Operating System
Assignment_2(Design_Document)
Group_18

Explanation of Simple Shell :- A Shell is a command line interface that allows users to interact with an operating System by running commands.

Major Components and Functions in the given code for create a Simpleshell listed here:-

- **Header Files:->**The code includes several standard c library and system header files to access various functions and system calls and those header files are:-
    1. #include<stdio.h>
    2. #include<stdlib.h>
    3. #include<string.h>
    4. #include<unistd.h>
    5. #include<sys/types.h>
    6. #include<sys/wait.h>
    7. #include<sys/time.h>
    8. #include<errno.h>
- **History File:->**A history file named “hostory.txt” is used to store a record of executed commands including their process IDs(PIDS) and execution times .
- **Struct Command History:->**It is defined to store information about executed commands .Including the command itself,its PID and execution time.
- Main Function:->
    1. It initializes variables like current_dir,initial_dir,input.execution and history_count.
    2. It enters an infinite loop to continuously read and execute user commands until the exit command is entered .
    3. The loop displays the current working directory (promptdisplay function) ,reads user input (user input function),and processes the input command).
    4. It updates the history of executed commands and their information (history array) and maintains an account of executed commands (history_count).
- **Promptdisplay Function:->** It displays the current working directory as a command prompt.
- **Run_Command Function:->**
    1. It takes an input command ,forks a child process and executes the command in the child processes using **excel.**

2. It records the start and end times to calculate the execution time of the command.
3. The information about the executed command is then stored in the history array.

- **WriteHistory Function:->**This function writes the history of executed commands to the **"history.txt"** file, including the command, PID, and execution time.
- **Launch Function:->**
    1. It is responsible for processing user input.
    2. If the input is "exit," it writes the command history to the history file and terminates the shell.
    3. If the input is "history," it displays the history of executed commands.
    4. Otherwise, it calls the run_command function to execute the entered command.
- **User_input Function:->**It reads user input from the command line, allocates memory to store it, and trims any leading or trailing spaces or newline characters.
- **Printhistory Function:->**
    1. This function is used to print the content of the “history.txt” file,displaying the history of executed commands with PIDs and execution times .
- **Temphistory Function:->**It is responsible for displaying the history commands during the program's execution.It is used when the user enters the “history command” .
- **File Operations:->**The code opens and closes the “history.txt” file for reading and writing as needed for storing and displaying command history.

Overall ,this code implements a basic interactive shell that can execute commands entered by the user ,record their execution history ,and displays that history when requested .It provides a simple interface for users to interact with the system via the command line.

Limitations of this code:-

1).cd Command: The cd (change directory) command cannot be executed directly within the shell. When you use cd to change the directory, it only affects the child process generated by fork(). When that child process exits, the parent process (your shell) remains in the same directory. So, while it may appear to

change the directory within the child process, it doesn't affect the parent process's working directory.

2).jobs Command: The jobs command is a built-in shell command, which means it is handled by the shell itself rather than being an external executable like other commands (e.g., ls, cat).

3).Backgound Command: When we run a command using the run_command function, it waits for the child process to complete using waitpid(). This means the shell will not accept new commands until the currently running command finishes.

Contributions for this assignment:-

Vipul Verma :- run_command ,user input,launch,Error handling,File history,Documentation, these are methods and other parts of this code completed by Vipul Verma.

*Wasif Ali:->*Remaining part of launch method, print history, temphistory, run_command,File History these are methods and other parts of this code completed by Wasif Ali .

Git Hub_Link:- https://github.com/A-WASIF/OS