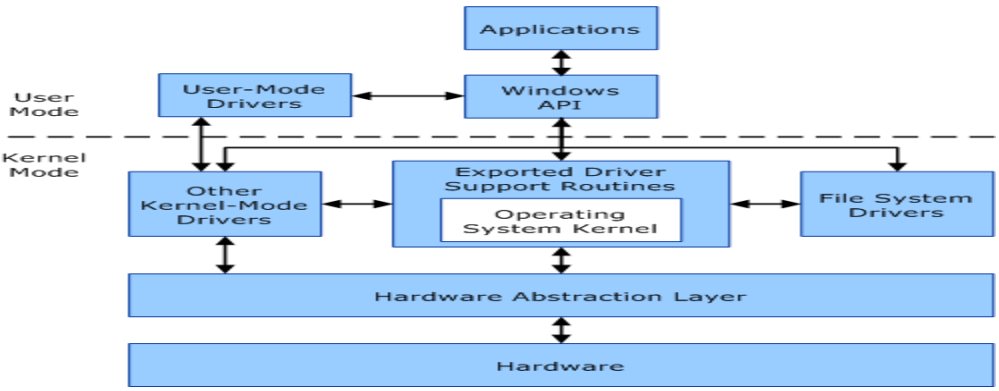
- A processor in a computer running Windows has two different modes
  4. User mode
  5. kernel mode.
- The processor switches between the two modes depending on what type of code is running on the processor.
- **Applications run in user mode**, and **core operating system components run in kernel mode**. While many drivers run in kernel mode, some drivers may run in user mode.

1. **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* 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.
- 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.

2. **Kernel mode**
   - All code that runs in kernel mode shares a single virtual address space. 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.



2Q) Write about System Calls and System Programs?

Ans:

- A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running.
- A system call is a method of interacting with the operating system via programs.
- A system call is a request from computer software to an operating system's kernel.
- The **Application Program Interface (API)** connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services. The kernel system can only be accessed using system calls.
- System calls are required for any programs that use resources.

Types of System Calls

There are commonly five types of system calls. These are as follows:

1. Process Control
2. File Management
3. Device Management
4. Information Maintenance
5. Communication

Process Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

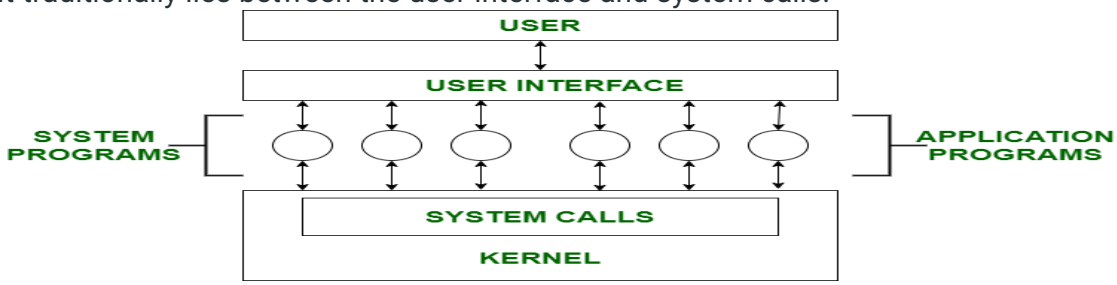
Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

System Programs

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. Some of the System Programs are simply user interfaces, others are complex. It traditionally lies between the user interface and system calls.



3Q) Viewpoints of Operating System

Ans: The operating system may be observed from the viewpoint of the user or the system. It is known as the user view and the system view. There are mainly two types of views of the operating system. These are as follows:

1. **User View**
2. **System View**

1. User View

The user view depends on the system interface that is used by the users. Some systems are designed for a single user to monopolize the resources to maximize the user's task. In these cases, the OS is designed primarily for ease of use, with

little emphasis on quality and none on resource utilization.

The user viewpoint focuses on how the user interacts with the operating system through the usage of various application programs. In contrast, the system viewpoint focuses on how the hardware interacts with the operating system to complete various tasks.

1) **Single User View Point**
   Most computer users use a monitor, keyboard, mouse, printer, and other accessories to operate their computer system. In some cases, the system is designed to maximize the output of a single user. As a result, more attention is laid on accessibility, and resource allocation is less important. These systems are much more designed for a single user experience and meet the needs of a single user, where the performance is not given focus as the multiple user systems.
2) **Multiple User View Point**
   Another example of user views in which the importance of user experience and performance is given is when there is one mainframe computer and many users on their computers trying to interact with their kernels over the mainframe to each other. In such circumstances, memory allocation by the CPU must be done effectively to give a good user experience. The client-server architecture is another good example where many clients may interact through a remote server, and the same constraints of effective use of server resources may arise.
3) **Handled User View Point**
   Moreover, the touchscreen era has given you the best handheld technology ever. Smartphones interact via wireless devices to perform numerous operations, but they're not as efficient as a computer interface, limiting their usefulness. However, their operating system is a great example of creating a device focused on the user's point of view.
4) **Embedded System User View Point**
   Some systems, like embedded systems that lack a user point of view. The remote control used to turn **on** or **off** the tv is all part of an embedded system in which the electronic device communicates with another program where the user viewpoint is limited and allows the user to engage with the application.

2. **System View**
   - The OS may also be viewed as just a resource allocator.
   - A computer system comprises various sources, such as hardware and software, which must be managed effectively.
   - The operating system manages the resources, decides between competing demands, controls the program execution, etc.
   - The operating system is responsible for managing hardware resources and allocating them to programs and users to ensure maximum performance. The hardware and the operating system interact for a variety of reasons, including:
     1. **Resource Allocation**
        The hardware contains several resources like registers, caches, RAM, ROM, CPUs, I/O interaction, etc. These are all resources that the operating system needs when an application program demands them. Only the operating system can allocate resources, and it has used several tactics and strategies to maximize its processing and memory space. The

operating system uses a variety of strategies to get the most out of the hardware resources, including paging, virtual memory, caching, and so on. These are very important in the case of various user viewpoints because inefficient resource allocation may affect the user viewpoint, causing the user system to lag or hang, reducing the user experience.

2. **Control Program**
   The control program controls how input and output devices (hardware) interact with the operating system. The user may request an action that can only be done with I/O devices; in this case, the operating system must also have proper communication, control, detect, and handle such devices.

4Q) What is Process Abstraction? Write about Process Hierarchy?

Ans:

In process abstraction, **details of the threads of execution are not visible to the consumer of the process**. An example of process abstraction is the concurrency scheduler in a database system. A database system can handle many concurrent queries.

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.

5Q) What is Thread? Explain types of Threads?

Ans:

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.

Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

User-level thread

The operating system does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level

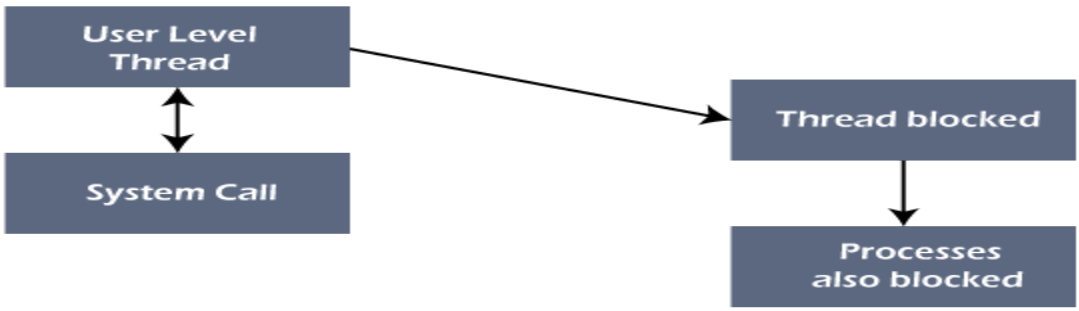
thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes

Advantages of User-level threads

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

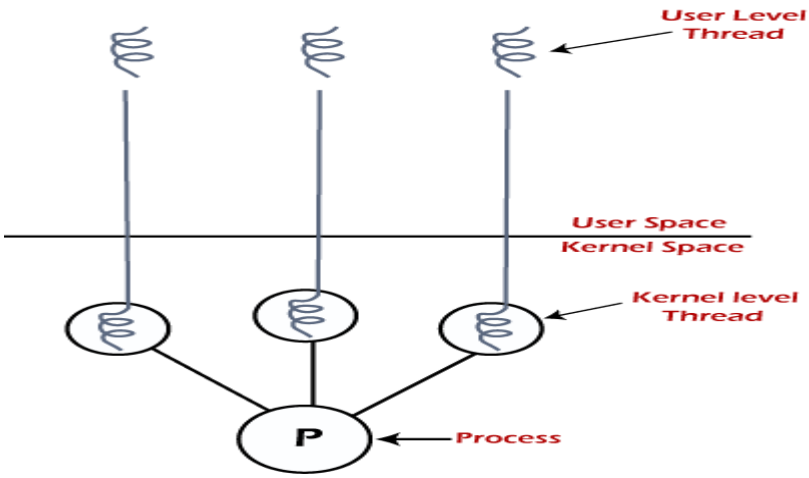
Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.



Kernel level thread

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.



Advantages of Kernel-level threads

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

Disadvantages of Kernel-level threads

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

Components of Threads

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

Benefits of Threads

- **Enhanced throughput of the system:**
  When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
- **Effective Utilization of Multiprocessor system:**
  When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:**
  The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
- **Responsiveness:**
  When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
- **Communication:**
  Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
- **Resource sharing:**
  Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between

threads. There is a stack and register for each thread.

6Q) Explain about Thread issues

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

Threading Issues in OS

1. System Calls
2. Thread Cancellation
3. Signal Handling
4. Thread Pool
5. Thread Specific Data

1. The fork() and exec() system calls

- The fork() is used to create a duplicate process. The meaning of the fork() and exec() system calls change in a multithreaded program.
- If one thread in a program which calls fork(), does the new process duplicate all threads, or is the new process single-threaded? If we take, some UNIX systems have chosen to have two versions of fork(), one that duplicates all threads and another that duplicates only the thread that invoked the fork() system call.
- If a thread calls the exec() system call, the program specified in the parameter to exec() will replace the entire process which includes all threads.

2. Signal Handling

Generally, signal is used in UNIX systems to notify a process that a particular event has occurred. A signal received either synchronously or asynchronously, based on the source of and the reason for the event being signaled. All signals, whether synchronous or asynchronous, follow the same pattern as given below −

- A signal is generated by the occurrence of a particular event.
- The signal is delivered to a process.
- Once delivered, the signal must be handled.

3. Thread Cancellation

Thread cancellation is the task of terminating a thread before it has completed.

A target thread is a thread that is to be cancelled, cancellation of target thread may occur in two different scenarios −

1) **Asynchronous cancellation** − One thread immediately terminates the target thread.
2) **Deferred cancellation** − The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an ordinary fashion.

4. Thread polls

Multithreading in a web server, whenever the server receives a request it creates a separate thread to service the request. Some of the problems that arise in creating a thread are as follows −

- The amount of time required to create the thread prior to serving the request together with the fact that this thread will be discarded once it has completed its work.
- If all concurrent requests are allowed to be serviced in a new thread, there is no bound on the number of threads concurrently active in the system.
- Unlimited thread could exhaust system resources like CPU time or memory.

A thread pool is to create a number of threads at process start-up and place them into a pool, where they sit and wait for work.

5. **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.**

7Q) Write about Thread Libraries

Ans:

- Thread libraries provide programmers with an API for creating and managing threads.
- Thread libraries may be implemented either in user space or in kernel space. The former involves API functions implemented solely within user space, with no kernel support. The latter involves system calls, and requires a kernel with thread library support. There are three main thread libraries in use today:
  1. **POSIX Pthreads** - may be provided as either a user or kernel library, as an extension to the POSIX standard.
  2. **Win32 threads** - provided as a kernel-level library on Windows systems.
  3. **Java threads**
     Since Java generally runs on a Java Virtual Machine, the implementation of threads is based upon whatever OS and hardware the JVM is running on, i.e. either Pthreads or Win32 threads depending on the system.

  The following sections will demonstrate the use of threads in all three systems for calculating the sum of integers from 0 to N in a separate thread, and storing the result in a variable "sum".

  1) **Pthreads**
     - The POSIX standard (IEEE 1003.1c) defines the specification for pThreads, not the implementation.
     - pThreads are available on Solaris, Linux, Mac OSX, Tru64, and via public domain shareware for Windows.
     - Global variables are shared amongst all threads.
     - One thread can wait for the others to rejoin before continuing.
     - pThreads begin execution in a specified function, in this example the runner( ) function:

```
#define NUM_THREADS 10

/* an array of threads to be joined upon */
pthread_t workers[NUM_THREADS];

for (int i = 0; i < NUM_THREADS; i++)
   pthread_join(workers[i], NULL);
```

  2) **Java Threads**
     - ALL Java programs use Threads - even "common" single-threaded ones.
     - The creation of new Threads requires Objects that implement the Runnable Interface, which means they contain a method "public void run()" . Any descendant of the Thread class will naturally contain such a method.
     - Creating a Thread Object does not start the thread running - To do

that the program must call the Thread's "start( )" method. Start( ) allocates and initializes memory for the Thread, and then calls the run( ) method.

- Because Java does not support global variables, Threads must be passed a reference to a shared Object in order to share data, in this example the "Sum" Object.
- Note that the JVM runs on top of a native OS, and that the JVM specification does not specify what model to use for mapping Java threads to kernel threads. This decision is JVM implementation dependant, and may be one-to-one, many-to-many, or many to one.

3) **Windows Threads**

Similar to pThreads. Examine the code example to see the differences, which are mostly syntactic & nomenclature:

8Q) Process Scheduling

Ans:

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.

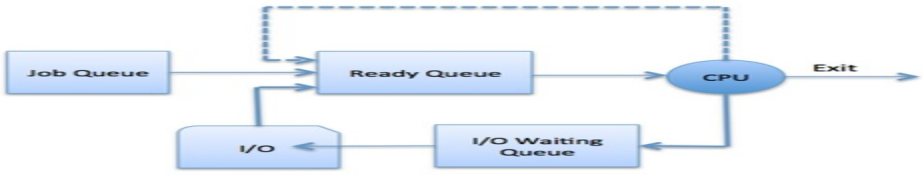
Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Process Scheduling Queues

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below –

S.No.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types –

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

1) Long Term Scheduler

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time -sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

2) Short Term Scheduler

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

3) Medium Term Scheduler

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term

scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

9Q) CPU SCHEDULING

Ans:

Scheduling of processes/work is done to finish the work on time. **CPU Scheduling** is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.

Whenever the CPU becomes idle, the operating system must select one of the processes in the line ready for launch. The selection process is done by a temporary (CPU) scheduler. The Scheduler selects between memory processes ready to launch and assigns the CPU to one of them.

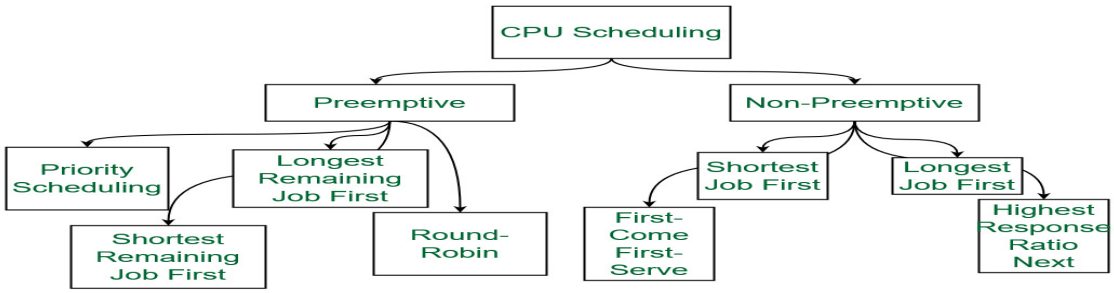
CPU scheduling decisions may take place under the following four circumstances:

1) When a process switches from the running state to the waiting state
2) When a process switches from the running state to the ready state
3) When a process switches from the waiting state to the ready state
4) When a process terminates

When scheduling takes place only under circumstances 1 and 4, we say the scheduling scheme is non-preemptive, otherwise the scheduling scheme is preemptive

There are mainly two types of scheduling methods:

- Preemptive Scheduling:
  Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
- Non-Preemptive Scheduling:
  Non-Preemptive scheduling is used when a process terminates, or when a process switches from running state to waiting state.



Different types of CPU Scheduling Algorithms

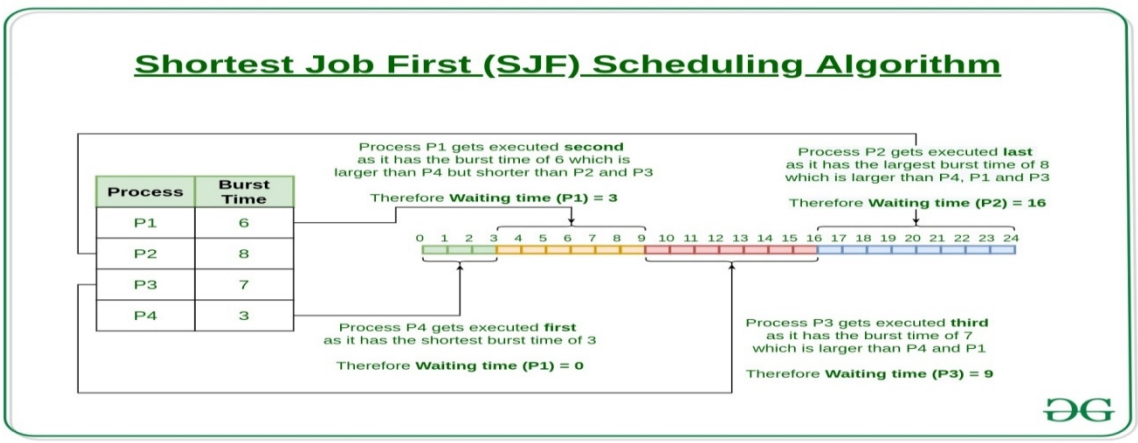
10Q) CPU scheduling algorithms in operating systems

Ans:

1. **First Come First Serve:**
   FCFS is considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.
   Characteristics of FCFS:
   - FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
   - Tasks are always executed on a First-come, First-serve concept.
   - FCFS is easy to implement and use.
   - This algorithm is not much efficient in performance, and the wait time is quite high.

   Advantages of FCFS:
   - Easy to implement
   - First come, first serve method

   Disadvantages of FCFS:
   - FCFS suffers from **Convoy effect**.
   - The average waiting time is much higher than the other algorithms.
   - FCFS is very simple and easy to implement and hence not much efficient.
2. **Shortest Job First(SJF):**
   Shortest job first (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.



   Characteristics of SJF:
   - Shortest Job first has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
   - It is associated with each task as a unit of time to complete.
   - It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

   Advantages of Shortest Job first:
   - As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
   - SJF is generally used for long term scheduling

   Disadvantages of SJF:
   - One of the demerit SJF has is starvation.

- Many times it becomes complicated to predict the length of the upcoming CPU request

3. **Longest Job First(LJF):**
   Longest Job First(LJF) scheduling process is just opposite of shortest job first (SJF), as the name suggests this algorithm is based upon the fact that the process with the largest burst time is processed first. Longest Job First is non-preemptive in nature.
   Characteristics of LJF:
   - Among all the processes waiting in a waiting queue, CPU is always assigned to the process having largest burst time.
   - If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first.
   - LJF CPU Scheduling can be of both preemptive and non-preemptive types.

   Advantages of LJF:
   - No other task can schedule until the longest job or process executes completely.
   - All the jobs or processes finish at the same time approximately.

   Disadvantages of LJF:
   - Generally, the LJF algorithm gives a very high average waiting time and average turn-around time for a given set of processes.
   - This may lead to convoy effect.

4. **Priority Scheduling:**
   Preemptive Priority CPU Scheduling Algorithm is a pre-emptive method of CPU scheduling algorithm that works **based on the priority** of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first. In the case of any conflict, that is, where there are more than one processor with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.
   Characteristics of Priority Scheduling:
   1) Schedules tasks based on priority.
   2) When the higher priority work arrives while a task with less priority is executed, the higher priority work takes the place of the less priority one
   3) The latter is suspended until the execution is complete.
   4) Lower is the number assigned, higher is the priority level of a process.

   Advantages of Priority Scheduling:
   1) The average waiting time is less than FCFS
   2) Less complex

   Disadvantages of Priority Scheduling:
   1) One of the most common demerits of the Preemptive priority CPU scheduling algorithm is the Starvation Problem. This is the problem in which a process has to wait for a longer amount of time to get scheduled into the CPU. This condition is called the starvation problem.

5. **Round robin:**
   Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.
   Characteristics of Round robin:

1) It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
2) One of the most widely used methods in CPU scheduling as a core.
3) It is considered preemptive as the processes are given to the CPU for a very limited time.

Advantages of Round robin:

1) Round robin seems to be fair as every process gets an equal share of CPU.
2) The newly created process is added to the end of the ready queue.

6. **Shortest Remaining Time First:**

Shortest remaining time first is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to the job closest to completion. In SRTF the process with the smallest amount of time remaining until completion is selected to execute.

Characteristics of Shortest remaining time first:

1) SRTF algorithm makes the processing of the jobs faster than SJF algorithm, given it's overhead charges are not counted.
2) The context switch is done a lot more times in SRTF than in SJF and consumes the CPU's valuable time for processing. This adds up to its processing time and diminishes its advantage of fast processing.

Advantages of SRTF:

1) In SRTF the short processes are handled very fast.
2) The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.

Disadvantages of SRTF:

1. Like the shortest job first, it also has the potential for process starvation.
2. Long processes may be held off indefinitely if short processes are continually added.

7. **Longest Remaining Time First:**

The longest remaining time first is a preemptive version of the longest job first scheduling algorithm. This scheduling algorithm is used by the operating system to program incoming processes for use in a systematic way. This algorithm schedules those processes first which have the longest processing time remaining for completion.

Characteristics of longest remaining time first:

- Among all the processes waiting in a waiting queue, the CPU is always assigned to the process having the largest burst time.
- If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first.
- LJF CPU Scheduling can be of both preemptive and non-preemptive types.

Advantages of LRTF:

- No other process can execute until the longest task executes completely.
- All the jobs or processes finish at the same time approximately.

Disadvantages of LRTF:

- This algorithm gives a very high average waiting time and average turn-around time for a given set of processes.
- This may lead to a convoy effect.

8. **Highest Response Ratio Next:**
   Highest Response Ratio Next is a non-preemptive CPU Scheduling algorithm and it is considered as one of the most optimal scheduling algorithms. The name itself states that we need to find the response ratio of all available processes and select the one with the highest Response Ratio. A process once selected will run till completion.

   Characteristics of Highest Response Ratio Next:
   - The **criteria** for HRRN is **Response Ratio,** and the **mode** is **Non-Preemptive.**
   - HRRN is considered as the modification of Shortest Job First to reduce the problem of starvation.
   - In comparison with SJF, during the HRRN scheduling algorithm, the CPU is allotted to the next process which has the **highest response ratio** and not to the process having less burst time.

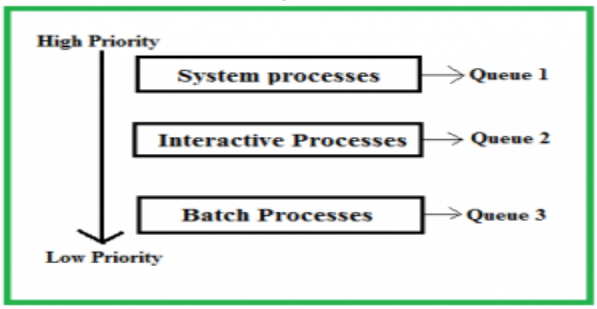
   Response Ratio = (W + S)/S

   *Here, **W** is the waiting time of the process so far and **S** is the Burst time of the process.*

   Advantages of HRRN:
   - HRRN Scheduling algorithm generally gives better performance than the shortest job first Scheduling.
   - There is a reduction in waiting time for longer jobs and also it encourages shorter jobs.

   Disadvantages of HRRN:
   - The implementation of HRRN scheduling is not possible as it is not possible to know the burst time of every job in advance.
   - In this scheduling, there may occur an overload on the CPU.

9. **Multiple Queue Scheduling:**
   Processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and a **background (batch)** process. These two classes have different scheduling needs. For this kind of situation **Multilevel Queue Scheduling** is used.



   The description of the processes in the above diagram is as follows:
   - **System Processes:** The CPU itself has its process to run, generally termed as System Process.
   - **Interactive Processes:** An Interactive Process is a type of process in which there should be the same type of interaction.
   - **Batch Processes:** Batch processing is generally a technique in the Operating system that collects the programs and data together in the form of a **batch** before the **processing** starts.

Advantages of multilevel queue scheduling:

- The main merit of the multilevel queue is that it has a low scheduling overhead.

Disadvantages of multilevel queue scheduling:

- Starvation problem
- It is inflexible in nature

10. **Multilevel Feedback Queue Scheduling::**

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like **Multilevel Queue Scheduling** but in this process can move between the queues. And thus, much more efficient than multilevel queue scheduling

Characteristics of Multilevel Feedback Queue Scheduling:

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system, and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand disadvantage of being inflexible.

Advantages of Multilevel feedback queue scheduling:

- It is more flexible
- It allows different processes to move between different queues

Disadvantages of Multilevel feedback queue scheduling:

- It also produces CPU overheads
- It is the most complex algorithm.

11Q) Non-Preemptive and Preemptive Scheduling Algorithms

Ans: A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16

Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8

Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0

P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

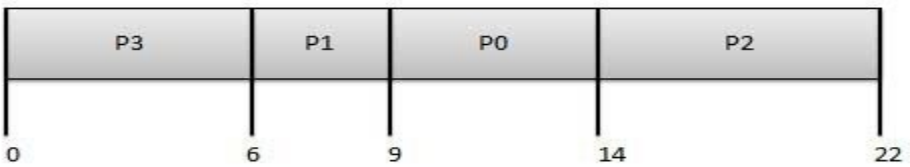
Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

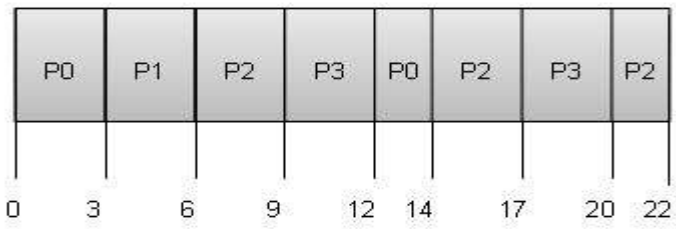
Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs

in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

UNIT III

1Q) What is Dead Lock?

Ans: Every process needs some resources to complete its execution. However, the resource is granted in a sequential order.

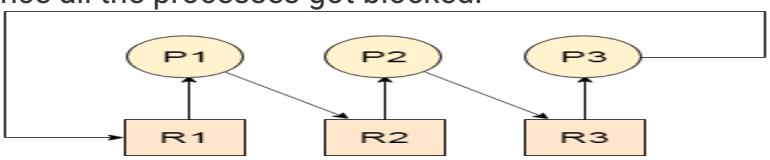
1. The process requests for some resource.
2. OS grant the resource if it is available otherwise let the process waits.
3. The process uses it and release on the completion.

A Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the process gets executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released.

Let us assume that there are three processes P1, P2 and P3. There are three different resources R1, R2 and R3. R1 is assigned to P1, R2 is assigned to P2 and R3 is assigned to P3.

After some time, P1 demands for R1 which is being used by P2. P1 halts its execution since it can't complete without R2. P2 also demands for R3 which is being used by P3. P2 also stops its execution because it can't continue without R3. P3 also demands for R1 which is being used by P1 therefore P3 also stops its execution.

In this scenario, a cycle is being formed among the three processes. None of the process is progressing and they are all waiting. The computer becomes unresponsive since all the processes got blocked.



Dead Lock Characterization:

Deadlock situation can arise if the following four conditions hold simultaneously in a system:

1) Resources are used in mutual exclusion.
2) Resources are acquired piecemeal
3) Resources are not preempted
4) Resources are not spontaneously given up by a process until it has satisfied all its outstanding requests for resources

2Q) Necessary and Sufficient Conditions for Deadlock

Ans:

1. **Mutual Exclusion**
   A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.
2. **Hold and Wait**
   A process waits for some resources while holding another resource at the same time.
3. **No preemption**
   The process which once scheduled will be executed till the completion. No other

process can be scheduled by the scheduler meanwhile.

4. **Circular Wait**
   All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

3Q) Strategies for handling Deadlock

Ans:

1) Deadlock Ignorance

Deadlock Ignorance is the most widely used approach among all the mechanism. This is being used by many operating systems mainly for end user uses. In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock. This approach is best suitable for a single end user system where User uses the system only for browsing and all other normal stuff.

There is always a tradeoff between Correctness and performance. The operating systems like Windows and Linux mainly focus upon performance. However, the performance of the system decreases if it uses deadlock handling mechanism all the time if deadlock happens 1 out of 100 times then it is completely unnecessary to use the deadlock handling mechanism all the time.

In these types of systems, the user has to simply restart the computer in the case of deadlock. Windows and Linux are mainly using this approach.

2) Deadlock prevention

Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously. If it is possible to violate one of the four conditions at any time then the deadlock can never occur in the system.

The idea behind the approach is very simple that we have to fail one of the four conditions but there can be a big argument on its physical implementation in the system.

3) Deadlock avoidance

In deadlock avoidance, the operating system checks whether the system is in safe state or in unsafe state at every step which the operating system performs. The process continues until the system is in safe state. Once the system moves to unsafe state, the OS has to backtrack one step.

In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

4) Deadlock detection and recovery

This approach let the processes fall in deadlock and then periodically check whether deadlock occur in the system or not. If it occurs then it applies some of the recovery methods to the system to get rid of deadlock.

4Q) Concurrent and Dependent Processes

Ans:

- **Concurrent processing** is a computing model in which multiple processors execute instructions simultaneously for better performance.
- Concurrent means, which occurs when something else happens. The tasks are broken into sub-types, which are then assigned to different processors to perform simultaneously, sequentially instead, as they would have to be

performed by one processor.

- Concurrent processing is sometimes synonymous with parallel processing.
- The term real and virtual concurrency in concurrent processing:

1. **Multiprogramming Environment :**
   In multiprogramming environment, there are multiple tasks shared by one processor. while a virtual concert can be achieved by the operating system, if the processor is allocated for each individual task, so that the virtual concept is visible if each task has a dedicated processor. The multilayer environment shown in figure.

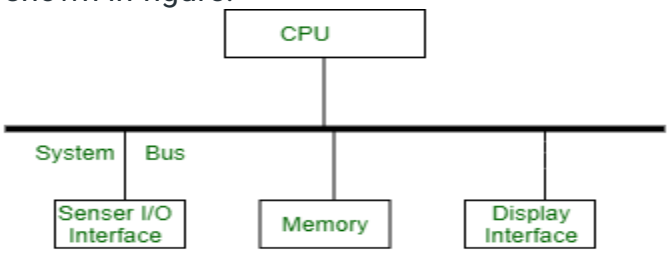


Figure - Multiprogramming (Single CPU) Environment

2. **Multiprocessing Environment :**
   In multiprocessing environment two or more processors are used with shared memory. Only one virtual address space is used, which is common for all processors. All tasks reside in shared memory. In this environment, concurrency is supported in the form of concurrently executing processors. The tasks executed on different processors are performed with each other through shared memory. The multiprocessing environment is shown in figure.

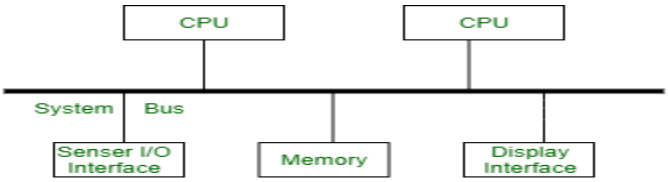


Figure - Multiprocessing Environment

3. **Distributed Processing Environment :**
   In a distributed processing environment, two or more computers are connected to each other by a communication network or high speed bus. There is no shared memory between the processors and each computer has its own local memory. Hence a distributed application consisting of concurrent tasks, which are distributed over network communication via messages. The distributed processing environment is shown in figure.

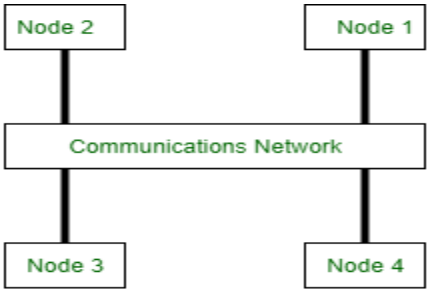


Figure - Distributed processing Environment

5Q) Critical Section

Ans:

Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.

The critical section cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.

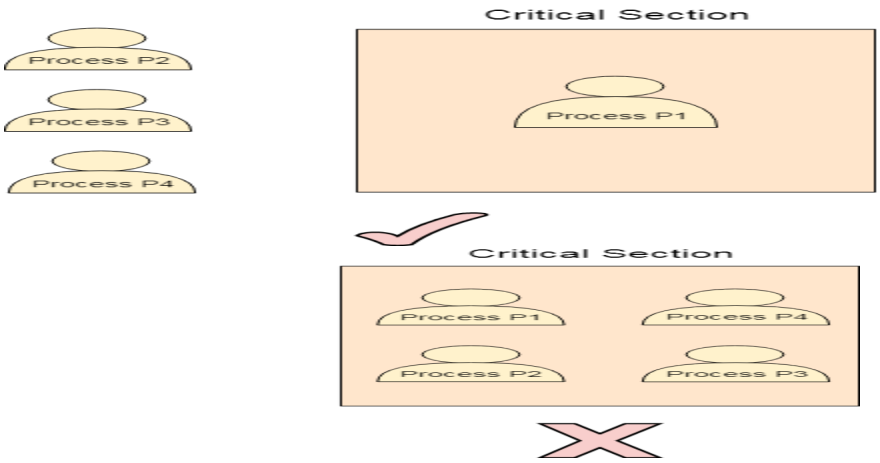
The critical section problem is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

Requirements of Synchronization mechanisms

Primary

1. **Mutual Exclusion**
   Our solution must provide mutual exclusion. By Mutual Exclusion, we mean that if one process is executing inside critical section then the other process must not enter in the critical section.



2. **Progress**
   Progress means that if one process doesn't need to execute into critical section then it should not stop other processes to get into the critical section.

Secondary

1. **Bounded Waiting**
   We should be able to predict the waiting time for every process to get into the critical section. The process must not be endlessly waiting for getting into the critical section.
2. **Architectural Neutrality**
   Our mechanism must be architectural natural. It means that if our solution is working fine on one architecture then it should also run on the other ones as well.

6Q) Semaphores

Ans:

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.

There are two kinds of semaphores:

1) **Binary Semaphores**

- Binary Semaphores can only have one of two values: 0 or 1. Because of their capacity to ensure mutual exclusion, they are also known as mutex locks.
- A single binary semaphore is shared between multiple processes.
- When the semaphore is set to 1, it means some process is working on its critical section, and other processes need to wait, and if the semaphore is set to 0, that means any process can enter the critical section.
- Hence, whenever the binary semaphore is set to 0, any process can then enter its critical section by setting the binary semaphore to 1. When it has completed its critical section, it can reset the binary semaphore to 0, enabling another process to enter it.

2) **Counting Semaphores**

- Counting Semaphores can have any value and are not limited to a certain area. They can be used to restrict access to a resource that has a concurrent access limit.
- Initially, the counting semaphores are set to the maximum amount of processes that can access the resource at a time. Hence, the counting semaphore indicates that a process can access the resource if it has a value greater than 0. If it is set to 0, no other process can access the resource. Hence,
- When a process wants to use that resource, it first checks to see if the value of the counting semaphore is more than zero.
- If yes, the process can then proceed to access the resource, which involves reducing the value of the counting semaphore by one.

- When the process completes its critical section code, it can increase the value of the counting semaphore, making way for some other process to access it.

7Q) Methods for Inter-process Communication;

Ans: Inter-process communication (IPC) is set of interfaces, which is usually programmed in order for the programs to communicate between series of processes. This allows running programs concurrently in an Operating System. These are the methods in IPC:

1. **Pipes (Same Process) –**
   This allows flow of data in one direction only. Analogous to simplex systems (Keyboard). Data from the output is usually buffered until input process receives it which must have a common origin.
2. **Names Pipes (Different Processes)**
   This is a pipe with a specific name it can be used in processes that don't have a shared common process origin. E.g. is FIFO where the details written to a pipe is first named.
3. **Message Queuing**
   This allows messages to be passed between processes using either a single queue or several message queue. This is managed by system kernel these messages are coordinated using an API.
4. **Semaphores –**
   This is used in solving problems associated with synchronization and to avoid race condition. These are integer values which are greater than or equal to 0.
5. **Shared memory**
   This allows the interchange of data through a defined area of memory. Semaphore values have to be obtained before data can get access to shared memory.
6. **Sockets –**
   This method is mostly used to communicate over a network between a client and a server. It allows for a standard connection which is computer and OS independent.

8Q) Process Synchronization

Ans:

It is the task phenomenon of coordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.

- It is a procedure that is involved in order to preserve the appropriate order of execution of cooperative processes.
- In order to synchronize the processes, there are various synchronization mechanisms.
- Process Synchronization is mainly needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or any data at the same time.

9Q) Classical Process Synchronization Problems

Ans:

The classical problems of synchronization are as follows:

1) Bound-Buffer problem
2) Sleeping barber problem
3) Dining Philosophers problem

4) Readers and writers problem

Bound-Buffer problem

- Also known as the **Producer-Consumer problem**.
- This problem is generalised in terms of the **Producer Consumer problem**, where a **finite** buffer pool is used to exchange messages between producer and consumer processes.
- 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.
- In this Producers mainly produces a product and consumers consume the product, but both can use of one of the containers each time.
- The main complexity of this problem is that we must have to maintain the count for both empty and full containers that are available.

Sleeping Barber Problem

- This problem is based on a hypothetical barbershop with one barber.
- When there are no customers the barber sleeps in his chair. If any customer enters he will wake up the barber and sit in the customer chair. If there are no chairs empty they wait in the waiting queue.

Dining Philosopher's problem

- The dining philosopher's problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.
- There are five philosophers sitting around a table, in which there are five chopsticks/forks kept beside them and a bowl of rice in the centre, When a philosopher wants to eat, he uses two chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place.

Readers and Writers Problem

- In this problem there are some processes(called **readers**) that only read the shared data, and never change it, and there are other processes(called **writers**) who may change the data in addition to reading, or instead of reading it.
- There are various type of readers-writers problem, most centred on relative priorities of readers and writers.
- The main complexity with this problem occurs from allowing more than one reader to access the data at the same time.

UNIT – IV

1Q) Physical and Virtual Address Space

Ans

Physical Address

- The physical address identifies the physical location of required data in memory.
- The user never directly deals with the physical address but can access it by its corresponding logical address.
- The user program generates the logical address and thinks it is running in it, but the program needs physical memory for its execution. Therefore, the logical address must be mapped to the physical address by MMU before they are used.
- The Physical Address Space is used for all physical addresses corresponding to the logical addresses in a logical address space.

Virtual Address

- A logical address is an address that is generated by the CPU during program execution.
- The logical address is a virtual address as it does not exist physically, and therefore, it is also known as a ***Virtual Address***.
- This address is used as a reference to access the physical memory location by CPU.
- The term Logical Address Space is used to set all logical addresses generated from a program's perspective.
- A logical address usually ranges from zero to maximum (max). The user program that generates the logical address assumes that the process runs on locations between 0 and max. This logical address (generated by CPU) combines with the ***base address*** generated by the MMU to form the physical address.
- The hardware device called Memory-Management Unit is used for mapping logical addresses to their corresponding physical address.

2Q) Difference between Logical Address and Physical Address

Ans:

Logical Address	Physical Address
the CPU generates the logical address while	The physical address is a location in

the program is running	memory.
The logical address does not exist physically in the memory, and therefore it is sometimes known as a virtual address.	The physical address is a location in the memory unit.
The logical address is used as a reference to access the physical address.	The physical address cannot be accessed directly.
The set of all the logical addresses generated about a program by the CPU is called Logical Address Space.	Whereas all the physical addresses mapped to the logical address is called Physical Address Space.

3Q) Memory Allocation Strategies

Ans: There are various methods which can be used to allocate disk space to the files. Selection of an appropriate allocation method will significantly affect the performance and efficiency of the system. Allocation method provides a way in which the disk will be utilized and the files will be accessed.

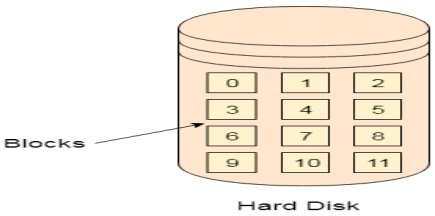
There are following methods which can be used for allocation.

1. Contiguous Allocation.
2. Extents
3. Linked Allocation
4. Clustering
5. FAT
6. Indexed Allocation
7. Linked Indexed Allocation
8. Multilevel Indexed Allocation
9. Inode

Contiguous Allocation

If the blocks are allocated to the file in such a way that all the logical blocks of the file get the contiguous physical block in the hard disk then such allocation scheme is known as contiguous allocation.

In the image shown below, there are three files in the directory. The starting block and the length of each file are mentioned in the table. We can check in the table that the contiguous blocks are assigned to each file as per its need.



File Name	Start	Length	Allocated Blocks
abc.text	0	3	0,1,2
video.mp4	4	2	4,5
jtp.docx	9	3	9,10,11

Directory

Contiguous Allocation

Advantages

1. It is simple to implement.
2. We will get Excellent read performance.
3. Supports Random Access into files.

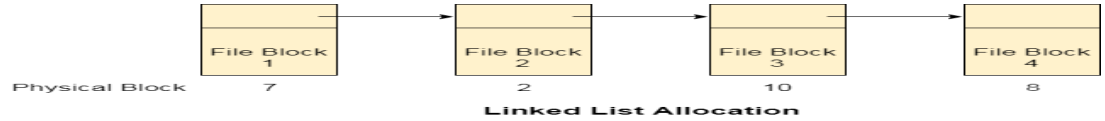
Disadvantages

1. The disk will become fragmented.

2. It may be difficult to have a file grow.

Linked List Allocation

Linked List allocation solves all problems of contiguous allocation. In linked list allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.



Linked List Allocation

Advantages

1. There is no external fragmentation with linked allocation.
2. Any free block can be utilized in order to satisfy the file block requests.
3. File can continue to grow as long as the free blocks are available.
4. Directory entry will only contain the starting block address.

Disadvantages

1. Random Access is not provided.
2. Pointers require some space in the disk blocks.
3. Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.
4. Need to traverse each block.

Indexed Allocation

File allocation table tries to solve as many problems as possible but leads to a drawback. The more the number of blocks, the more will be the size of FAT.

Therefore, we need to allocate more space to a file allocation table. Since, file allocation table needs to be cached therefore it is impossible to have as many space in cache. Here we need a new technology which can solve such problems.

Indexed Allocation Scheme

Instead of maintaining a file allocation table of all the disk pointers, Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block. Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file. Directory entry will only contain the index block address.

Advantages

1. Supports direct access
2. A bad data block causes the lost of only that block.

Disadvantages

1. A bad index block could cause the lost of entire file.
2. Size of a file depends upon the number of pointers, a index block can hold.
3. Having an index block for a small file is totally wastage.
4. More pointer overhead

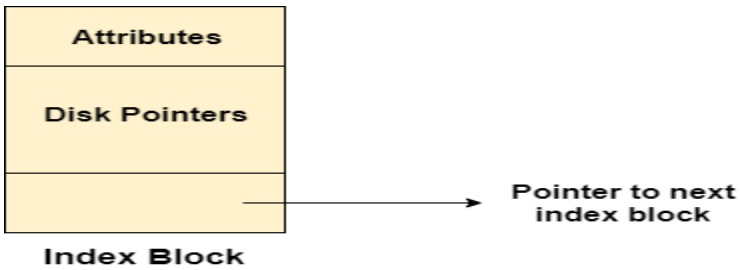
Linked Index Allocation

Single level linked Index Allocation

In index allocation, the file size depends on the size of a disk block. To allow large files, we have to link several index blocks together. In linked index allocation,

- Small header giving the name of the file
- Set of the first 100 block addresses
- Pointer to another index block

For the larger files, the last entry of the index block is a pointer which points to another index block. This is also called as linked schema.



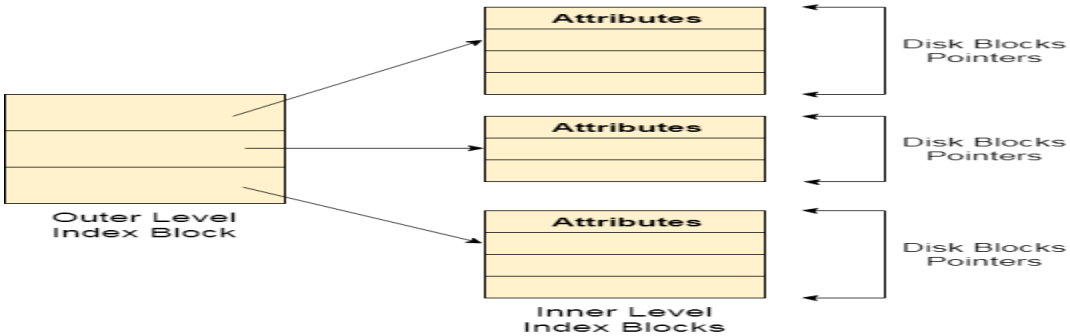
Advantage: It removes file size limitations
Disadvantage: Random Access becomes a bit harder

Multilevel Index Allocation

In Multilevel index allocation, we have various levels of indices. There are outer level index blocks which contain the pointers to the inner level index blocks and the inner level index blocks contain the pointers to the file data.

- The outer level index is used to find the inner level index.
- The inner level index is used to find the desired data block.

Advantage: Random Access becomes better and efficient.
Disadvantage: Access time for a file will be higher.



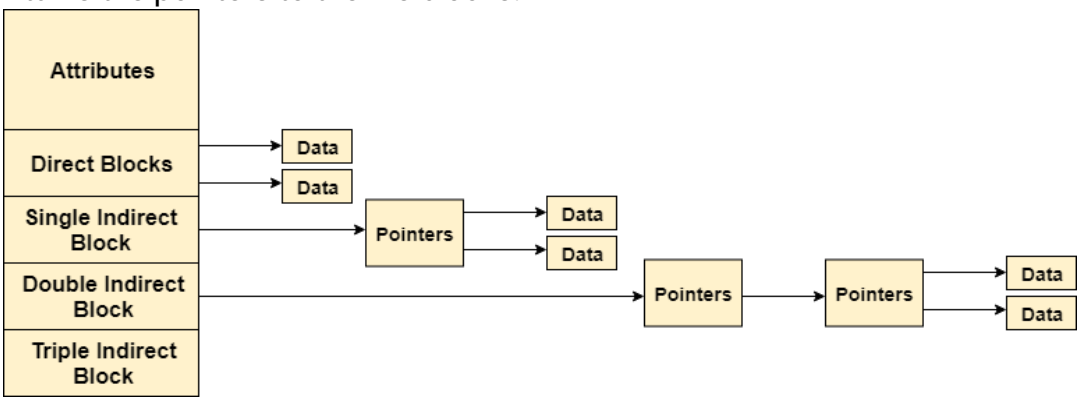
Inode

In UNIX based operating systems, each file is indexed by an Inode. Inode are the special disk block which is created with the creation of the file system. The number of files or directories in a file system depends on the number of Inodes in the file system.

An Inode includes the following information

1. Attributes (permissions, time stamp, ownership details, etc) of the file
2. A number of direct blocks which contains the pointers to first 12 blocks of the file.
3. A single indirect pointer which points to an index block. If the file cannot be indexed entirely by the direct blocks then the single indirect pointer is used.
4. A double indirect pointer which points to a disk block that is a collection of the pointers to the disk blocks which are index blocks. Double index pointer is used if the file is too big to be indexed entirely by the direct blocks as well as the single indirect pointer.
5. A triple index pointer that points to a disk block that is a collection of pointers.

Each of the pointers is separately pointing to a disk block which also contains a collection of pointers which are separately pointing to an index block that contains the pointers to the file blocks.



4Q) Fixed Partitioning

Ans The earliest and one of the simplest technique which can be used to load more than one processes into the main memory is Fixed partitioning or Contiguous memory allocation.

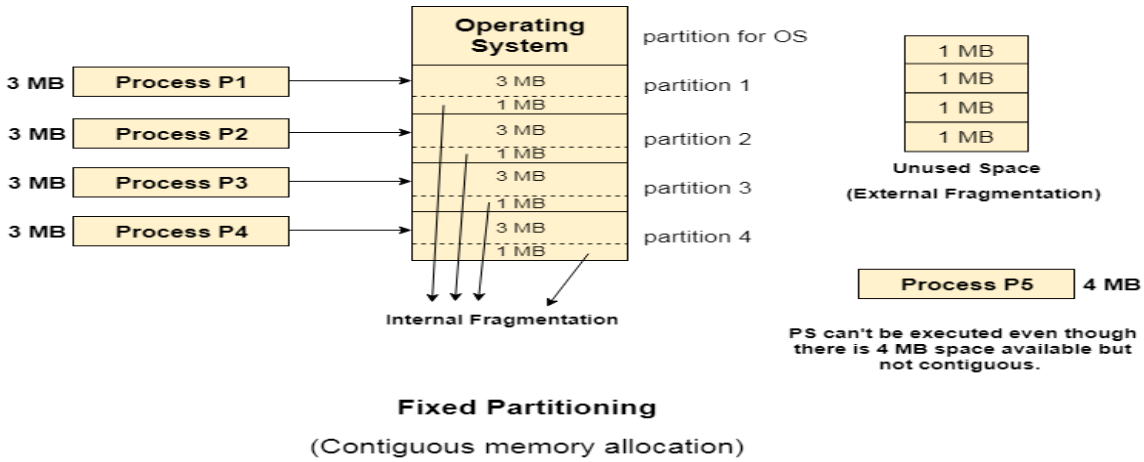
In this technique, the main memory is divided into partitions of equal or different sizes. The operating system always resides in the first partition while the other partitions can be used to store user processes. The memory is assigned to the processes in contiguous way.

In fixed partitioning,

1. The partitions cannot overlap.
2. A process must be contiguously present in a partition for the execution.

There are various cons of using this technique.

1. **Internal Fragmentation**
   If the size of the process is lesser then the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation.
   As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.
2. **External Fragmentation**
   The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.
   As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process will not be loaded.
3. **Limitation on the size of the process**
   If the process size is larger than the size of maximum sized partition then that process cannot be loaded into the memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.
4. **Degree of multiprogramming is less**
   By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.



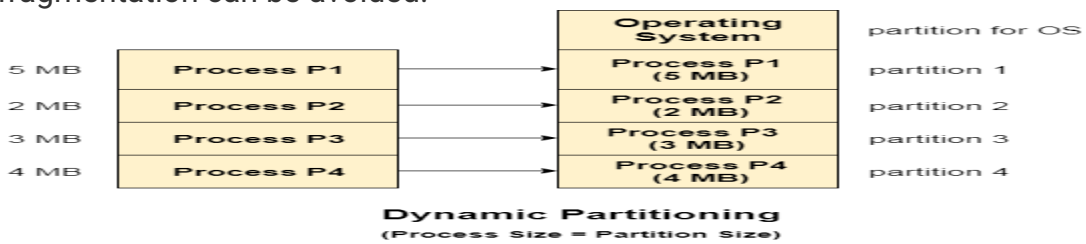
Fixed Partitioning

(Contiguous memory allocation)

Dynamic Partitioning

Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading.

The first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.



Dynamic Partitioning

(Process Size = Partition Size)

Advantages of Dynamic Partitioning over fixed partitioning

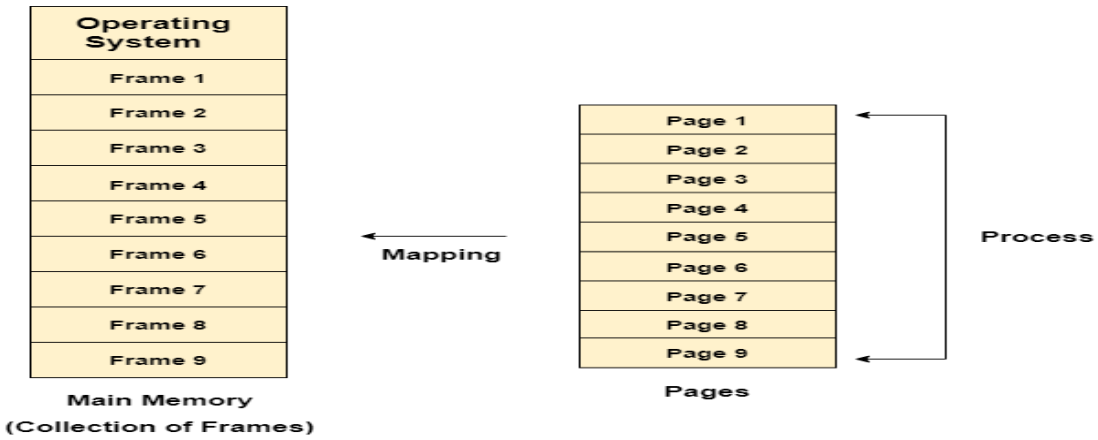
1. **No Internal Fragmentation**
   Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.
2. **No Limitation on the size of the process**
   In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.
3. **Degree of multiprogramming is dynamic**
   Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.

5Q) Paging with Example

Ans:

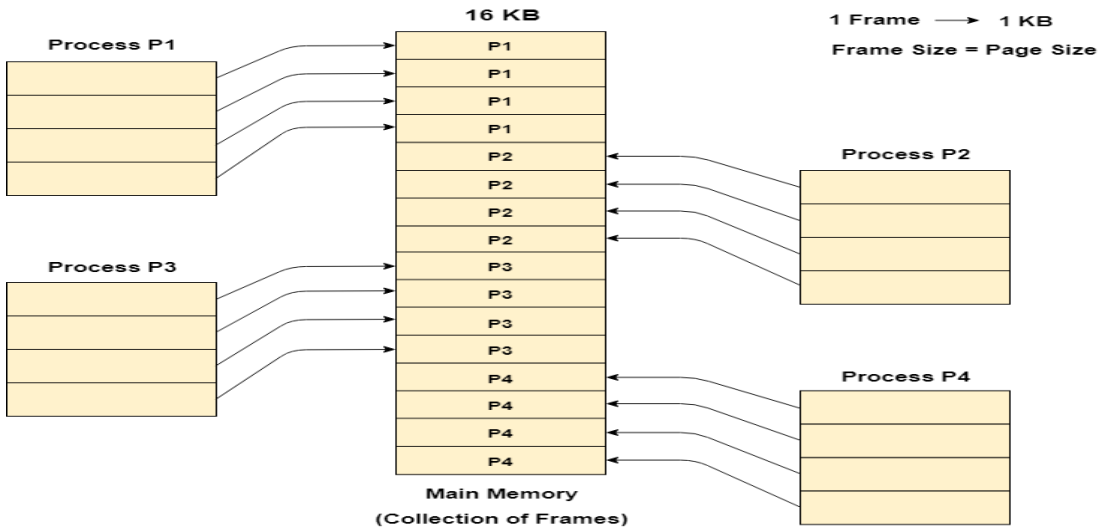
- In Operating Systems, Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages.

- The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.
- One page of the process is to be stored in one of the frames of the memory. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.
- Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.
- Different operating system defines different frame sizes. The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame size.



Example

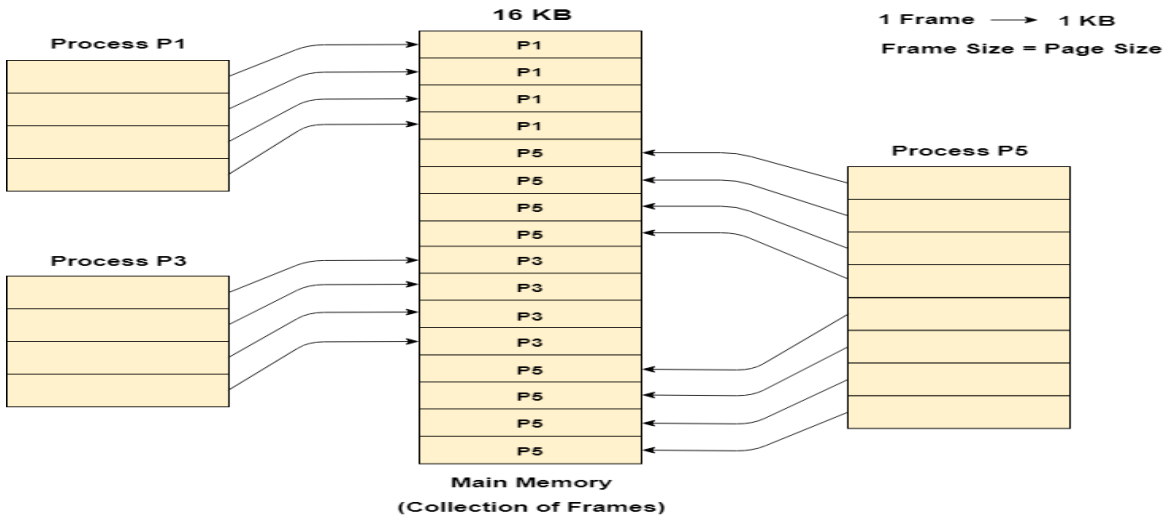
- Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.
- There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each. Each process is divided into pages of 1 KB each so that one page can be stored in one frame.
- Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.
- Frames, pages and the mapping between the two is shown in the image below.



Paging

Let us consider that, P2 and P4 are moved to waiting state after some time. Now, 8 frames become empty and therefore other pages can be loaded in that empty place. The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.



Paging

6Q) Segmentation

Ans: In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments. Segment table contains mainly two information about segment -

1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

Advantages of Segmentation

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.

Disadvantages

1. It can have external fragmentation.
2. It is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

7Q) Paging VS Segmentation

Paging	Segmentation
Non-Contiguous memory allocation	Non-contiguous memory allocation
Paging divides program into fixed size pages.	Segmentation divides program into variable size segments.
OS is responsible	Compiler is responsible.
Paging is faster than segmentation	Segmentation is slower than paging
Paging is closer to Operating System	Segmentation is closer to User
It suffers from internal fragmentation	It suffers from external fragmentation
There is no external fragmentation	There is no external fragmentation
Logical address is divided into page number and page offset	Logical address is divided into segment number and segment offset
Page table is used to maintain the page information.	Segment Table maintains the segment information
Page table entry has the frame number and some flag bits to represent details about pages.	Segment table entry has the base address of the segment and some protection bits for the segments.

8Q) Virtual Memory

Ans:

- Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.
- In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.
- Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.
- By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

How Virtual Memory Works?

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the

execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced. We will discuss each one of them later in detail.

Advantages of Virtual Memory

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

UNIT V

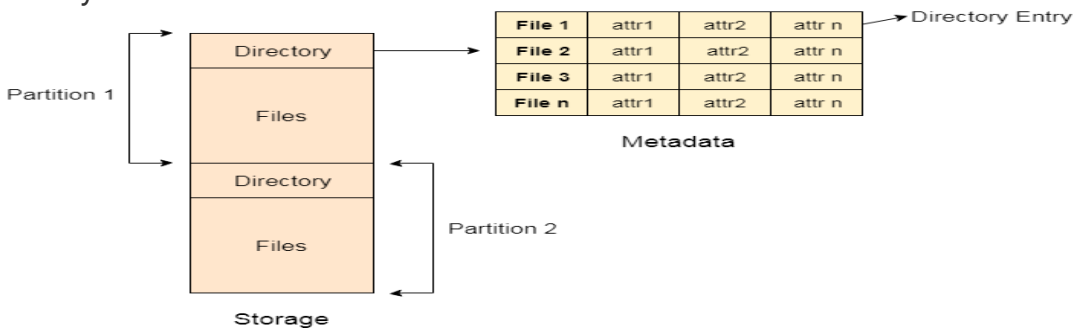
1Q) Directory Structure

Ans:

Directory

- Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

- To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.
- Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that file.



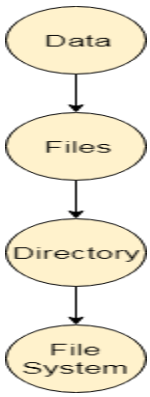
A directory can be viewed as a file which contains the Meta data of the bunch of files. Every Directory supports a number of common operations on the file:

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files

2Q) What is File? Write the File Operations?

Ans: A file can be defined as a data structure which stores the sequence of records. Files are stored in a file system, which may exist on a disk or in the main memory. Files can be simple (plain text) or complex (specially-formatted).

The collection of files is known as Directory. The collection of directories at the different levels, is known as File System.



Attributes of the File

1. **Name**
   Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.
2. **Identifier**
   Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension **.txt,** A video file can have the

extension **.mp4.**

3. **Type**
   In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.
4. **Location**
   In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.
5. **Size**
   The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.
6. **Protection**
   The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of Users.
7. **Time and Date**
   Every file carries a time stamp which contains the time and date on which the file is last modified.

3Q) Operations on the File

Ans:

A file is a collection of logically related data that is recorded on the secondary storage in the form of sequence of operations. The content of the files are defined by its creator who is creating the file. The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations. These operations are performed by the user by using the commands provided by the operating system. Some common operations are as follows:

1. **Create operation:**
   This operation is used to create a file in the file system. It is the most widely used operation performed on the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.
2. **Open operation:**
   This operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.
3. **Write operation:**
   This operation is used to write the information into a file. A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.
4. **Read operation:**
   This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.
5. **Re-position or Seek operation:**
   The seek system call re-positions the file pointers from the current position to a

specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.

6. **Delete operation:**
   Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.
7. **Truncate operation:**
   Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.
8. **Close operation:**
   When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it deallocates all the internal descriptors that were created when the file was opened.
9. **Append operation:**
   This operation adds data to the end of the file.
10. **Rename operation:**
    This operation is used to rename the existing file.

4Q) Device Management in Operating System

Ans: Device management in an operating system means controlling the Input/Output devices like disk, microphone, keyboard, printer, magnetic tape, USB ports, camcorder, scanner, other accessories, and supporting units like supporting units control channels. A process may require various resources, including main memory, file access, and access to disk drives, and others. If resources are available, they could be allocated, and control returned to the CPU. Otherwise, the procedure would have to be postponed until adequate resources become available. The system has multiple devices, and in order to handle these physical or virtual devices, the operating system requires a separate program known as an ad device controller. It also determines whether the requested device is available.

The fundamentals of I/O devices may be divided into three categories:

1. **Boot Device**
2. **Character Device**
3. **Network Device**

Boot Device

It stores data in fixed-size blocks, each with its unique address. For example- Disks.

Character Device

It transmits or accepts a stream of characters, none of which can be addressed individually. For instance, keyboards, printers, etc.

Network Device: It is used for transmitting the data packets.

Types of devices

There are three types of Operating system peripheral devices: dedicated, shared, and virtual. These are as follows:

1. **Dedicated Device**
   In device management, some devices are allocated or assigned to only one task at a time until that job releases them. Devices such as plotters, printers, tape drivers, and other similar devices necessitate such an allocation

mechanism because it will be inconvenient if multiple people share them simultaneously. The disadvantage of such devices is the inefficiency caused by allocating the device to a single user for the whole duration of task execution, even if the device is not used 100% of the time.

2. **Shared Devices**
   These devices could be assigned to a variety of processes. By interleaving their requests, disk-DASD could be shared by multiple processes simultaneously. The Device Manager carefully controls the interleaving, and pre-determined policies must resolve all difficulties.

3. **Virtual Devices**

   Virtual devices are a hybrid of the two devices, and they are dedicated devices that have been transformed into shared devices. For example, a printer can be transformed into a shareable device by using a spooling program that redirects all print requests to a disk. A print job is not sent directly to the printer; however, it is routed to the disk until it is fully prepared with all of the required sequences and formatting, at which point it is transmitted to the printers. The approach can transform a single printer into numerous virtual printers, improving performance and ease of use.

Features of Device Management

1. The OS interacts with the device controllers via the device drivers while allocating the device to the multiple processes executing on the system.
2. Device drivers can also be thought of as system software programs that bridge processes and device controllers.
3. The device management function's other key job is to implement the API.
4. Device drivers are software programs that allow an operating system to control the operation of numerous devices effectively.
5. The device controller used in device management operations mainly contains three registers: command, status, and data.

5Q) Pipes

Ans Communication in client–server systems may use (1) sockets, (2) remote procedure calls (RPCs), or (3) pipes. A socket is defined as an endpoint for communication. A connection between a pair of applications consists of a pair of sockets, one at each end of the communication channel. RPCs are another form of distributed communication. An RPC occurs when a process (or thread) calls a procedure on a remote application. Pipes provide a relatively simple ways for processes to communicate with one another. Ordinary pipes allow communication between parent and child processes, while named pipes permit unrelated processes to communicate.

A pipe acts as a conduit allowing two processes to communicate. Pipes were one of the first IPC mechanisms in early UNIX systems. They typically provide one of the simpler ways for processes to communicate with one another, although they also have some limitations. In implementing a pipe, four issues must be considered:

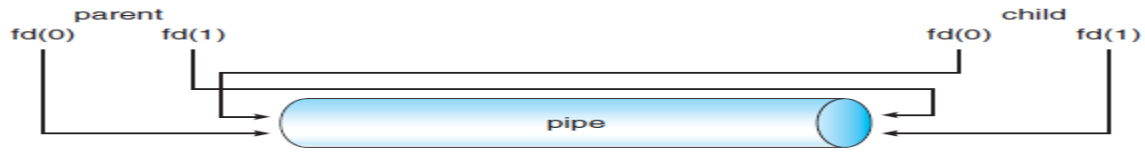
1. Does the pipe allow bidirectional communication, or is communication unidirectional?
2. If two-way communication is allowed, is it half duplex (data can travel only one way at a time) or full duplex (data can travel in both directions at the same time)?
3. Must a relationship (such as parent–child) exist between the communicating processes?
4. Can the pipes communicate over a network, or must the communicating processes reside on the same machine?

Two common types of pipes used on both UNIX and Windows systems: ordinary pipes and named pipes.

Ordinary Pipes

ordinary pipes are unidirectional, allowing only one-way communication-Ordinary pipes allow two processes to communicate in standard producer– consumer fashion: the producer writes to one end of the pipe (the write-end) and the consumer reads from the other end (the read-end).

If two-way communication is required, two pipes must be used, with each pipe sending data in a different direction.



File descriptors for an ordinary pipe.

On UNIX systems, ordinary pipes are constructed using the function pipe(int fd[]) This function creates a pipe that is accessed through the int fd[] file descriptors: fd[0] is the read-end of the pipe, and fd[1] is the write-end.

- UNIX treats a pipe as a special type of file. Thus, pipes can be accessed using ordinary read() and write() system calls.
- An ordinary pipe cannot be accessed from outside the process that created it.
- Typically, a parent process creates a pipe and uses it to communicate with a child process that it creates via fork().

6Q) Buffering in Operating System

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 or online video streaming.

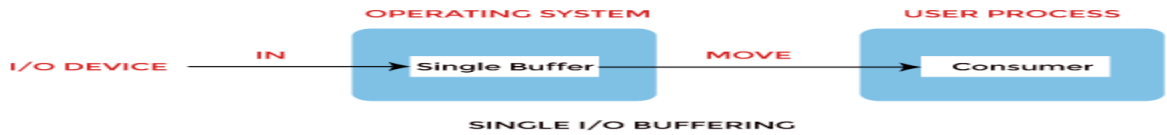
A buffer often adjusts timing by implementing a queue or FIFO algorithm in memory, simultaneously writing data into the queue at one rate and reading it at another rate.

Types of Buffering

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

a. **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.



SINGLE I/O BUFFERING

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.

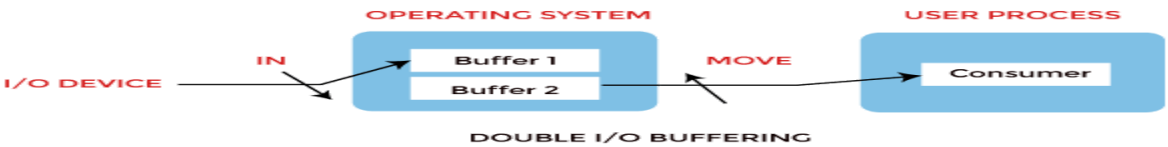
Stream oriented device:

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.

b. **Double Buffer**

In **Double Buffering**, two schemes or two buffers are used in the place of one. 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.



DOUBLE I/O BUFFERING

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.

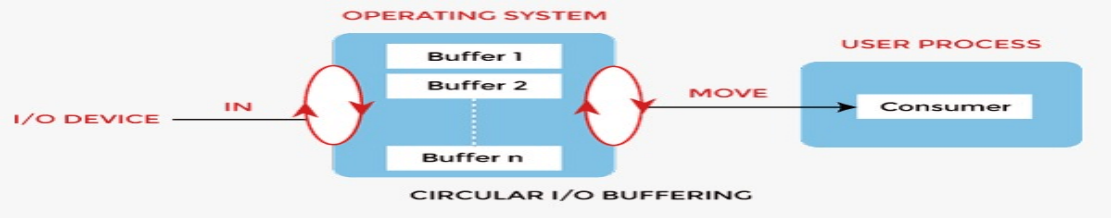
Stream oriented:

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.

c. **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.

Advantages of Buffer

- The use of buffers allows uniform disk access. It simplifies system design.
- The system places no data alignment restrictions on user processes doing I/O. By copying data from user buffers to system buffers and vice versa, the kernel eliminates the need for special alignment of user buffers, making user programs simpler and more portable.
- The use of the buffer can reduce the amount of disk traffic, thereby increasing overall system throughput and decreasing response time.

Disadvantages of Buffer

- It is costly and impractical to have the buffer be the exact size required to hold the number of elements. Thus, the buffer is slightly larger most of the time, with the rest of the space being wasted.
- Buffers have a fixed size at any point in time. When the buffer is full, it must be reallocated with a larger size, and its elements must be moved. Similarly, when the number of valid elements in the buffer is significantly smaller than its size, the buffer must be reallocated with a smaller size and elements be moved to avoid too much waste.
- Use of the buffer requires an extra data copy when reading and writing to and from user processes. When transmitting large amounts of data, the extra copy slows down performance.

7Q) Shared Memory

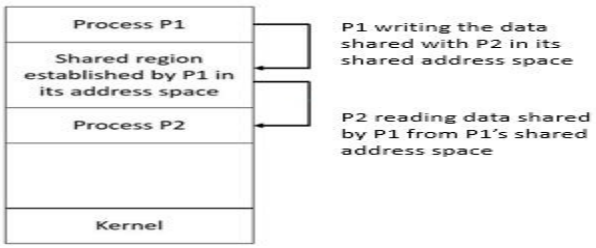
Ans: Shared memory system is the fundamental model of inter process communication. In a shared memory system, in the address space region the cooperating communicate with each other by establishing the shared memory region. Shared memory concept works on fastest inter process communication.

If the process wants to initiate the communication and it has some data to share, then establish the shared memory region in its address space. After that, another process wants to communicate and tries to read the shared data, and must attach itself to the initiating process's shared address space.

Working of Shared Memory

In the Shared Memory system, the cooperating processes communicate, to exchange the data with each other. Because of this, the cooperating processes establish a shared region in their memory. The processes share data by reading and writing the data in the shared segment of the processes.

Let us discuss it by considering two processes. The diagram is shown below –



Let the two cooperating processes P1 and P2. Both the processes P1 and P2, have their different address spaces. Now let us assume, P1 wants to share some data with P2. So, P1 and P2 will have to perform the following steps –

Step 1 – Process P1 has some data to share with process P2. First P1 takes initiative and establishes a shared memory region in its own address space and stores the data or information to be shared in its shared memory region.

Step 2 – Now, P2 requires the information stored in the shared segment of P1. So, process P2 needs to attach itself to the shared address space of P1. Now, P2 can read out the data from there.

Step 3 – The two processes can exchange information by reading and writing data in the shared segment of the process.

Advantages

- Shared memory is a faster inter process communication system.
- It allows cooperating processes to access the same pieces of data concurrently.
- Modularity is achieved in a shared memory system.
- Users can perform multiple tasks at a time.

8Q) Security Policy mechanism and Protection

Ans: A security policy is a statement of what is and what is not allowed. A security mechanism is a method or procedure for enforcing a security policy. Mechanisms can be nontechnical, such as requiring proof of identity before changing a password; in fact policies often require some procedural mechanisms that technology cannot enforce. The commonly accepted aspects of security are as follows:

1. Identification and authentication
2. Authorization
3. Auditing
4. Confidentiality
5. Data Integrity

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
- Message digests and digital signatures
- Digital certificates
- Public key infrastructure

Identification and authentication

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:

Authorization protects critical resources in a system by limiting access only to authorized uses and their applications. It prevents the unauthorized use of a resource or the use of a resource in an unauthorized manner

Auditing

It is the process of recording and checking events to detect whether any unexpected or unauthorized activity has taken place or a whether any attempt has been made to perform such activity.

Confidentiality

The confidentiality service protects sensitive information from unauthorized disclosure.

Data Integrity

The data integrity service detects whether there has been unauthorized modification of data.

Cryptographic concepts

Cryptographic protocols provide secure connections, enabling two parties to communicate with privacy and data integrity. The transport Layer security protocol evolved from that of the secure sockets layer

9Q) Write about Authentication and Internal Access Authorization.

Ans: Authentication process can be described in two distinct phases:

1. Identification
2. Actual Authentication

Identification

Identification phase provides a user identity to the security systems. This identity is provided in the form of a user ID. The security system will search all the abstract objects that it know and find the specific one of which the actual user is currently applying. Once this is done the user has been identified.

Actual Authorization

The process of determining claimed user identity by checking user-provided evidence is called authentication and the evidence which is provided by the user during process of authentication is called a credential

Authorization is a security mechanism to determine access levels or user/client privileges related to system resources including files, services , computer programs, data and application features. This is the process of granting or denying access to a network resource which allows the user access to various resources based on the user's identity.

Most Web security systems are based on a two-step process. The first step is authentication, which ensures about the user identity and the second stage is authorization which allows the user to access the various resources based on the user's identity.

Internal Access Control

Access control in computer systems and networks relies on access policies and it is

divided into two phases

1. Policy definition phase where access is authorized
2. Policy enforcement phase where access requests are permitted or not permitted

Thus authorization is the function of the policy definition phase which precedes the policy enforcement phase where access requests are permitted or not permitted based on the previously defined authorizations. Access control also uses authentication to check the identity of consumers. When a consumer attempts to access a resource, the access control process investigates that the consumer has been authorized to use that resource. Authorization services are implemented by the security server which can control access at the level of individual files or programs

10Q) Write about Authentication

Ans:

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.