Q1: Types of OS (notes)

Q2: Hardware dependent and independent OS (notes)

Q3: Kernel and Shell types (notes)

Q4: Types of services provided by OS?

Operating systems provide a wide range of services to facilitate the management and operation of computer hardware and software. Here are some of the key services typically provided by operating systems:

1. **Process Management:**
   - Creation and termination of processes.
   - Scheduling and allocation of CPU time to processes.
   - Interprocess communication and synchronization.
   - Management of process states (e.g., running, waiting, ready).
2. **Memory Management:**
   - Allocation and deallocation of memory to processes.
   - Virtual memory management, including paging and segmentation.
   - Memory protection to prevent unauthorized access.
3. **File System Management:**
   - Creation, deletion, and management of files and directories.
   - File access control and permissions.
   - File system integrity and reliability.
   - File caching and buffering for improved performance.
4. **Device Management:**
   - Control and coordination of peripheral devices such as disk drives, printers, and network interfaces.
   - Device drivers for hardware communication.
   - Input/output (I/O) operations handling and buffering.
   - Device status monitoring and error handling.
5. **User Interface:**
   - Graphical user interface (GUI) or command-line interface (CLI) for user interaction.
   - Management of input and output devices (e.g., keyboard, mouse, display).
   - User authentication and access control.
6. **Networking Services:**
   - Networking protocols implementation (e.g., TCP/IP stack).
   - Network configuration and administration.
   - Support for network applications and services (e.g., web servers, email clients).
7. **Security Services:**
   - User authentication and authorization.
   - Access control mechanisms (e.g., permissions, encryption).
   - Auditing and logging for security monitoring.

- Protection against malware and unauthorized access.

8. **System Administration and Configuration:**
    - Configuration of system settings and parameters.
    - Software installation, updates, and patch management.
    - System monitoring and performance optimization.
    - Backup and recovery services.
9. **Error Handling and Recovery:**
    - Detection and reporting of errors and exceptions.
    - Fault tolerance mechanisms to ensure system reliability.
    - Recovery procedures after system failures or crashes.

These services collectively enable the efficient and reliable operation of computer systems, providing a platform for running applications and managing resources effectively.

Q5: Multitasking, Multiprogramming, Multiprocessing

1) Multitasking:

Multitasking is a computer's ability to execute multiple tasks or processes concurrently, seemingly allowing multiple programs to run simultaneously. In a multitasking system, the CPU (Central Processing Unit) rapidly switches between different tasks, giving the illusion of parallel execution. This enables users to perform multiple activities simultaneously without having to wait for one task to finish before starting another.

Here's an example to illustrate multitasking:

Imagine you're using a modern computer with a multitasking operating system like Windows, macOS, or Linux. You have several applications open, such as a web browser, a word processor, and a music player. Each of these applications represents a separate task.

1. **Web Browsing:** You're reading an article in your web browser.
2. **Word Processing:** Simultaneously, you're writing a report in your word processor.
3. **Music Playing:** Additionally, you're listening to music in the background using a media player.

While you're working on your report in the word processor, the operating system is continuously switching between these tasks:

- **Time-sharing:** The CPU allocates a small slice of time to each task. For example, it might allocate a few milliseconds to the web browser, then switch to the word processor, and then to the media player.
- **Context Switching:** When the CPU switches between tasks, it saves the current state of the task (such as its registers and memory) and loads the state of the next task to be executed. This process is known as a context switch.

From the user's perspective, it appears that all three tasks are running simultaneously, even though the CPU is actually rapidly switching between them. This allows you to browse the web, write your report, and listen to music without noticeable delays or interruptions.

Multitasking greatly enhances productivity by allowing users to perform multiple tasks concurrently, making efficient use of system resources and reducing idle time.

2) Multiprogramming:

Multiprogramming is a technique used by operating systems to maximize CPU utilization by allowing multiple programs to reside in memory simultaneously. It involves loading several programs into main memory and executing them concurrently, switching between them as necessary.

Here's how multiprogramming works:

1. **Loading Programs:** The operating system loads multiple programs into memory, even though only one program can execute at a time on a single CPU core.
2. **CPU Scheduling:** The CPU scheduler selects one program to execute and allocates CPU time to it. When the program needs to wait for I/O (Input/Output) operations or encounters a blocking condition, such as waiting for user input, the CPU scheduler switches to another ready program.
3. **Context Switching:** When the CPU switches between programs, it saves the state of the currently running program (context) and loads the state of the next program to be executed. This process is known as a context switch.
4. **Concurrent Execution:** While a program is waiting for I/O or other resources, the CPU can execute another program, thereby keeping the CPU busy and maximizing throughput.

Multiprogramming offers several benefits:

- **Improved CPU Utilization:** By allowing multiple programs to execute concurrently, multiprogramming ensures that the CPU is utilized efficiently, reducing idle time.
- **Faster Response Time:** Users perceive faster response times because the CPU can switch to another program while one program is waiting for I/O or other resources.
- **Increased Throughput:** With multiple programs running simultaneously, the system can handle more tasks and process them concurrently, leading to increased throughput and better system performance.

3) Multiprocessing:

Multiprocessing is a computing technique that involves the simultaneous execution of multiple processes or threads across multiple CPU cores or processors within a single computer system. Unlike multitasking, where multiple tasks are managed by a single CPU through time-sharing,

multiprocessing involves the parallel execution of tasks across multiple CPUs or CPU cores, enabling true simultaneous processing.

Here's how multiprocessing works:

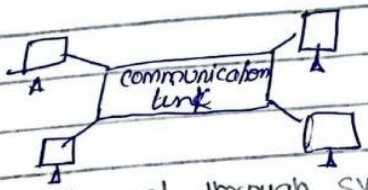
1. **Multiple Processors or Cores:** A multiprocessing system contains multiple physical processors (CPUs) or multiple cores within a single processor chip.
2. **Concurrent Execution:** Each processor or core is capable of executing its own set of instructions independently of the others. This allows multiple processes or threads to be executed simultaneously, with each processor/core handling its allocated tasks concurrently.
3. **Parallel Processing:** Tasks are distributed among the available processors/cores, and each processor/core works on its assigned tasks concurrently. This parallel processing capability significantly increases computational speed and throughput compared to single-core/single-processor systems.
4. **Communication and Coordination:** In multiprocessing systems, processes or threads may need to communicate and synchronize with each other, especially when sharing resources or working on interdependent tasks. Proper synchronization mechanisms, such as locks, semaphores, and message passing, are used to ensure data integrity and prevent race conditions.

Multiprocessing offers several advantages:

- **Increased Performance:** By distributing tasks across multiple processors/cores, multiprocessing systems can handle a higher workload and achieve faster processing speeds compared to single-core/single-processor systems.
- **Improved Scalability:** Multiprocessing systems can scale performance by adding more processors/cores, allowing them to efficiently handle increasingly demanding workloads without sacrificing performance.
- **Enhanced Reliability:** Redundancy can be built into multiprocessing systems, with multiple processors/cores capable of taking over tasks in case of hardware failures or errors, thus increasing system reliability and fault tolerance.

Q6: Distributed system and types?

Distributed System:



→ Fault tolerance available
→ Scalibility holds
→ Message passing (Kernel through system call
→ Concurrency (multi task execution under one processor)
→ Resource sharing

⇒ contain multiple nodes that are seprated but connected through a communication link to each orther

Example:-
Internet, Intranet, mobile computing (smartphones, portable devices)

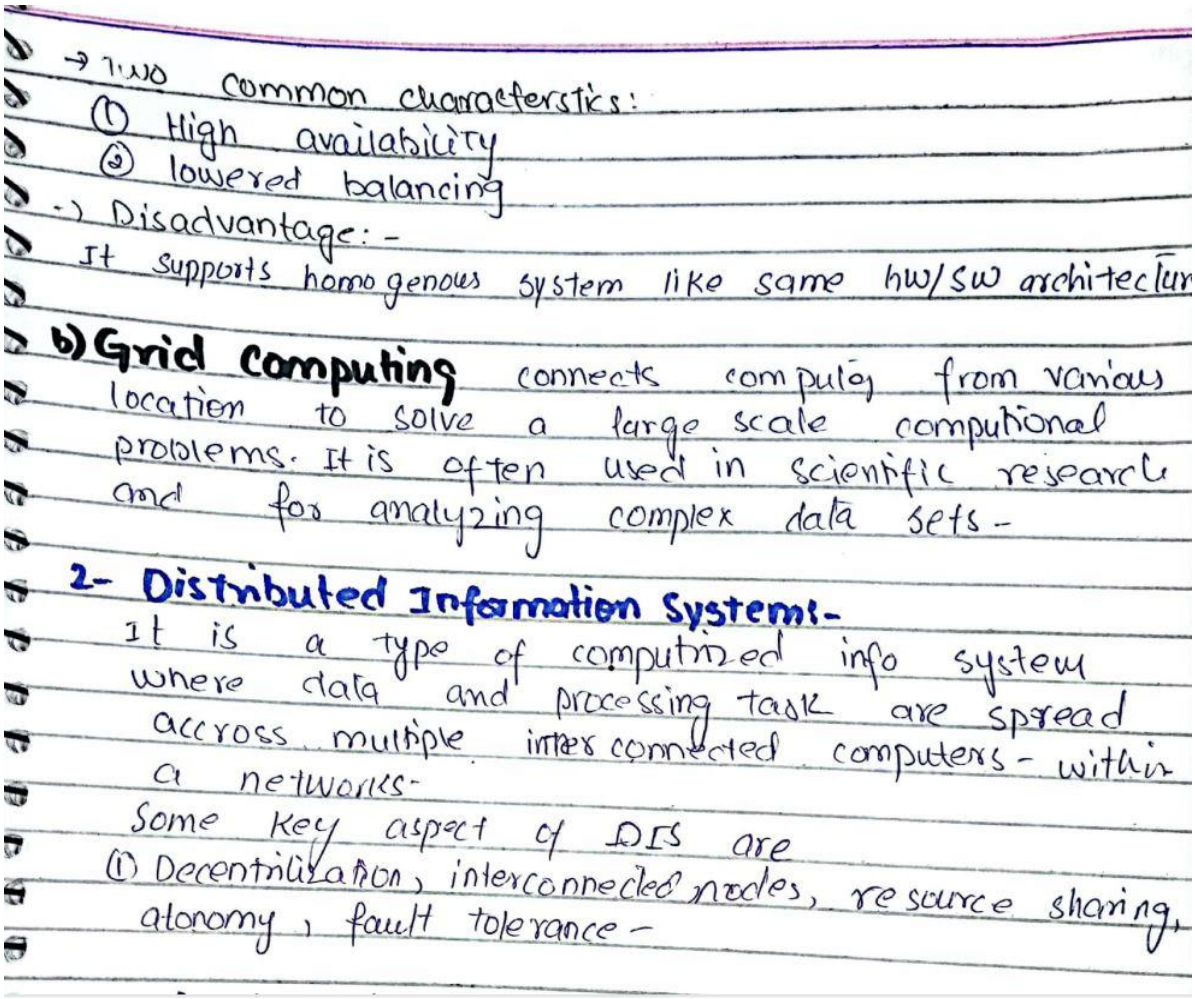
Types:- (3 major types)

1- Distributed computing system

→ This distribution system uses group of computers/system that share common computing problem so a to generate in a short time.

a) Cluster:-

→ involves connecting multiple computers called nodes together to work as a single system
→ the nodes are typically connected through a high speed network and share resources such as memory, storage & processing power



3-Distributed pervasive system

In a distributed pervasive system, computing resources, sensors, actuators, and communication devices are distributed across a network of interconnected nodes, which may include mobile devices, embedded systems, wearable devices, smart appliances, and other Internet of Things (IoT) devices. These devices collaborate and communicate with each other to gather data, process information, and provide services in a pervasive manner, seamlessly integrating computing capabilities into various aspects of daily life.

Q7: process states, PCB, independent and cooperating processes (slide no.3)

process states

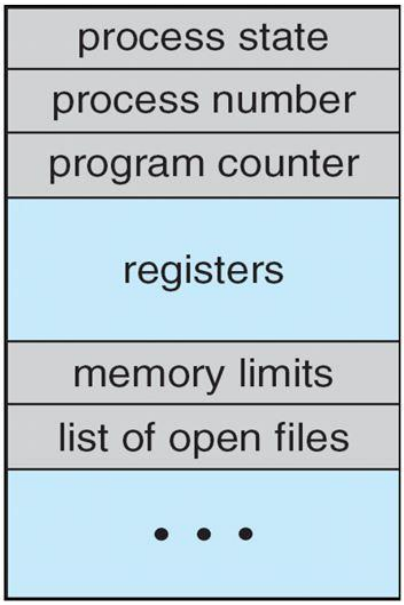
- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Process Control Block (PCB)

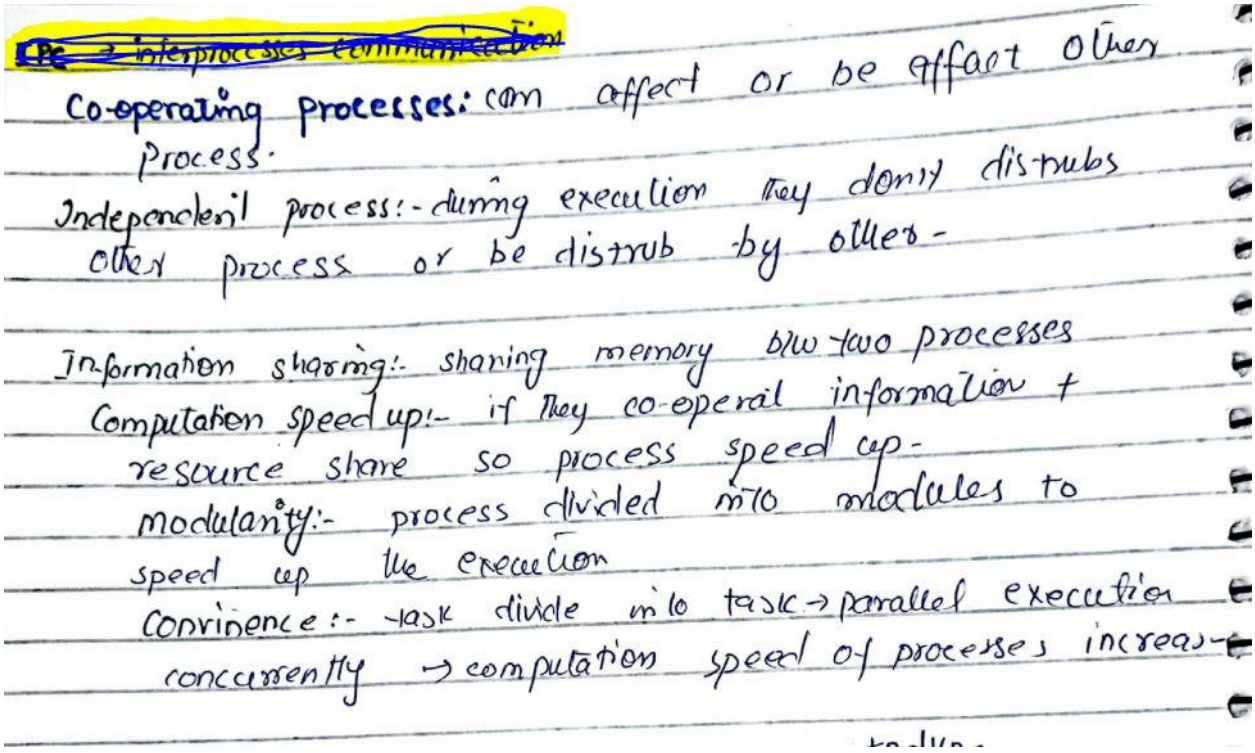
Information associated with each process

(also called task control block)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files



Independent and cooperating processes

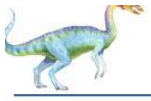


Q8: Scheduling Queues? (slide no. 03)

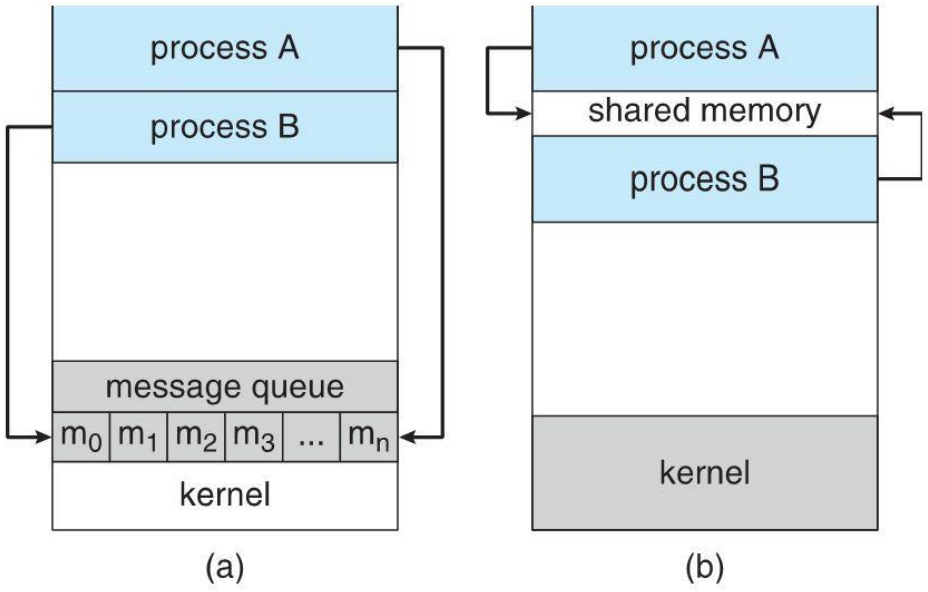
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues

Q9: Inter process communication (slide no. 3)

- Processes within a system may be ***independent*** or ***cooperating***
- Cooperating process can affect or be affected by other processes, including sharing data
- Reasons for cooperating processes:
  - Information sharing
  - Computation speedup
  - Modularity
  - Convenience
- Cooperating processes need **interprocess communication** (**IPC**)
- Two models of IPC
  - **Shared memory**
  - **Message passing**



Communications Models



Q10: CPU bound processes and I/O bound processes? (types of processes?)

CPU bound processes

CPU-bound processes are tasks or programs that primarily require computational resources (CPU time) to execute, rather than relying heavily on I/O (Input/Output) operations or other external factors. These processes typically spend most of their execution time performing calculations, data processing, or other CPU-intensive operations.

Characteristics of CPU-bound processes include:

1. **High CPU Utilization:** CPU-bound processes consume a significant amount of CPU time during their execution, often utilizing a large portion of the available CPU resources.
2. **Limited I/O Operations:** CPU-bound processes may involve minimal or occasional I/O operations, such as reading input data or writing output results, but their main focus is on computational tasks.
3. **Shorter Waiting Time:** Since CPU-bound processes primarily require CPU time to complete their tasks, they tend to spend less time waiting for I/O operations to complete compared to I/O-bound processes.

Examples of CPU-bound processes include:

- Mathematical calculations: Programs performing complex mathematical computations, simulations, or numerical analysis tasks.
- Image or video processing: Software applications manipulating images, videos, or graphics that involve intensive processing of pixel data.

I/O bound processes

I/O bound processes are tasks or programs that rely heavily on Input/Output (I/O) operations, such as reading from or writing to storage devices, network communication, or user interactions. Unlike CPU-bound processes, which primarily require computational resources for execution, I/O-bound processes spend a significant portion of their execution time waiting for I/O operations to complete.

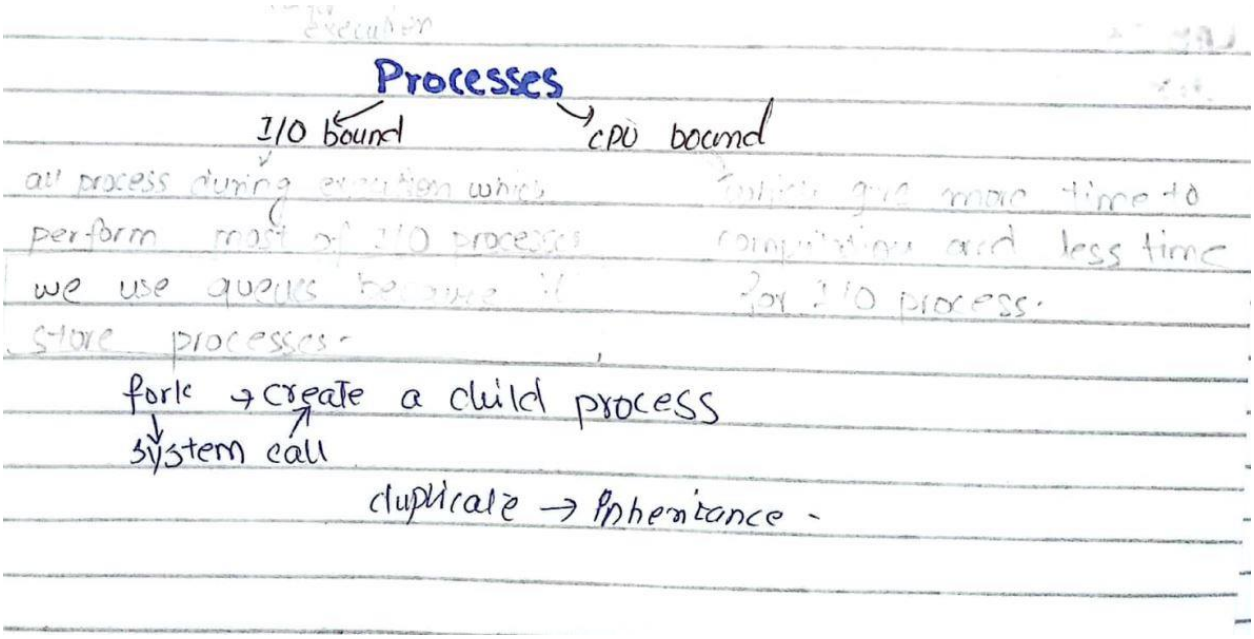
Characteristics of I/O bound processes include:

1. **High I/O Wait Time:** I/O-bound processes spend a substantial amount of time waiting for I/O operations to finish, during which the CPU remains idle.
2. **Frequent I/O Operations:** These processes involve frequent interactions with I/O devices, such as reading data from disk, writing data to disk, sending or receiving network packets, or waiting for user input.
3. **Lower CPU Utilization:** While I/O operations are pending, the CPU is often underutilized, as the process is waiting for external events or resources to become available.

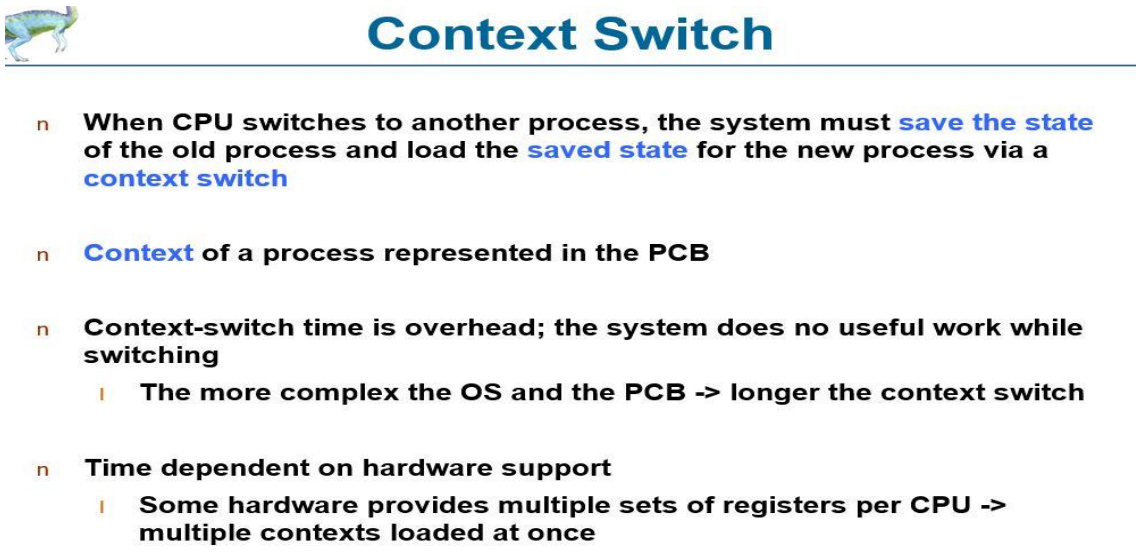
4. **Longer Execution Times:** I/O-bound processes may have longer execution times compared to CPU-bound processes, primarily due to the time spent waiting for I/O operations to complete.

Examples of I/O-bound processes include:

- File I/O operations: Programs that read or write large amounts of data from/to files stored on disk, such as database servers, file servers, or backup utilities.
- Network communication: Applications that send or receive data over a network, such as web servers, email servers, or file transfer protocols (FTP).



Q11: Context Switching? (slide no. 03)



1. **Save State:** Just like the TV saves where you left off, the computer saves all the important stuff about what it was doing with the current task—like which part of the program it was running and what data it was working with.
2. **Load State:** Then, it loads up all the details of the new task you want to do—like which program you want to run and where you left off in that program.
3. **Switch:** Finally, it starts running the new task, just like your TV switches to the new channel.

So, context switching is like the computer's way of quickly jumping between different tasks or programs, making it seem like they're all running at the same time, even though the computer's processor can only do one thing at a time.

Q 12: Scheduling Criteria? (notes) (example)

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

28th March, 2024 Tuesday

Scheduling Criteria

OS ko examine karna ka liya jo parameters use karta hain process criatria.

→ Kis ko kia cheez dani ha ya scheeling criteria kahta hain

Scheduling Criteria

(i) CPU utilization } → maximize
(ii) Throughput / Bandwidth
(iii) Turnaround time } → Minimize
(iv) Waiting time
(v) Response time.

Throughput / Bandwidth.

Formula.

Time ⇒ Data size / throughput

= 4000 / 100

= 40 s

Q = 100 mbps. → throughput / Bandwidth
500 MB → Data size / file size

500 MB

= 500 x 8 = 4000 mb

	P_1	P_2	P_3
→ Turnaround time			
→ Arrival time	0	2	3
→ Burst time	5	4	6
→ execution time	5	9	13

Formula of turnaround time.

Turnaround = Execution time − Arrival time

$P_1(t)$ = 5 − 0 => 5 → efficient

$P_2(t)$ = 9 − 2 => 7

$P_3(t)$ = 13 − 3 => 10

→ Waiting time.

Ready queue ma wait krna sb se kam execute hna wale ka time waiting time hota ha.

	P_1	P_2	P_3
→ Arrival time	0	2	3
→ Burst time	4	5	6
→ Turnaround time	7	9	12

Waiting time = Turnaround Time − Burst time

P_1 = 7 − 4 = 3

P_2 = 9 − 5 = 4

P_3 = 12 − 6 = 6

→ Response time.

execution wa lab 1st time enter hota ha or usa 1st response milna ma jo time lagta ha usa response time kahta