Os Assignment 2 - Design Document

GROUP-21- Abhirup Das(2022019) & Armaan Singh(2022096)

Contributions :-

1. **Armaan Singh -** SIMPLESHELL boilerplate, all the makefiles, history(), signal handling.

2. **Abhirup Das -** Generalising of shell_loop(), pipes handling, background processing, running scriptable files.

SimpleShell Implementation :-

- We start with our code in main.c which calls the ‘shell_loop()’ method from the ‘Shellcommands/shell.c’.

- The ‘shell_loop()’ method then runs a do-while loop take in the user input. The input is the checked for “|” (for pipes) or “run”(to run the .sh file) or “ ” to check if nothing was returned then resend the prompt. This input is stored in a global variable called ‘user_input’, which is printed in a systematic format when the user enters ‘history’ as input.

- The ‘shell_loop()’ calls in the ‘launch()’ method which handles all the execution of commands, including piped commands and also commands that are not in the usr/bin(like history,cd,mkdir,rmdir), and if a command does not exist, it would print the necessary error message.

- When Ctrl+C is encountered, the signal handler defined in the ‘shell_loop()’ calls for the ‘Escape_Sequence()’ to print out all the entered commands by the user their respective pids and their duration along with start and end time stamps.

- For further details, please refer to the ‘Readme’ in the git repository.

Github repo link :- AbhirupDas04/OS-Assignments: Repository of OS Assignments of Armaan Singh and Abhirup Das (github.com)

Command to run any .sh file :-

We use the ‘run <file_name.sh>’ inside the simpleshell program to execute that particular .sh file.

Additional commands that can be implemented :-

1. clear
2. gcc (with args)
3. date
4. rm (with args)
5. pwd
6. mkdir

…and all other UNIX commands that are present in the /usr/bin folder.

Commands that are not Supported :-

1. **cd -** The function that we tried to use to implement ‘cd’, ‘chdir()’, changes the current working directory only in the child process, which

meant that when we exited from the child, the parent process would still be in the same directory and ‘cd’ wouldn’t actually be performed. We couldn’t find a convincing way to make it work. Also, it isn’t in /usr/bin, so we couldn’t directly ‘execvp’ it.

2. **fg/bg -** There didn’t seem to be a way to obtain job ids or even details about any job, so it was impossible to make them go to the background or foreground at will. Of course at the start of execution of command, we can use ‘&’ but not in the middle of it.

3. **exit -** There seemed to be no way to close the terminal, so it can’t be implemented.

4. **set -** We couldn’t figure out a way to obtain all the environment variables at once and then subsequently modify them based on our needs. We could change one of them using setenv() but not multiple of them.

Assumptions :-

1. The command that we are using to run the ‘.sh’ file will not be shown in the history but rather the commands given in the ‘.sh’ file. Similarly on Ctrl+C, the command ‘run’ won’t be visible, rather only the other commands.