**CS 250 Operating Systems**

**Assignment 2 Report**

**Name- Yash Teckchandani Date - 13-03-2022**

**Id - 12041770**

**Email-** [**yashteckchandani@iitbhilai.ac.in**](mailto:yashteckchandani@iitbhilai.ac.in)

Slip day used - 1

**##**

**INSTRUCTIONS FOR RUNNING THESE FILES ARE GIVEN IN README**

**##**

**PART A**

In this part , we made a simple linux like shell using c.

Making a shell using c was a big challenge since we were not taught c before and that’s why some of the commands are taken from some sources which we have cited in references.

The basic logic behind the working of this basic shell is the system calls which are generated while performing a process.

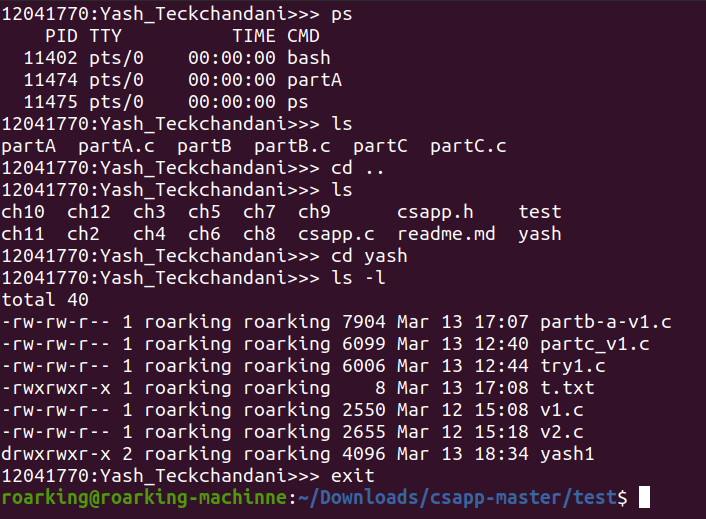
Using this assignment we learned about various system calls in Linux such as fork, exec, wait, exit.

In making this shell, we made the following approach -

1. Take user input
2. Fork a child process
3. Execute the command
4. Wait for the child process to finish
5. Repeat the above steps

We haven’t created our own functions but used the ones already present in linux /bin directory. Since cd is not available in /bin , we used the chdir command in c to change the directory.

This completes the basic functioning of our shell created in part A.



**PART B**

In part B, the major task was to add pipe (“|”) and multiple commands (“&&”) functionalities.

In pipe function, output of one process goes to the input of another process.

This is done using c pipe command and creating file descriptor for both the ends.

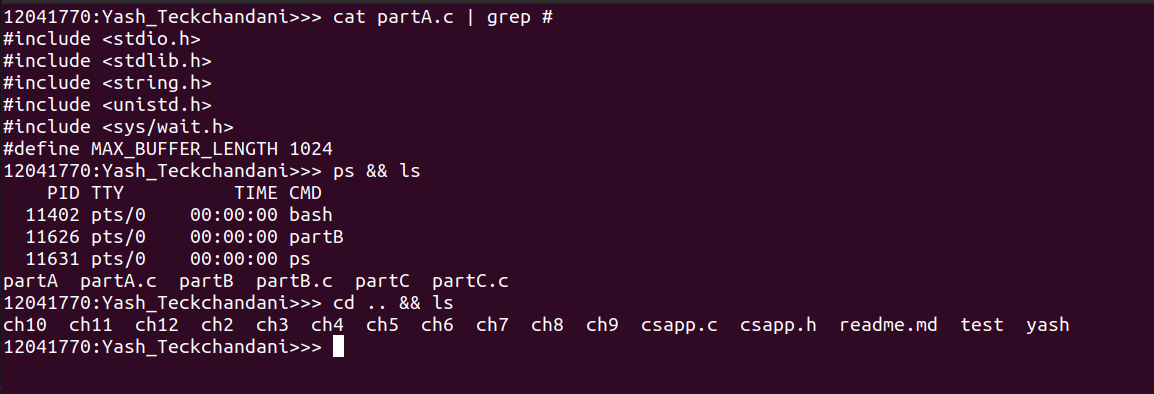
For the code I have made, only a single pipe works. Multiple pipes execution was out of my scope.

Basically how pipe works here is by duplication the Standard output of first process to the read end of file descriptor and the putting this to the standard input of second process by setting the file descriptor for this as write.

First we fork the first process and then the second process.

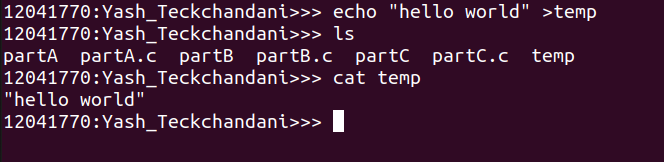
At the end we have to close both ends of the file descriptor.

For multiple commands (‘&&’) , we took the input character stream separated by spaces and store them in sequence to a temporary array. As soon as we get “&&” , we replace it with NULL in the temporary array and send this array for execution. If this executes successfully then we clear the temporary array and continue the same process. If the execution fails we break out of the parsing.



**PART C**

In this part , we added the redirect function. It was easy to do this after learning how pipe works. First we divided the input into 2 parts by splitting by “>” and then converting first string into token of arguments and then forking a child process , executing the first command and then the standard output is stored in a file saved by the name of second string.



**References**

1.<https://www.youtube.com/watch?v=5fnVr-zH-SE&list=PLfqABt5AS4FkW5mOn2Tn9ZZLLDwA3kZUY&index=15>

2.<https://www.youtube.com/watch?v=cex9XrZCU14&list=PLfqABt5AS4FkW5mOn2Tn9ZZLLDwA3kZUY>

3.<https://www.youtube.com/watch?v=6xbLgZpOBi8>

4.<https://www.geeksforgeeks.org/making-linux-shell-c/>

5.<https://stackoverflow.com/questions/28502305/writing-a-simple-shell-in-c-using-fork-execvp>