OS Assignment - 3
Report

Submitted by Group ID Number - **62**

1. **Akshit Gupta 2022058**
2. **Manit Kaushik 2022277**

Contributions -

Manit Kaushik	Akshit Gupta
• Developed core control flow in the program by implementing the **main** loop where simple shell is implemented. • Implemented a mechanism for managing the **process control block (pcb)**. This included adding processes to the pcb, a fundamental component for process management within the program. • Structured program to appropriately handle signals. • Created a shared memory and used it for process queue and binary semaphore. • Implemented the **scheduler_handler()**	• Created assignment report which provides us with the assignment's context as well as insights into the implemented functionalities. • Created global variables, structures for process management and initialized pcb at the beginning of the code. • Incorporated error checks at various stages to ensure that the program gracefully handles unexpected scenarios, provides error messages, and takes appropriate actions accordingly. • Processed user input and enabled users to view the history of the SimpleShell & SimpleScheduler.

SimpleScheduler Implementation -

The SimpleScheduler is a program that manages the execution of processes based on their priority and a specified time quantum. It uses shared memory and a binary semaphore for inter-process communication. The scheduler initiates execution upon receiving a designated signal and iterates through a process queue, efficiently executing processes according to their priority and the defined time slice. This allows

for effective utilization of system resources and controlled process management.

- ## Global Variables:

  This SimpleScheduler implementation uses global variables such as **pid_list, start_time_list, total_time_list, no_of_commands, and cmd_list,** which are essential for storing information related to the processes and commands executed within the program.

  These variables serve as a central repository for process data, facilitating easy access and management throughout the program.

- ## Structures for Process Management:

  The struct **process_node** structure represents a process with details like file name, process ID, execution status, priority, start and end times.

  The struct **process_queue** structure represents a queue of processes using a linked list.

- ## Process Control Block (PCB) Initialization:

  pcb_initialize function initializes a process queue (PCB) and returns a pointer to the initialized queue.

  A process queue is created within the PCB, enabling the addition and retrieval of processes in a structured manner.

- ## Adding Processes to PCB:

  add_to_pcb function adds a process (specified by a file name and priority) to the process queue (PCB).

  It creates a new process node and populates it with pertinent process details such as the file name, process ID, execution status, and priority. This function is critical for building and maintaining the process queue within the PCB.

- ## Signal Handling:

  The **scheduler_handle**r function is a signal handler that handles the **SIGUSR1** signal, which is used to start the scheduler.

  Upon receiving this signal, the program initiates the scheduler, demonstrating the role of signal handling in coordinating program flow.

- ## Scheduler Logic:

  The scheduler logic is initiated upon receiving the SIGUSR1 signal. It iterates through the process queue and executes processes based on their priority and time quantum (**TSLICE**).

- ## Shared Memory and Semaphore:

  Shared memory and a binary semaphore are used for inter-process communication. The shared memory stores the process queue and a binary semaphore (mutex) is used to synchronize access to the shared memory.

- ## User Input Processing:

  The main loop of SimpleScheduler continuously reads user input, processing commands based on the specified instructions.

  User input is critical for initiating the scheduler, adding processes to the scheduler's queue, or exiting the program.

  This user interaction mechanism is the primary means of controlling and directing the program's behavior.

- ## Printing History:

  At the conclusion of program execution, it is important to provide users with a comprehensive view of the commands executed and their corresponding details.

  The **"Complete History"** section of the program displays a summarized history, including process IDs, start times, total execution times, and the commands themselves.

  This history section offers users insights into the program's behavior and provides a record of executed commands.

Private Github Repository Link -

https://github.com/ManitK/CSE231-Assignments