OS Assignment-2 Design Document

Topic: SimpleShell: A Unix Shell in C from Scratch

Team Members: Medha Kashyap (2022292)
Himanshu Raj (2022216)

Group Number: 10

GitHub Repository Link: https://github.com/rahi-senpai/OS

Members Contribution:

- Medha Kashyap: process creation, termination part, documentation
- Himanshu Raj: exec and pipe implementation, bonus part

References:
https://man7.org/linux/man-pages/man3/exec.3.html
https://man7.org/linux/man-pages/man2/pipe.2.html
wait(2) - Linux manual page - man7.org
dup(2) - Linux manual page - Michael Kerrisk
strftime(3) - Linux manual page - Michael Kerrisk
Lecture slides

Explanation:
In this we have implemented a SimpleShell that waits for user input, executes commands provided in the user input, including piped commands and bonus parts - background processes and shell scripts and then repeats until terminated using ctrl-c. In our code the 'main' function initializes the signal handler (for this we have declared a function which sets up a signal handler for ctrl+C (SIGINT) to terminate the code) and enters into the shell loop, in which there is an infinite loop where the shell continuously reads the user input (using the "read_user_input" function which removes the trailing "\n" character), processes commands in "launch" and "create_process_and_run", and waits for the command execution in

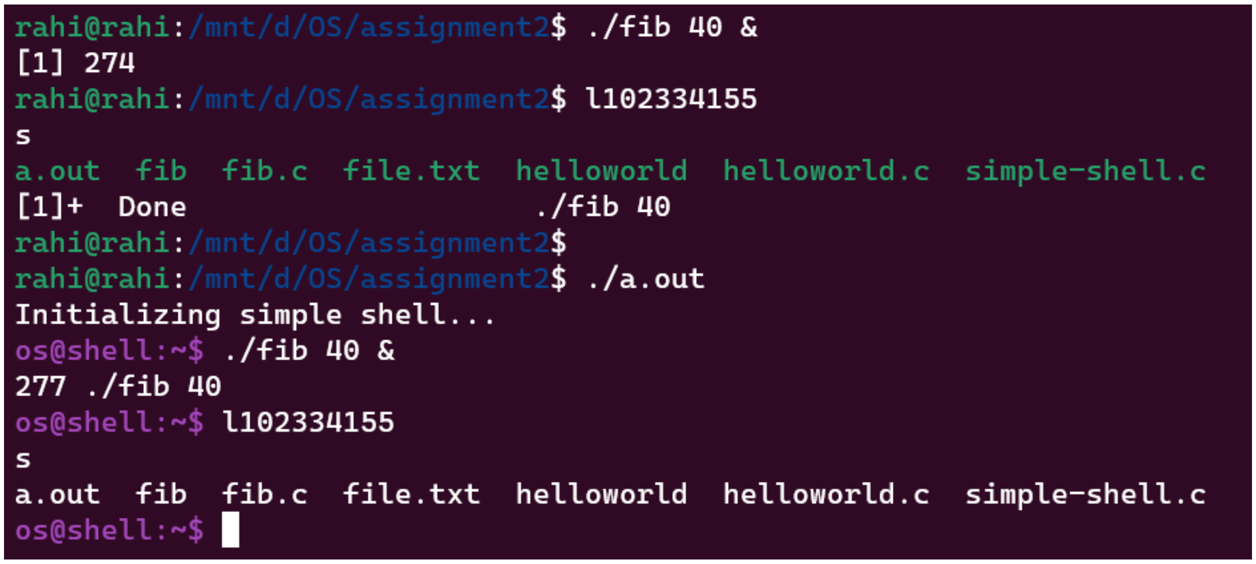
“create_child_process” to complete. It updates the command history with execution details. At the end during termination the “termination_report” function prints a summary of executed commands, PIDs, start, end and execution times.

Commands which cannot be executed: (This list is not exhaustive)

1) **cd** - this command changes the directory over the terminal to essentially modify the internal state of the shell, so a running c program cannot change its directory during execution.
2) **export** - this command is used to set environment variables which are internal settings of the shell, so it cant be executed in the simple-shell.
3) **unset** - this command works very similar to export, the difference being it removes environment variables, so its execution is not possible in simple-shell.

Limitation:
We have only used static memory, so there are certain restrictions over input size (200), number of pipes (9) in a single prompt, number of words (50) in a prompt and maximum number of history records (100) in a single execution.
Also we have implemented ‘&’ for background processes and not as command separator and ‘&’ can be used with pipes, so no problems with that.

We have attached an image showing background process execution (./fib 40 &) in terminal and our program, and want to highlight the fact that the output from the background process in terminal just pops up when we are giving any other command (ls,here, or any other command) as input and that exactly what we have tried to implement in simple-shell.c



```
rahi@rahi:/mnt/d/OS/assignment2$ ./fib 40 &
[1] 274
rahi@rahi:/mnt/d/OS/assignment2$ l102334155
s
a.out  fib  fib.c  file.txt  helloworld  helloworld.c  simple-shell.c
[1]+  Done                    ./fib 40
rahi@rahi:/mnt/d/OS/assignment2$
rahi@rahi:/mnt/d/OS/assignment2$ ./a.out
Initializing simple shell...
os@shell:~$ ./fib 40 &
277 ./fib 40
os@shell:~$ l102334155
s
a.out  fib  fib.c  file.txt  helloworld  helloworld.c  simple-shell.c
os@shell:~$ 
```