🧠 Assignment 1: Linux Commands & Shell Programming

1. **What is a shell?**
   → A shell is a command-line interface between the user and the kernel. It interprets user commands and executes them in the operating system.
2. **What is a shell script?**
   → A shell script is a text file containing a sequence of shell commands that are executed together like a program to automate tasks.
3. **What is exit status in Linux?**
   → Every command returns an exit code: 0 indicates successful execution, while any non-zero value indicates an error or failure.
4. **What does $? represent?**
   → It stores the exit status of the last executed command and is useful for checking whether the previous command succeeded.
5. **Differentiate user-defined and system variables.**
   → System variables (like $HOME, $PATH) are predefined by Linux, whereas user-defined variables are created by users for their own scripts.
6. **Explain the purpose of the test command.**
   → test (or [ ]) checks conditions such as file existence, numeric or string comparisons, and returns true or false status.
7. **What loops are supported in shell scripting?**
   → Shell supports for, while, and until loops for repeated execution of commands.
8. **What does the grep command do?**
   → It searches for specific patterns or strings within files and displays matching lines.
9. **What is the function of sed?**
   → sed (stream editor) performs text editing tasks like search, replace, insertion, or deletion without opening the file.
10. **What does the man command do?**
    → It displays the manual page for any Linux command, explaining its syntax, options, and usage examples.

⚙️ Assignment 2: Process Control System Calls

1. **Define a process.**
   → A process is an executing instance of a program, having its own address space, resources, and process ID (PID).
2. **What is fork() used for?**
   → fork() creates a new process called a child, which runs concurrently with the parent process.
3. **What does fork() return?**
   → It returns 0 to the child, the child's PID to the parent, and –1 on error.
4. **Explain the use of exec() family of calls.**
   → These replace the current process image with a new program, allowing one process to run another executable.
5. **What is the purpose of wait()?**
   → It makes the parent process pause until its child finishes, ensuring proper synchronization and cleanup.
6. **What is a zombie process?**
   → A process that has terminated but whose parent hasn't collected its exit status yet; it remains in the process table temporarily.
7. **What is an orphan process?**
   → A child process whose parent has terminated; it is automatically adopted by the init process.
8. **What does getpid() return?**
   → The unique process ID of the calling process.
9. **What does getppid() return?**
   → The PID of the parent process.
10. **What is system() function used for?**
    → It executes a shell command directly from a C program, similar to typing it in a terminal.

🧩 Assignment 3: CPU Scheduling (SJF & Round Robin)

1. **Define burst time.**
   → The CPU time required by a process to complete execution once it gets the CPU.

2. **What is waiting time?**
   → The total time a process spends in the ready queue waiting to be executed.
3. **What is turnaround time?**
   → The total time from process submission to its completion, i.e., waiting + burst time.
4. **Explain Shortest Job First (SJF).**
   → In SJF, the process with the smallest burst time executes first, giving minimal average waiting time.
5. **Differentiate preemptive and non-preemptive scheduling.**
   → In preemptive, a process can be interrupted; in non-preemptive, a process runs until completion.
6. **What is Round Robin scheduling?**
   → Each process gets a fixed time quantum to run in cyclic order; suitable for time-sharing systems.
7. **What is time quantum or time slice?**
   → The small fixed CPU time allotted to each process in Round Robin scheduling.
8. **Which algorithm gives minimum average waiting time?**
   → Shortest Job First (SJF).
9. **List CPU scheduling criteria.**
   → CPU utilization, throughput, turnaround time, waiting time, and response time.
10. **What is context switching?**
    → The act of saving the current process state and loading another process's state when CPU changes.

📝 Assignment 4A: Thread Synchronization (Producer–Consumer)

1. **What is a semaphore?**
   → A semaphore is a synchronization tool used to control access to shared resources by multiple processes.
2. **Types of semaphores?**
   → Binary (0/1) for mutual exclusion and Counting for controlling multiple resource instances.

3. **Explain wait() and signal() operations.**
   → wait() decrements the semaphore and may block if zero; signal() increments it and unblocks waiting processes.
4. **What is a mutex?**
   → A mutual-exclusion lock that allows only one thread to access a shared resource at a time.
5. **What is the critical section problem?**
   → Ensuring that only one process executes a critical section (shared data access) at any given time.
6. **Explain the producer-consumer problem.**
   → It involves two processes sharing a buffer — the producer adds items, and the consumer removes them using semaphores for synchronization.
7. **What happens if semaphore value < 0?**
   → Processes performing wait() are blocked until another process performs signal().
8. **What is the initial value of a mutex?**
   → Usually 1, representing an unlocked state.
9. **List common pthread mutex functions.**
   → pthread_mutex_init, pthread_mutex_lock, pthread_mutex_unlock, pthread_mutex_destroy.
10. **What is deadlock?**
    → A situation where two or more processes wait indefinitely for resources held by each other.

📘 Assignment 4B: Reader–Writer Problem

1. **What is the Reader–Writer problem?**
   → It deals with synchronization between processes that read and write a shared data area, ensuring consistency.
2. **Can multiple readers read simultaneously?**
   → Yes, multiple readers can read at the same time since they don't modify data.
3. **Can a writer write while readers are reading?**
   → No, writers must wait until all readers finish to avoid inconsistent data.

4. **What is writer starvation?**
   → When continuous reader activity prevents writers from ever accessing the shared resource.
5. **Which synchronization tools are used?**
   → Semaphores and mutexes are used to coordinate reader and writer access.
6. **Which semaphore ensures mutual exclusion for writers?**
   → The wsem semaphore ensures that only one writer can write at a time.
7. **What does reader priority mean?**
   → Readers are given access first if other readers are already reading, delaying writers.
8. **What is the role of the readcount variable?**
   → It tracks how many readers are currently reading; first reader locks the resource, last reader releases it.
9. **Which header file provides thread synchronization functions?**
   → pthread.h.
10. **What is the initial value of wsem?**
    → It is initialized to 1, meaning the shared data is initially free for access.

🏦 Assignment 5: Deadlock Avoidance – Banker's Algorithm

1. **What is a deadlock?**
   → A situation where two or more processes wait indefinitely for resources held by each other.
2. **What is the Banker's algorithm used for?**
   → It is used to avoid deadlock by checking whether a system is in a safe or unsafe state before allocating resources.
3. **What is a safe state?**
   → A state where all processes can complete their execution without leading to a deadlock.
4. **What is an unsafe state?**
   → A state where processes may not complete, potentially leading to deadlock.
5. **What are the matrices used in the Banker's algorithm?**
   → Allocation, Max, Need, and Available matrices.

6. **How is the Need matrix calculated?**
   → Need[i][j] = Max[i][j] - Allocation[i][j]
7. **What is the condition for resource allocation in Banker's algorithm?**
   → Request ≤ Need and Request ≤ Available.
8. **What does the algorithm do after a resource request?**
   → Temporarily allocates resources, checks for a safe sequence, and grants only if safe.
9. **Why is it called the Banker's algorithm?**
   → Because it models how a banker safely allocates loans to customers ensuring all can be repaid.
10. **What happens if the system is unsafe?**
    → The resource request is denied to avoid potential deadlock.

💾 Assignment 6: Page Replacement Algorithms (FCFS, LRU, Optimal)

1. **What is page replacement?**
   → It's the process of swapping pages between main memory and secondary storage when memory is full.
2. **What causes a page fault?**
   → When a process tries to access a page not currently in memory.
3. **Explain FCFS page replacement.**
   → The oldest loaded page is replaced first, regardless of how frequently it's used.
4. **Explain LRU (Least Recently Used).**
   → The page that has not been used for the longest time is replaced.
5. **Explain the Optimal page replacement algorithm.**
   → Replaces the page that will not be used for the longest time in the future.
6. **Which page replacement algorithm gives the least number of page faults?**
   → The Optimal algorithm.
7. **What is Belady's anomaly?**
   → In some cases, increasing frame size increases page faults (seen in FCFS).

8. **How does LRU approximate the Optimal algorithm?**
   → By using past behavior (recent use) to predict future use.
9. **What is a reference string?**
   → A sequence of memory page numbers representing the order of page requests.
10. **Why is the Optimal algorithm not practical?**
    → Because future page references cannot be predicted in real systems.

🔁 **Assignment 7: Inter-Process Communication (FIFO & Shared Memory)**

(A) FIFO Communication

1. **What is IPC?**
   → Inter-Process Communication allows processes to exchange data or signals.
2. **What is FIFO?**
   → A named pipe that allows communication between unrelated processes in a first-in, first-out manner.
3. **How does FIFO differ from an unnamed pipe?**
   → FIFO exists as a special file in the filesystem, unlike unnamed pipes which exist only during process execution.
4. **Which system call is used to create a FIFO?**
   → mkfifo()
5. **Can FIFO provide full-duplex communication?**
   → Yes, two FIFOs can be used for full-duplex (two-way) communication.

(B) Shared Memory Communication

6. **What is shared memory?**
   → A memory segment accessible by multiple processes for high-speed data exchange.
7. **Which system calls are used for shared memory?**
   → shmget(), shmat(), shmdt(), and shmctl().

8. **What is the advantage of shared memory IPC?**
   → It is the fastest form of IPC because processes directly access the same memory.
9. **What synchronization mechanism is used with shared memory?**
   → Semaphores or mutex locks to prevent concurrent access issues.
10. **What happens if synchronization is not maintained?**
    → Data inconsistency or race conditions can occur.

Assignment 8: Disk Scheduling Algorithms (SSTF, SCAN, C-LOOK)

1. **What is disk scheduling?**
   → It determines the order in which I/O requests to the disk are serviced to improve performance.
2. **What is SSTF (Shortest Seek Time First)?**
   → The disk head moves to the request closest to its current position.
3. **What is SCAN (Elevator) algorithm?**
   → The disk arm moves in one direction, servicing requests, then reverses direction like an elevator.
4. **What is C-LOOK?**
   → Similar to SCAN, but the head moves in one direction and jumps back to the start without servicing on return.
5. **Which disk scheduling algorithm gives minimum average seek time?**
   → SSTF usually gives better average seek time than FCFS.
6. **What is seek time?**
   → The time taken by the disk arm to move to the track containing the desired sector.
7. **What is rotational latency?**
   → The delay waiting for the desired disk sector to rotate under the read-write head.
8. **Which algorithm prevents starvation?**
   → SCAN and C-LOOK reduce starvation compared to SSTF.
9. **What does “initial head position” mean?**
   → The starting track number of the disk arm before servicing begins.

10. **What is the main goal of disk scheduling?**
    → To minimize seek time and improve throughput of I/O operations.

🧩 Assignment 9: Study Assignment – System Call Implementation

1. **What is a system call?**
   → A request from user space to the kernel to perform a privileged operation, like I/O or process management.
2. **How are system calls implemented?**
   → By adding a function in kernel source, registering it in the system call table, recompiling the kernel, and calling it from user space.
3. **Example of a system call in Linux?**
   → fork(), read(), write(), open(), close(), exec().
4. **What is the difference between user mode and kernel mode?**
   → User mode restricts direct hardware access, while kernel mode has full control over hardware and memory.
5. **Why are system calls needed?**
   → They provide controlled access to critical kernel resources securely.
6. **What happens when a system call is invoked?**
   → CPU switches from user mode to kernel mode to execute the kernel-defined routine.
7. **How can you identify system calls in C code?**
   → By using commands like strace to trace system calls made by a program.
8. **What is kernel compilation?**
   → The process of building a custom Linux kernel after modifying its source code.
9. **What are examples of system call categories?**
   → Process control, file management, device management, and inter-process communication.
10. **What is the purpose of adding a custom system call?**
    → To extend kernel functionality or add custom features for experimentation or research.