

### 1] Define OS. List out any 2 examples.

**Operating System** lies in the category of system software. It basically manages all the resources of the computer.

A program that acts as an intermediary between a user of a computer and the computer hardware.

The operating system is designed in such a way that it can manage the overall resources and operations of the computer.

Examples

- Microsoft Windows.
- Mac OS.
- Android OS.
- Linux.
- Ubuntu.
- Chrome OS.
- Fedora.

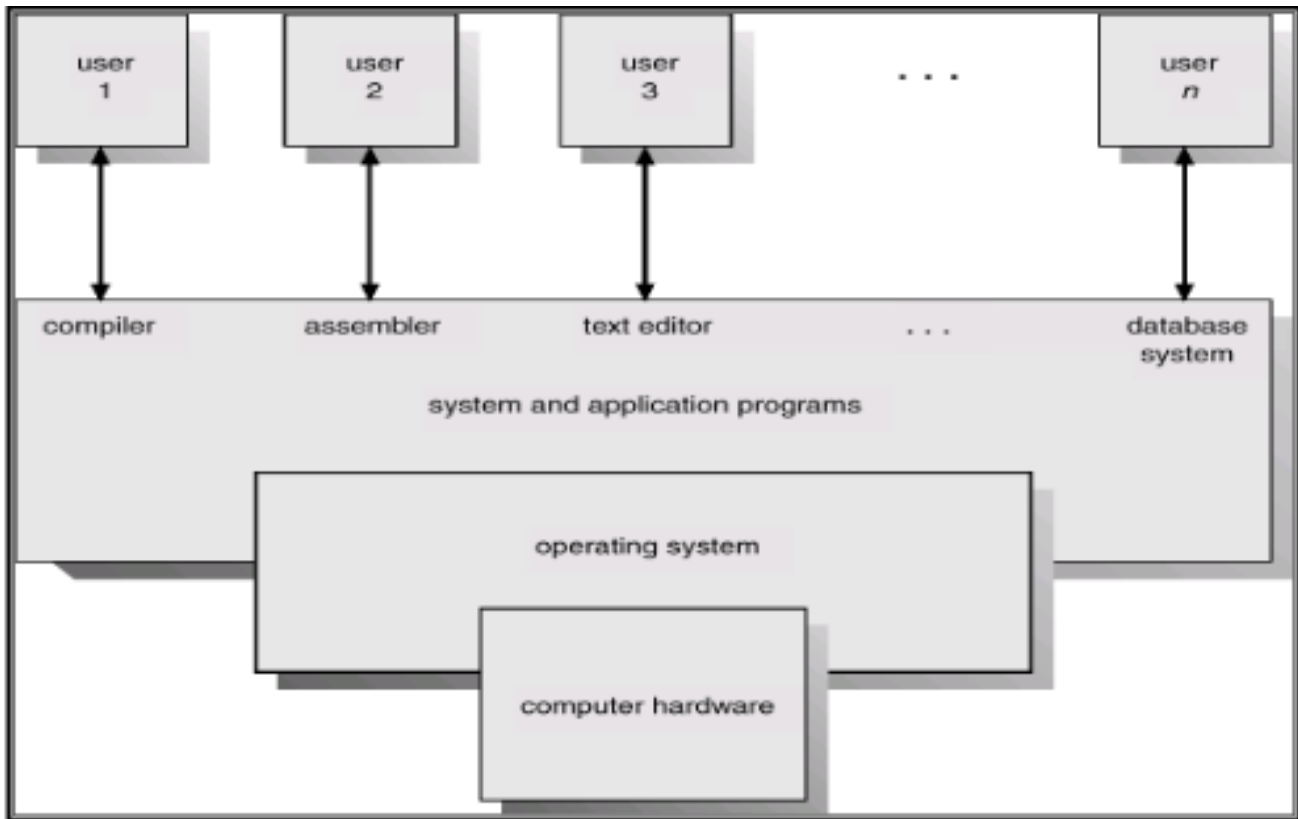
### 2] List out goals of an OS.

- Execute user programs and make solving user problems easier.
- Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.
- To hide the details of the hardware resources from the users.
- To manage the resources of a computer system.
- To execute and provide services for applications software.

### 3] Explain comp sys components with prop diag.

1. Hardware – provides basic computing resources (CPU, memory, I/O devices).
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.

3. Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
4. Users (people, machines, other computers).



4] Explain diff services of an os.

- Memory Management
- Device Management
- Processor Management
- Security
- Error detecting aids
- Coordination between other software and users
- Job accounting
- File Management

## Memory Management

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

## Processor Management

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

## Device Management

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

## File Management

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

## Security :-

- By means of password and similar other techniques, it prevents unauthorized access to programs and data.

## Job accounting :-

- Keeping track of time and resources used by various jobs and users.

### Error detecting aids :-

Production of dumps, traces, error messages, and other debugging and error detecting aids.

### Coordination between other software and users :-

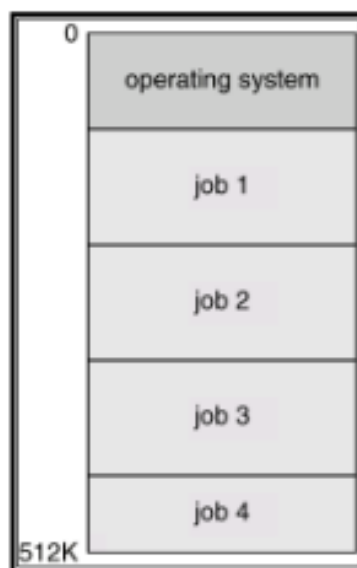
Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

5] Enlist the types of an OS.

- Batch Operating System
- Multiprogramming operating system
- Time-sharing Operating Systems
- Distributed Operating System
- Real-Time Operating System

6] Explain multi-programming OS

Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

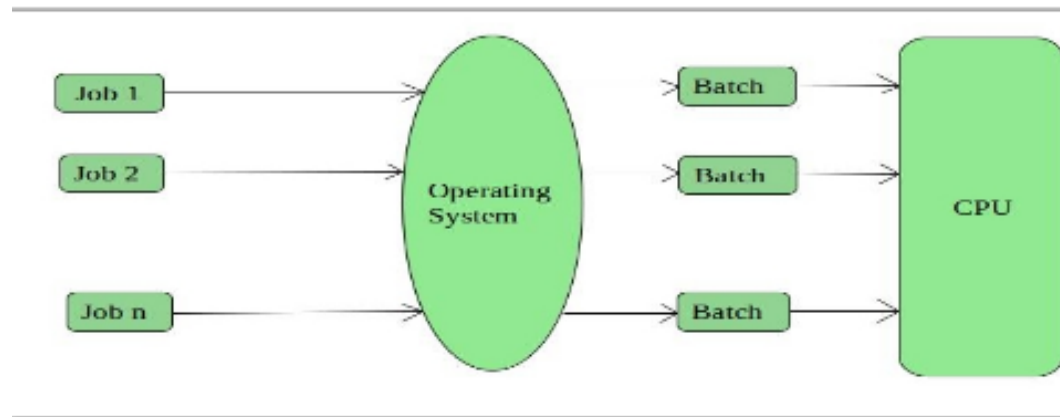


## OS Features Needed for Multiprogramming

- I/O routine supplied by the system.
- Memory management – the system must allocate the memory to several jobs.
- CPU scheduling – the system must choose among several jobs ready to run.
- Allocation of devices.

On a single processor computer, a multiprogramming OS can run many programs. In a multiprogramming OS, if one program must wait for an input/output transfer, the other programmes are ready to use the CPU. As a result, different jobs may have to split CPU time. However, their jobs are not scheduled to be completed at the same time.

## 7] Explain Batch-OS.



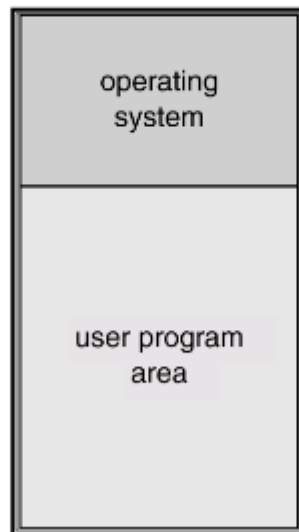
- ❖ This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.

11

- In a punch card system, human intervention was required for each punch card, which was an overhead.
- Thus in Batch Systems, a collection of cards known as batch used to submit reducing human intervention.
- 2 Types of Cards were used :
  - 1. Data Cards : used to contain actual information
  - 2. Control Cards : Used to control the flow of execution.

12

## Memory Layout for a Simple Batch System



13

8] Write about distributed os.

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
  - Resources Sharing
  - Computation speed up – load sharing
  - Reliability
  - Communications
- Requires networking infrastructure.
- Local area networks (LAN) or Wide area networks (WAN)
- May be either client-server or peer-to-peer systems.

## Applications of Distributed Operating System

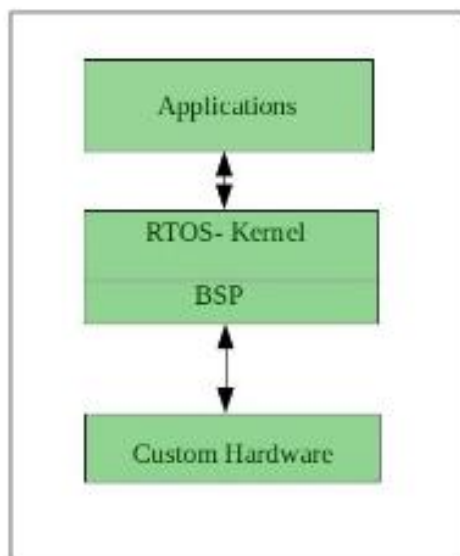
The applications of distributed OS are as follows –

- Internet Technology
- Distributed databases System
- Air Traffic Control System
- Airline reservation Control systems
- Peep-to-peer networks system
- Telecommunication networks

9] Write a note on real time os.

RTOSes are designed to handle multiple processes at one time, ensuring that these processes respond to events within a predictable time limit.

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Real-Time systems may be either *hard* or *soft* real-time.





- Hard real-time:
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time
  - Limited utility in industrial control of robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

24

### Examples

- air traffic control systems;
- anti-lock brakes and air bags;
- cameras;
- medical systems; and
- PCs.

10] Difference between CLI and GUI.

Features	CLI	GUI
<b>Definition</b>	A CLI is an interface that allows the user to perform tasks by issuing commands in successive lines of text or command lines.	A graphical user interface enables users to interact with the operating system or application.
<b>Memory Requirement</b>	It needs less memory than the GUI.	It needs more memory because it has various graphics components.
<b>Ease of use</b>	It is not easy to use.	It is easy to use.
<b>Speed</b>	It is faster than the GUI.	It is slower than the CLI.
<b>Flexibility</b>	It is less flexible than GUI.	It is more flexible than CLI.
<b>Device Used</b>	It needs the only keyboard.	It needs both a keyboard and a mouse.
<b>Appearance</b>	Its appearance may not be modified or changed.	Its appearance may be modified or changed.
<b>Precision</b>	Its precision is high as compared to GUI.	Its precision is low as compared to CLI.
<b>Data Presentation</b>	The information can be viewed to the user in plain text and files in the CLI.	In a GUI, information can be viewed or presented to the user in several ways, including simple text, videos, graphics, etc.
<b>Errors</b>	Spelling mistakes and typing errors are not avoided.	Spelling mistakes and typing errors are avoided.
<b>Graphics</b>	No graphics are used in the CLI.	Graphics are used in the GUI.
<b>Menus</b>	No menus are provided in the CLI.	Menus are provided in the GUI.

<b>11</b>	<b>Explain two different types of services of operating system</b>
<b>A</b>	<p><b>Types of OS Services</b></p> <p>Different types of services provided by an operating system (OS) are essential for managing computer resources and providing a user-friendly environment. Here's a breakdown:</p> <ol style="list-style-type: none"> <li><b>1. Program Execution:</b> The OS loads programs into memory and schedules their execution. It allocates necessary resources and ensures fairness in resource sharing among different programs. Context switching allows smooth multitasking.</li> <li><b>2. I/O Operations:</b> The OS handles input and output operations to devices like keyboards, displays, disks, and network interfaces. It provides device drivers to mediate communication between software and hardware. Buffering and spooling techniques optimize data transfer.</li> <li><b>3. File Manipulation:</b> The OS manages files and directories, providing services like file creation, deletion, reading, and writing. It maintains a file system that organizes data on storage devices and implements access control and security measures.</li> <li><b>4. Communication Services:</b> Inter-process communication (IPC) services enable data sharing between different processes. This can involve message passing, shared memory, or socket-based communication, crucial for collaborative tasks and networking.</li> <li><b>5. Error Detection &amp; Handling:</b> The OS monitors system activities for errors and inconsistencies. It alerts users or administrators about potential issues and implements recovery mechanisms to prevent system crashes. Techniques like error codes and exception handling are used.</li> </ol>
<b>12</b>	<b>Explain different types of services of operating system for users</b>
<b>A</b>	<ol style="list-style-type: none"> <li><b>1. Security and Access Control:</b> It safeguards data and resources, controlling user access through authentication and authorization.</li> <li><b>2. Memory Management:</b> The OS allocates and deallocates memory to processes, optimizing usage and preventing conflicts.</li> <li><b>3. Task Scheduling:</b> It prioritizes and schedules processes, ensuring fair CPU time allocation and efficient multitasking.</li> <li><b>4. User Interface:</b> The OS provides a user-friendly interface, allowing users to interact with the system through graphical or command-based methods.</li> <li><b>5. Networking Services:</b> It enables network connections, supporting tasks like internet browsing, email, and file sharing.</li> <li><b>6. Program Execution:</b> The OS allows users to run programs efficiently, managing resources for optimal performance.</li> <li><b>7. File Management:</b> It manages files and directories, providing organization and access control.</li> <li><b>8. I/O Operations:</b> It handles input and output devices, ensuring data exchange between users and hardware.</li> <li><b>9. Communication Services:</b> The OS facilitates communication between processes, both on the same computer and across networks.</li> </ol>
<b>13</b>	<b>Explain different types of services of operating system for OS itself</b>

A	**Same as Q.11**
14	<b>Define System Call and state any 2 system call with their example/function</b>
A	<p>A System Call is the interface between user-level processes and the operating system. It allows processes to request services from the OS, such as input/output operations, process control, and file management.</p> <p>Here are four examples of system calls along with their functions:</p> <ol style="list-style-type: none"> <li>1. <b>open()</b> <ul style="list-style-type: none"> <li>• <b>Function:</b> Used to open a file and obtain a file descriptor.</li> <li>• <b>Example:</b> <code>int fd = open("file.txt", O_RDONLY);</code></li> </ul> </li> <li>2. <b>read()</b> <ul style="list-style-type: none"> <li>• <b>Function:</b> Reads data from a file descriptor into a buffer.</li> <li>• <b>Example:</b> <code>ssize_t bytes_read = read(fd, buffer, sizeof(buffer));</code></li> </ul> </li> <li>3. <b>write()</b> <ul style="list-style-type: none"> <li>• <b>Function:</b> Writes data from a buffer to a file descriptor.</li> <li>• <b>Example:</b> <code>ssize_t bytes_written = write(fd, buffer, bytes_to_write);</code></li> </ul> </li> <li>4. <b>fork()</b> <ul style="list-style-type: none"> <li>• <b>Function:</b> Creates a new process (child) by duplicating the existing process (parent).</li> <li>• <b>Example:</b> <code>pid_t child_pid = fork();</code></li> </ul> </li> </ol>
15	<b>State any four types of System Call</b>
A	<ol style="list-style-type: none"> <li>1. <b>Process Control:</b> System calls that manage processes, like <code>fork()</code>, <code>exec()</code>, and <code>exit()</code>.</li> <li>2. <b>File Management:</b> System calls for file operations, including <code>open()</code>, <code>read()</code>, <code>write()</code>, and <code>close()</code>.</li> <li>3. <b>Device Management:</b> Calls to handle devices, such as <code>read()</code>/<code>write()</code> for devices and <code>ioctl()</code> for device control.</li> <li>4. <b>Information Maintenance:</b> Calls to get system or process information, like <code>getpid()</code>, <code>getuid()</code>, and <code>time()</code>.</li> </ol>
16	<b>Describe various activities performed by following OS Components (Any 2 for each)</b>
	<ol style="list-style-type: none"> <li>a) Main Memory Management</li> <li>b) File Management</li> <li>c) Process Management</li> <li>d) Secondary Storage Management</li> </ol>
A	<p><b>a) Main Memory Management:</b> Main memory is a critical resource in an operating system. Its management involves several activities to ensure efficient utilization and allocation of memory for running processes.</p> <ol style="list-style-type: none"> <li>1. <b>Memory Allocation:</b> OS allocates memory blocks to active processes. This can be done using various techniques like contiguous allocation or paging.</li> <li>2. <b>Memory Deallocation:</b> When a process completes, memory is released and marked as available for other processes to use.</li> <li>3. <b>Memory Protection:</b> The OS ensures that processes do not access memory areas they are not authorized to, preventing unauthorized data modification.</li> <li>4. <b>Virtual Memory:</b> The OS uses virtual memory techniques to use secondary storage (like hard disks) as an extension of main memory, allowing larger programs to run.</li> </ol> <p><b>b) File Management:</b> File management involves handling files on storage devices efficiently and providing an organized way for user and application data storage.</p>

	<ol style="list-style-type: none"> <li>1. <b>File Creation and Deletion:</b> OS enables users and programs to create new files or remove existing ones.</li> <li>2. <b>File Organization:</b> It manages how files are physically stored on storage devices, using techniques like contiguous allocation, linked allocation, or indexed allocation.</li> <li>3. <b>File Access Control:</b> It defines and enforces file access permissions to maintain data security and integrity.</li> <li>4. <b>File I/O Operations:</b> OS provides mechanisms for reading from and writing to files, ensuring proper synchronization and buffering.</li> </ol> <p><b>c) Process Management:</b> Process management involves controlling and coordinating the execution of multiple processes within an operating system.</p> <ol style="list-style-type: none"> <li>1. <b>Process Creation and Termination:</b> OS starts, pauses, resumes, and terminates processes as needed.</li> <li>2. <b>Process Scheduling:</b> It allocates CPU time to processes using scheduling algorithms like Round Robin, Priority Scheduling, or Multilevel Queue.</li> <li>3. <b>Process States:</b> Processes transition through different states (e.g., ready, running, waiting, terminated), managed by the OS scheduler.</li> <li>4. <b>Deadlock Handling:</b> OS identifies and resolves deadlocks where multiple processes are stuck waiting for each other to release resources.</li> <li>5. <b>Resource Allocation:</b> OS manages allocation of resources (CPU time, memory, I/O devices) to processes to ensure fair utilization.</li> </ol> <p><b>d) Secondary Storage Management:</b> Secondary storage management involves handling storage devices like hard disks for long-term data storage.</p> <ol style="list-style-type: none"> <li>1. <b>File System Creation:</b> OS creates and manages file systems on secondary storage, organizing data into files and directories.</li> <li>2. <b>Disk Space Allocation:</b> It allocates disk space for new files using techniques like contiguous, linked, or indexed allocation.</li> <li>3. <b>Disk Space Management:</b> OS tracks free and used space on disks, optimizing space utilization and avoiding fragmentation.</li> <li>4. <b>Caching:</b> The OS uses disk caching to store frequently accessed data in main memory for faster retrieval.</li> </ol>
17	<b>Write any four calls related to File Management</b>
A	<ol style="list-style-type: none"> <li>1. <b>open():</b> This system call is used to open a file and returns a file descriptor, which is a unique identifier for the opened file.</li> <li>2. <b>close():</b> The close() system call is used to close a file that was previously opened. It frees up resources and makes the file descriptor available for reuse.</li> <li>3. <b>read():</b> This system call is used to read data from an open file into a buffer in memory. It takes parameters like the file descriptor, buffer, and number of bytes to read.</li> <li>4. <b>write():</b> The write() system call is used to write data from a buffer in memory to an open file. It also takes parameters including the file descriptor, buffer, and number of bytes to write.</li> <li>5. <b>seek():</b> This system call is used to change the current position (offset) within a file. It's often used to move the file pointer to a specific location within the file.</li> <li>6. <b>unlink():</b> The unlink() system call is used to delete a file from the file system. It removes the file's directory entry and deallocates the associated disk space.</li> </ol>

**18** Define Process and draw the diagram of Process in Memory

**A** **Process:**  
 A process is defined as, "an entity which represents the basic unit of work to be implemented in the system". A process is a program in execution. Process is also called as job, task and unit of work.

A process includes:

- program counter
- stack
- data section

**Diagram:**

The diagram illustrates the memory layout of a process. It is a vertical rectangle divided into several sections. At the top, a grey box is labeled 'stack'. Below it is a large blue area. A downward-pointing arrow starts from the bottom of the 'stack' box and points into the blue area. An upward-pointing arrow starts from the bottom of the blue area and points to a grey box labeled 'heap'. Below the 'heap' box is another grey box labeled 'data', and at the very bottom is a grey box labeled 'text'. To the left of the entire stack, the word 'max' is at the top and '0' is at the bottom, indicating the memory address range.

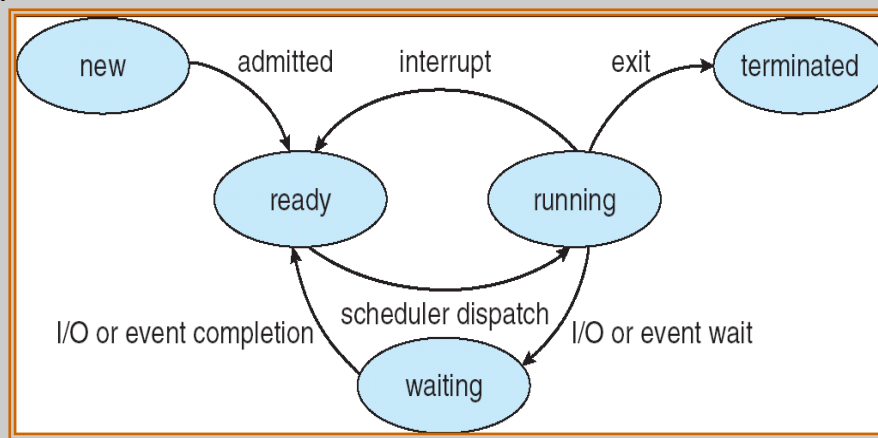
**19** Differentiate between Process and Program

A	Feature	Process	Program
	Definition	An instance of a computer program that is being executed	A set of instructions that are written in a programming language to perform a specific task

	<b>State</b>	Active, running, waiting, ready, or terminated	Passive, not executing
	<b>Resources</b>	Requires CPU, memory, I/O, and other resources	Does not require any resources
	<b>Control</b>	Has its own control block that stores information about its state, resources, and other data	Does not have a control block
	<b>Lifespan</b>	Temporary, created and terminated by the operating system	Permanent, exists until it is deleted
	<b>Example</b>	A web browser, a word processor, or a game	A compiler, an interpreter, or a library

20 . Draw and Explain Process State or Process Transition Diagram

A Diagram:



#### Explanation:

A process state is a condition of a process during its lifetime. The operating system manages the process states and transitions between them.

The following are the common process states:

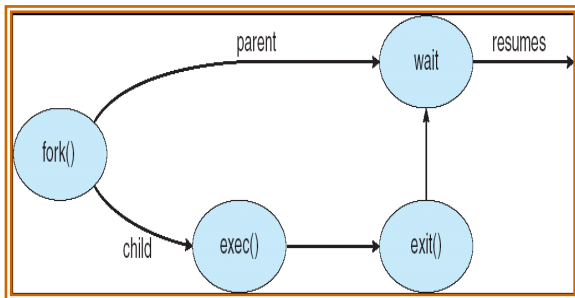
- **New:** A newly created process is in the new state. It is not yet ready to be executed.
- **Ready:** A process in the ready state is waiting to be assigned to a processor.
- **Running:** A process in the running state is currently being executed by the processor.
- **Waiting:** A process in the waiting state is waiting for an event to occur, such as an I/O operation to complete.
- **Terminated:** A process in the terminated state has finished its execution.

The process state transition diagram shows the possible transitions between the different process states. The arrows in the diagram represent the possible transitions between the states. For example, a process can move from the new state to the ready state when the operating system has allocated resources to it. A process can move from the running state to the waiting state when it needs to wait for an I/O operation to complete.

- **New to Ready:** A process moves from the new state to the ready state when the operating system has allocated resources to it and it is ready to be executed.
- **Ready to Running:** A process moves from the ready state to the running state when the operating system selects it to be executed.
- **Running to Waiting:** A process moves from the running state to the waiting state when it needs to wait for an event to occur, such as an I/O operation to complete.
- **Waiting to Ready:** A process moves from the waiting state to the ready state when the event it is waiting for occurs.

	<ul style="list-style-type: none"> <li>• <b>Running to Terminated:</b> A process moves from the running state to the terminated state when it finishes its execution.</li> </ul>
21.	<p>Write syntax of the following commands</p> <p>a. Sleep \$ sleep [OPTION] NUMBER[SUFFIX]</p> <p>b. Kill \$ kill [OPTIONS] PID</p> <p>c. Wait \$ wait [PID ....]</p> <p>d. Ps Ps [options] -u, -a, -e etc....</p>
22.	<p>Draw and explain PCB or TCB with proper diagram</p> <p>Each process is represented as a process control block (PCB) in the operating system. It contains information associated with specific process.</p> <p>Process State: It indicates current state of a process. Process state can be new, ready, running, waiting and terminated.</p> <p>Process number: Each process is associated with a unique number which is known process identification number.</p> <p>Program Counter: It indicates the address of the next instruction to be executed for the process.</p> <p>CPU Registers: The registers vary in number and type depending on the computer architecture. Register includes accumulators, index registers, stack pointers and general purpose registers plus any condition code information.</p> <p>Memory Management Information: It includes information such as value of base and limit registers, page tables, segment tables, depending on the memory system used by OS.</p> <p>Accounting Information: This information includes the amount of CPU used, time limits, account holders, job or process number and so on. It also includes information about listed I/O devices allocated to the process such as list of open files. Each PCB gives information about a particular process for which it is designed.</p>



	<table><tr><td>process state</td></tr><tr><td>process number</td></tr><tr><td>program counter</td></tr><tr><td>registers</td></tr><tr><td>memory limits</td></tr><tr><td>list of open files</td></tr><tr><td>...</td></tr></table>	process state	process number	program counter	registers	memory limits	list of open files	...
process state								
process number								
program counter								
registers								
memory limits								
list of open files								
...								
23.	<p>Explain process creation and process termination</p> <p>a. Process creation</p> <ul style="list-style-type: none"><li>When a new process is to be added to those currently being managed, the operating system builds the data structures that are used to manage the process and allocates address space in main memory to the process. This is the creation of a new process.</li><li>Parent process create children processes, which, in turn create other processes, forming a tree of processes</li></ul> <div></div> <p>b. Process termination</p> <ul style="list-style-type: none"><li>Depending upon the condition, a process may be terminated either normally or forcibly by some another process.</li><li>Normal termination occurs when the process completes its task (operation) and invokes an appropriate system call <code>ExitProcess( )</code> in Windows and <code>exit( )</code> in Unix to tell the operating system that it is finished.</li><li>A process may cause abnormal termination of some another process. For this, the process invokes an appropriate system call <code>TerminateProcess( )</code> in Windows and <code>kill( )</code> in Unix that tells the operating system to kill some other process.</li></ul>							
24.	<p>Give commands to perform following tasks</p> <ol style="list-style-type: none"><li>Add delay in a script <code>Sleep [options]SECONDS</code></li><li>To terminate the process <code>Kill pid</code></li></ol>							
25.	<p>Define</p> <p>a. Scheduler It decides which task should run and for how long , it helps manage the order in which program gets cpu time</p> <p>b. Context switch</p>							

	<p>Context switching is when the cpu switches from running one program to another program , saving and restoring necessary information so the programmers can continue where they left off</p> <p>c. IPC (inter process communication) It is a set of method of mechanisms that allow programs running on the same computer to exchange information or data with each other</p>
26.	<p>Explain basic IPC models</p> <p>a. Shared memory</p> <ul style="list-style-type: none"> <li>Two processes exchange data or information through sharing region. They can read and write data from and to this region.</li> <li>In this a region of the memory residing in an address space of a process creating a shared memory segment can be accessed by all processes who want to communicate with other processes</li> <li>All the processes using the shared memory segment should attach to the address space of the shared memory</li> <li>All the processes can exchange information by reading and/or writing data in shared memory segment. The form of data and location are determined by these processes who want to communicate with each other</li> <li>. These processes are not under the control of the operating system. The processes are also responsible for ensuring that they are not writing to the same location simultaneously</li> <li>After establishing shared memory segment, all accesses to the shared memory segment are treated as routine memory access and without assistance of kernel</li> </ul> <div data-bbox="328 1102 850 1796" data-label="Diagram"> <pre> graph TD     PA[Process A] -- 1 --&gt; SM[Shared memory]     PB[Process B] -- 2 --&gt; SM     subgraph MemoryStack [ ]         direction TB         PA         SM         PB         K[Kernel]     end </pre> </div> <p>b. Message passing</p> <ul style="list-style-type: none"> <li>In message passing model the data or information is exchanged in the form of messages.</li> </ul>

- It allows processes to communicate and synchronize their action without sharing the same address space. It is particularly useful in a distributed environment when communication process may reside on a different computer connected by a network
- Communication requires sending and receiving messages through the kernel. The processes that want to communicate with each other must have a communication link between them. Between each pair of processes exactly one communication link.

