Vaibhav Malhotra
CPR E 308
Project 1
Section F

SUMMARY

The project really helped me to learn and apply the concepts that I learned in my class that helped me better understand the point of the functions implemented in our environment. The simpler version of the UNIX shell that was implemented by us to understand the concept of the functions that we normally use in a bash shell. For example, we were asked to implement a prompt function where the user could give a name to the shell and then the shell would run in an infinite loop to execute the users command.

The infinite loop I implemented works in the **main()** function. The function takes in a string in the form of a char array and then passes it to a **split()** function to split the arguments in to different words so as to execute the commands correctly. The loop continues to run if the user has called an inbuilt function, or a different function, or anything other than exit. If the user enters exit then the loop will break and the shell process will stop accepting commands from the user.

Moreover, we can look at the different requests given by the user, which are handled differently depending on what they are. The learning process upon figuring out how to implement these different categories was a bit tricky. For example, I had difficulties with **ufunx**() for the non-built-in functions. The main reason was trying to figure out the correct logic for parsing the right arguments.

Another difficulty I faced was to implement the **fork**() function correctly and also keep the track of the current processes id but the previous labs helped me a lot to implement this process. I also faced issues for implementing the **set** function because I wasn't passing the correct number of arguments to my **setenv**() function.

I also noticed that I was getting a warning from my compilation about using gets function. I got the following warning:
shell.c:175:4: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:510) [-Wdeprecated-declarations] - gets(line);
I got the aforementioned error because the gets line might cause overflow and I was able to fix this warning by using **fgets**() function as this avoids the overflow problem for a char array.

Finally, what I thought was going to be the hardest, foreground processing, turned out to be the easiest to implement. Just by implementing a flag and getting the last position of our command buffer character, I was able to apply process waiting and background processing easily.