CSE-3033
Operating Systems
Project2

Süleyman Barış ESER - 150116055
Murat Şenol – 150117039

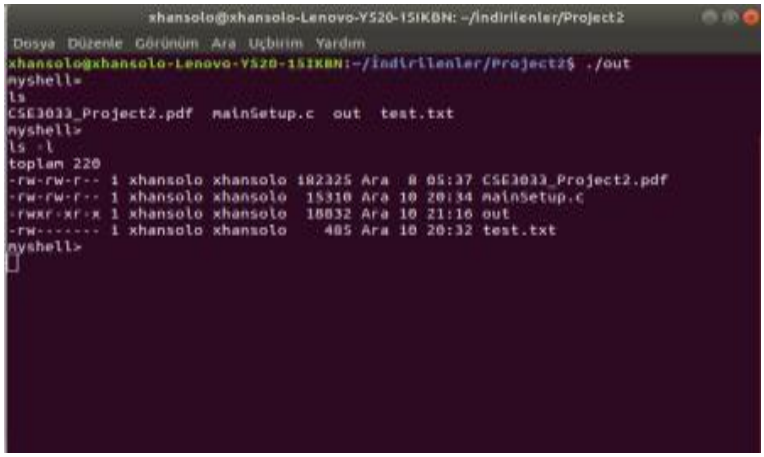
In this project, our aim is to create a User İnterface Shell in Linux by using c.

Part A)

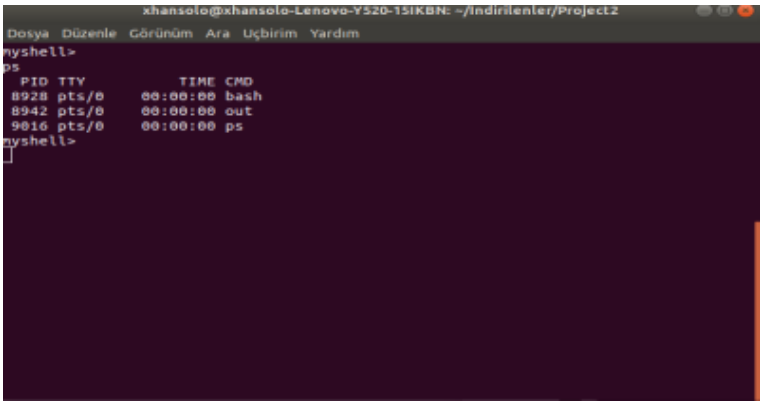
To execute a command with execv(commandPath, args) function, there is need to command exact path. To get this path, there are several different method but we used popen('printenv PATH', 'r') in the commanPath(char **) method. After getting command Path, fork a child and control status. If the status is true for fork() ==0, then we can notice code is in the child process. Then, execv function can be used and show output.

Examples;

ls and ls -l:



ps:

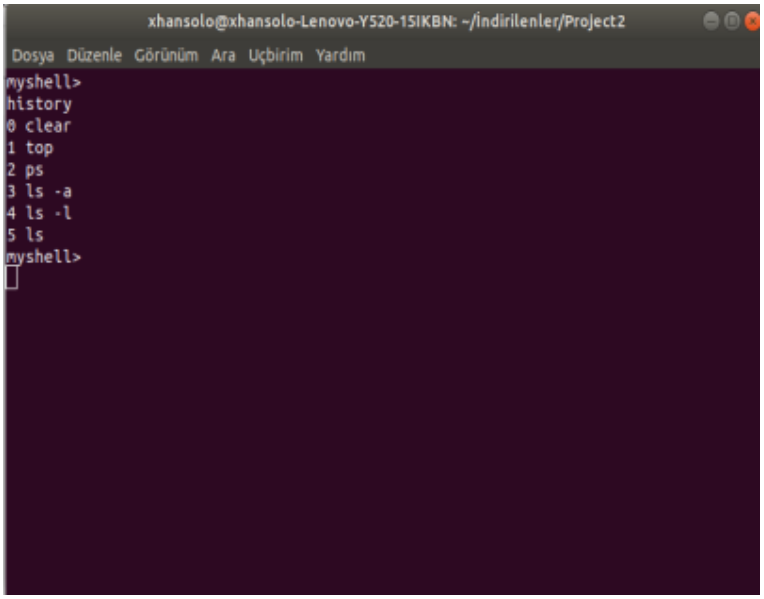


and more command can be executed in this part, for example, clear, htop, echo, etc. And processes can run in background by putting '&' to the end of command.

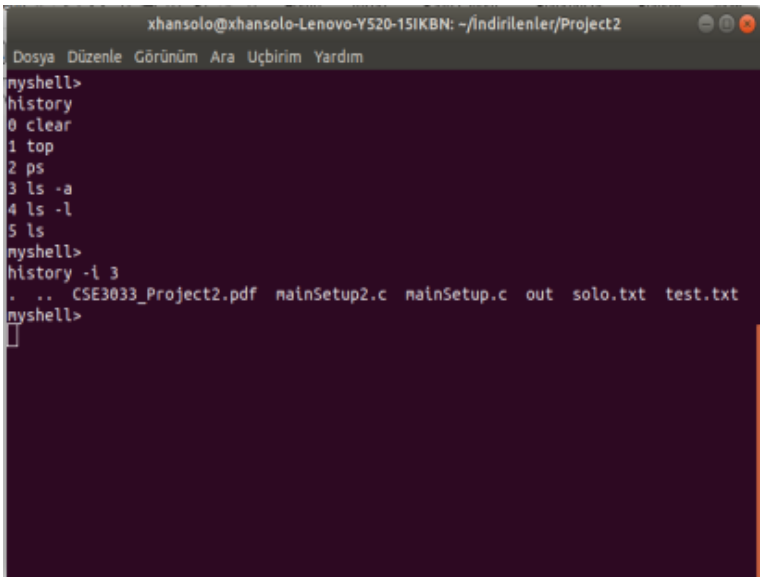
Part B)

In this Part, to keep all old command in the code, there is a history array. At each process, the commands are adding to history array. If the user want to see the command history, he/she can print ‘history’ in the USS terminal. If the user want to re-execute the an old command, the user can print ‘history -i num’ , ‘num’ is the place of the the commandin the history array.

history:

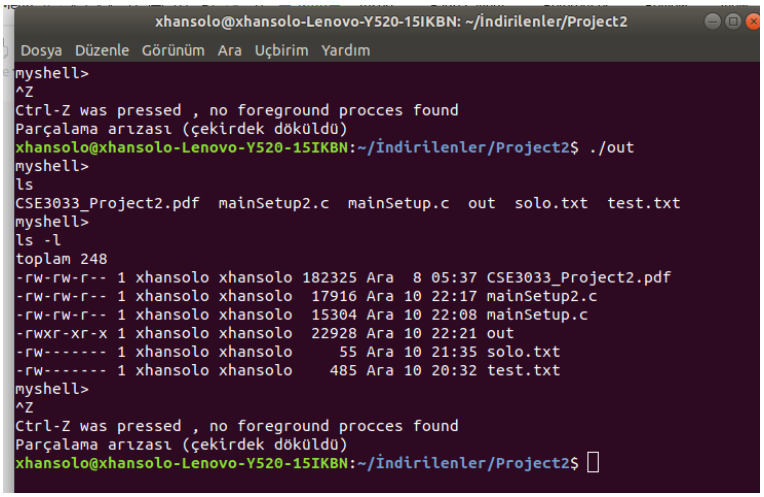


history -i num:



if the user press Control + Z then, code checks wheter there is still a foreground process or not by handling the SIGTSTP signal. If there is then stop it, else then, do nothing.

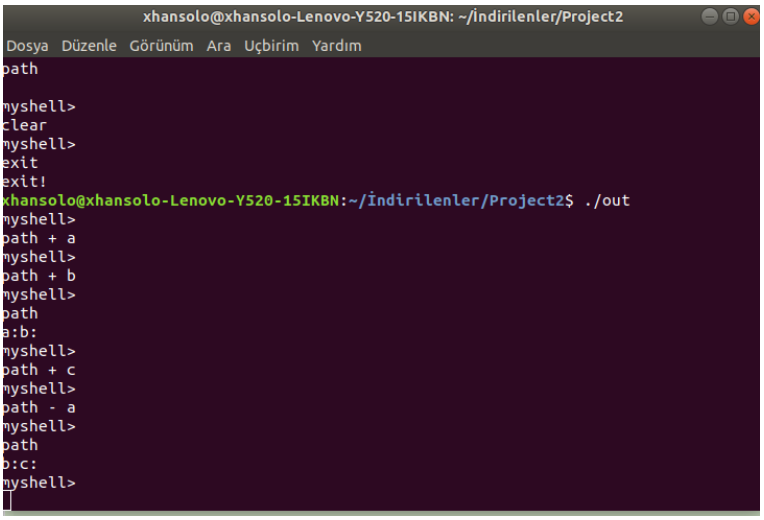
^Z:



But in this part, we can detect the the press and do the operations but, often the segmentation error occurs.

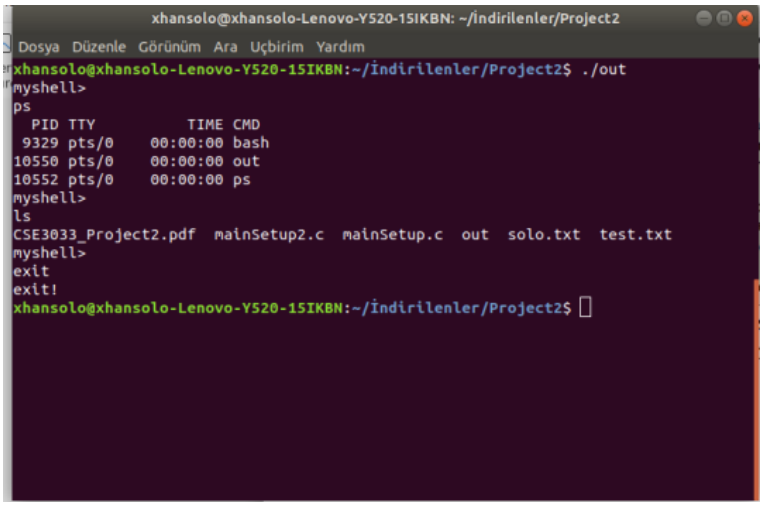
If the user print path, the executable files that added by the user shown in the terminal. The user can add or delete any path from the path array.

path:



If the user print exit, then the exitShell function by using waitpid(-1, NULL, WNOHANG) controls the number of child processes that run in background. The Function stops them and exit from code.

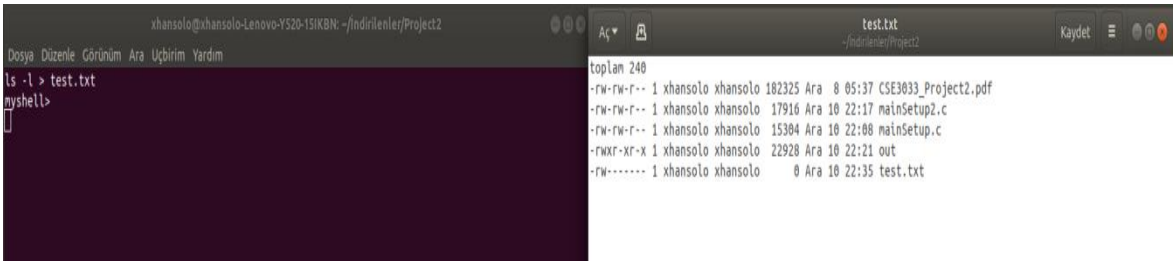
exit:



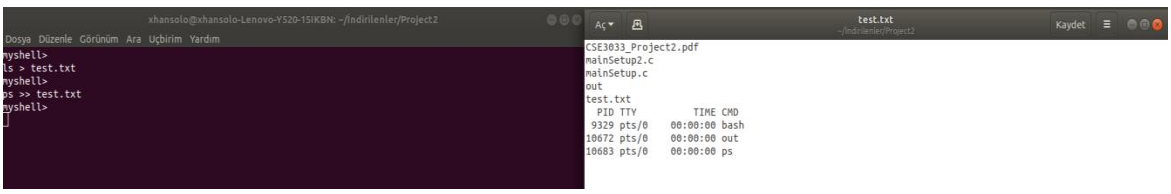
Part C)

If the user print redirection command, then in the execShell function controls the redirection type. Then, the function create a pipe in and pipe out to connect the out file or in file with the program. Then, the function create a dup2() and write or read the file.

command > a.txt:



command >> a.txt:

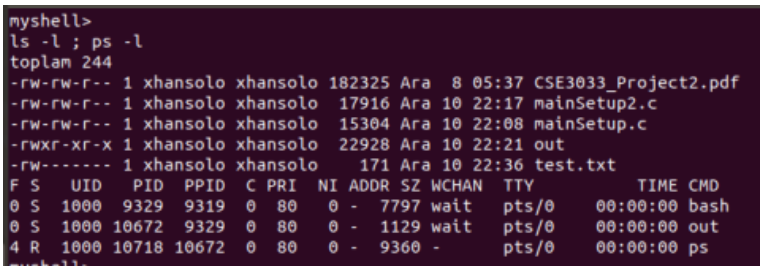


And, can do ‘program < file-in > file-out’, ‘program < file-in’ and ‘program 2> file-out’.

Part Bonus -)

Split the args from ‘;’, and adding them to an array. Then, execute them one by one.

command 1 ; command 2:



***But in this part, when we use a for loop, the segmentation error occurs and code exits abnormally. So, code execute them one by one. It works for just 2 command.