Assignment - 06
CS341: Operating System Lab

---

1. Write a C program that implements a simple, signal-safe logging system to log messages when certain signals are received (e.g., SIGUSR1 and SIGUSR2). The logging function should be designed to handle signal interruptions, as functions like printf() and fprintf() are not safe to call from within a signal handler. Instead, use the low-level system call write() to safely log messages to a log file. Your program should log "SIGUSR1 received!" and "SIGUSR2 received!" when the corresponding signals are caught, and it should avoid deadlock or undefined behavior even when signals are received rapidly or during a critical section.

2. Signal-based Parallel Sorting System: Design a sorting system where multiple processes are responsible for sorting different parts of an array in parallel. The array will be divided into sub-arrays, and each process will handle sorting its assigned part. The processes will be triggered to start sorting by receiving specific signals, such as SIGUSR1 for one process and SIGUSR2 for another. After each process completes sorting its sub-array, it will notify the parent process using a signal. Once all processes have finished, the parent process will merge the sorted sub-arrays to produce the final sorted array. The system must ensure that the sorting operations happen in parallel and the results are combined correctly. Implement detailed step-by-step output to show the progress of sorting and merging operations.

3. Signal-Based Inter-Process Communication and Task Synchronization: In this problem, you will design a system that demonstrates inter-process communication and synchronization using signals. The system consists of two separate programs (processA.c and processB.c), where each process performs specific tasks in a coordinated manner. The processes will communicate using signals (SIGUSR1 and SIGUSR2) to synchronize their tasks. Process A will begin by generating a random number and storing it in shared memory (Task 1), after which it sends a signal to Process B to start its Task 1. Process B reads the number from shared memory, adds 10 to it, and updates the memory. Once Process B finishes, it signals Process A to begin Task 2, where it multiplies the updated number by 2 and stores the result. The final result is read and printed by Process B. You must ensure proper signal handling, synchronization, and shared memory usage to avoid race conditions and ensure orderly task execution between the processes.

4. Design a signal-based file downloader that can be paused and resumed using user signals. The program simulates the download of a large file in chunks. When the SIGUSR1 signal is received, the download should pause, and it should not continue until the SIGUSR2 signal is received to resume. The download's progress should be saved during pauses so that it can continue from the last saved point when resumed. Additionally, the program must handle the SIGTERM signal for graceful termination, saving the current progress to allow resumption if the download is restarted. The output should display progress during download, the status when paused or resumed, and the final status when the download completes or the program is terminated.
