OS Assignment of Group No. 44(Section-A) , Prof Vivek Kumar

SimpleScheduler Assignment Documentation

Introduction

This documentation provides an overview of the SimpleScheduler and Simple-Shell assignment, two related C programs that demonstrate process scheduling and interaction through a command-line shell. The SimpleScheduler program serves as the core scheduler, while Simple-Shell is a user interface for submitting and monitoring jobs. This assignment is designed to help students understand the basics of process scheduling and shell interactions.

Simple-Shell (simple-shell.c)

Program Overview

Simple-Shell is a basic command-line shell that allows users to submit processes, view the process execution history, and exit the shell. It is designed to interact with the SimpleScheduler for process scheduling.

Key Features

- Submit a job with an optional priority.
- View the execution history of submitted jobs.
- Gracefully exit the shell.

Implementation Details

1. **Command History**: The shell maintains a command history using a data structure called `CommandExecution`. It stores information about the executed command, its process ID (PID), priority, start time, end time, and wait time.
2. **Submit a Job**: Users can submit jobs using the "submit" command, specifying an executable name and an optional priority.
3. **Execution Timing**: The shell records the start and end times of executed processes. Additionally, it calculates the wait time for each job, reflecting the time it spent waiting in the queue before execution.
4. **Interactive Interface**: Users can interact with the shell by entering commands in the command-line interface.

Example Usage

SimpleShell$ submit ./a.out

Enter a non-negative integer: 5
Factorial of 5 is 120
SimpleShell$ history
Job History:
Name: ./a.out
PID: 2790
Execution Time: 3673643 microseconds
Wait Time: 1697479512020656 microseconds
Priority: 1
SimpleShell$ exit

SimpleScheduler (SimpleScheduler.c)

Program Overview

SimpleScheduler is a basic process scheduler responsible for managing a queue of processes and allocating CPU time to them in a round-robin fashion. It operates based on the number of CPU resources (NCPU) and a time quantum (TSLICE).

Key Features

- Initialize the scheduler with the number of CPU resources and the time quantum.
- Add processes to the ready queue.
- Start and manage the scheduling loop, distributing CPU time to processes.

Implementation Details

1. **Scheduling Algorithm**: SimpleScheduler employs a round-robin scheduling algorithm, ensuring that each process receives a fixed time quantum of CPU time before moving to the next process.
2. **Ready Queue**: The ready queue, represented by the `ready_queue` array, holds processes awaiting execution. Processes are ordered based on priority.
3. **Signal Handling**: The scheduler handles the SIGALRM signal to implement the time-slicing mechanism. When a time slice expires, the current running process is paused, and control is transferred to the next process in the queue.
4. **Scheduling Loop**: The scheduler's main function orchestrates the scheduling loop, controlling the execution and timing of processes.
5. **Execution Time and Wait Time**: SimpleScheduler tracks each process's execution time and wait time, reflecting the total time the process has been executed and the time spent waiting in the queue.

Example Usage

$ ./SimpleScheduler 2 1000

Usage

To execute the SimpleScheduler, run it from the command line with two arguments: the number of CPU resources (NCPU) and the time quantum (TSLICE).

./SimpleScheduler <NCPU> <TSLICE>

Conclusion

The SimpleScheduler and Simple-Shell assignment provides a practical introduction to process scheduling and interaction through a simple command-line shell. Students can explore fundamental concepts of job submission, prioritization, and execution tracking, gaining a foundational understanding of process management in operating systems.

Contributions

In the SimpleScheduler and Simple-Shell assignment, Athiyo Chakma and Yash Goyal made equal contributions to various aspects of the project. Here are their contributions:

Athiyo Chakma's Contributions

1. **Simple-Shell Implementation**: Athiyo played a significant role in developing the Simple-Shell program. They designed and implemented the user interface, allowing users to submit jobs, view job history, and gracefully exit the shell.
2. **Command History**: Athiyo was responsible for creating the `CommandExecution` data structure and implemented the logic for recording and managing the execution history of submitted jobs within the Simple-Shell.
3. **Execution Timing**: Athiyo worked on recording and calculating the execution time and wait time for each executed job. This includes capturing start and end times and accurately calculating wait times for processes in the queue.
4. **Interactive Interface**: Athiyo ensured that the Simple-Shell provided a user-friendly and interactive interface for users to communicate with the shell through the command line.

Yash Goyal's Contributions

1. **SimpleScheduler Implementation**: Yash took the lead in designing and developing the core scheduling logic within the SimpleScheduler program. This involved creating the ready queue, handling signal events, and orchestrating the scheduling loop.
2. **Round-Robin Scheduling**: Yash implemented the round-robin scheduling algorithm, ensuring that each process in the ready queue received a fair and equal share of CPU time, based on the time quantum.

3. **Signal Handling**: Yash was responsible for setting up and managing the SIGALRM signal for time slicing. This signal ensured that processes were paused and scheduled in a timely manner.
4. **Execution Time and Wait Time**: Yash implemented the functionality to track the execution time and wait time for each process. This involved keeping accurate records of how much time each process spent executing and waiting in the queue.