OS Assignment - 2 Report

Submitted by Group ID Number - **62**

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

Contributions -

Manit Kaushik	Akshit Gupta
• Implemented the **main** loop for reading and executing user commands and the command parsing logic to separate commands and arguments. • Implemented the launch methods (**running_command & running_command_with_pipes**) to create child processes and execute user commands, for both cases whether pipelining was required or not. • Added and modified code such that SimpleShell can execute the commands from inside a **Shell Script**	• Implement error handling wherever necessary in the code. • Writing SimpleShell implementation in the assignment report. • Checked command restrictions for SimpleShell ,and reasons for not supporting certain commands in the assignment report. • Implemented the **history** command to display command history. • Ensured and implemented that history is printed even when **ctrl+C** is entered.

SimpleShell Implementation -

The SimpleShell program is a basic command-line shell that allows users to execute commands, manage command history, and exit the shell. It supports both single commands and commands with pipes. The program provides a basic user interface for interacting with the shell.The code uses several global variables to maintain and access information about executed commands, including process IDs, start times, total execution times, and the command history. Some of them are -

- **pid_list[100]**: An array to store process IDs of executed commands.
- **start_time_list[100]**: An array to store the start times of executed commands.
- **total_time_list[100]**: An array to store the total execution times of commands.
- **no_of_commands**: An integer to keep track of the number of executed commands.

- ➢ **cmd_list[100][100]**: A 2D array to store the command history.

The code defines a signal handler function sigint_handler to handle the **SIGINT** signal (**Ctrl+C**). When the user presses **Ctrl+C**, the program displays the complete history of executed commands and exits gracefully.Functions **running_command** and **running_command_with_pipes,** they are responsible for executing single commands and commands with pipes. **running_command** takes a command name, an array of command arguments, and the number of pipes as input, whereas **running_command_with_pipes** an array of command arguments, the number of pipes, and the input string as input. **running_command** function forks a child process and executes the specified command using **execvp**. The child process waits for the command to finish, and the parent process waits for the child to complete. The same process occurs **num_pipes + 1** times in **running_command_with_pipes** function after the function splits the input string into multiple commands based on the pipe symbol (|) and stores them in the args array.Then in the main function main execution loop is contained, where the user can input commands and interact with the shell. It registers the **sigint_handler** function to handle **Ctrl+C** signals and repeatedly prompts the user for input and processes the entered commands. **SimpleShell** supports commands like "**exit**" to terminate the shell and "**history**" to display the command history.It also handles errors and displays error messages if command execution fails.

BONUS PART - Running Bash or Sh Files The code includes a bonus feature that allows the user to execute Bash or Sh scripts by specifying the script file as a command. It uses popen to execute the script and displays the output.

SimpleShell Limitations -

1. Whenever a file is to be used in any form whatsoever, the shell runs only when the full path of file is entered by the user. Giving the local path of the file in use shall result in an error.
2. **cd command -** SimpleShell will not execute accordingly when this command is used, the reason for this is because built-in commands like **cd** are not executed as separate processes like external commands (e.g., ls, echo) are. **cd** is not an external program but rather a shell-specific command that changes the working directory of the shell process itself, not a child process.
3. **alias command -** SimpleShell doesn't run this command because it doesn't have support for handling shell built-in commands like alias. In most shells, built-in commands like alias are part of the shell's internal functionality and are not executed as separate processes. Instead, they modify the shell's environment and behavior directly.

Private Github Repository Link - https://github.com/ManitK/CSE231-Assignments