Jacob Kurian

10/09/2016

Project Report: Lab 2 (The Shell)

**First**

* Describing the project based on my understanding:

The objectives of the project is to make the students understand the functionality of the shell in the Linux system. This project help me to understand how does the basic built-in functions works. Also it help me to understand the concept behind input/output redirection. The idea behind piping is also clear because of this project. Now I understand of how does the system calls such us exec (), write (), read () works. The importance of fork () functions is also very clear now. The concept behind the background processors are also clear.

* What are Given
  + We have got many sample works from professor and the TA. The samples include the example of input/output redirection, dup2, and piping.
* What Are Required
  + Implement all the built-in commands
  + Implement a way to execute all the non-built-in commands
  + Use and improvise the given examples to implement the piping and input/output redirection.
  + Implement Method for background processors
  + Implement the methods to get the path name
  + Implement method to execute the batch file
  + Implement separate input/output redirection for the built-in
  + Implement append and trunc options available
  + Create a sample user manual with groff format.
  + Create a readme file for the Grader.
  + Create a ‘makefile’
* What methods are suggested
  + All methods that mentioned above are suggested. Also professor have mentioned that split a large method into few small ones. So that that it would be easy to find the bugs if there is any.
* What step I follow
  + First I create a method to parse the user command properly.
  + Then I create a method that execute the non-built-in functions.
  + Then implement the built-in methods and tested each method one by one.
  + Then I create the input/output redirection methods.
  + Then start to implement piping.
  + The user manual

**Second:**

* My solution and approach
  + I have created a detailed pseudo code for this project, then just follow that.
  + Whenever I have to write a long function, I tested each part of the function by putting print statement.
  + Also I tried to split the methods into small different ones as possible, but sometimes it make more sense to me if I just implement a single large method than many few ones.
* The Structure of My main methods

int main(argc, int \*argv[]){

* Check the user pass a bath file or not
* Get the path name;
* While(1){
  + Get the user input
  + Parse the user input
  + Check weather is it a background processor or not
  + Check piping or input/output redirection involved. If piping involved call the specific method for piping

if(no piping involved){

check the user command math with built-in command methods

if the user command is a non-built-in

* + - * Fork()
        + In the child

exec(the commad)

* + - * + In the parent

Wait if is not a backgroung process

* + Update the path name;

}

}

}

* Basic Data Structure of the Lab;
  + I use double char array to store the user input.
  + But I could not able to free the pointer so far (date: 10/09/2016). But I will try to free those pointers again, if I succeeded I will update the report. So far it is working perfectly without free () function.
  + I created different C files for piping methods, redirection methods, and parsing methods.

**THIRD**:

* Bugs/Problems and it’s solution:
  + Bugs in the command line parsing was tedious. It crashed my shell several times. After re-creating the parsing method the problems is solved.
* Piping
  + The problem caused by piping function was also hard to solve. It took me almost three days to solve. Whenever I call the piping method, my shell never come back and ask the user to execute the next function. I could not figure out why it was doing it. Then professor upload his sample code for piping in blackboard. I use that code but still the same problem occurred. Then I realized that this is happening because of the zombie processor. So I call a wait function in the main with a flag and it fix the problem.
* Input/output Redirection for built-in commands:
  + Input/output redirection for the built-in command s was little hard. Whenever I tried to redirect the output of the echo command to a file, it always print on the screen. Somehow the method that I implemented for the echo function don’t work properly with redirection. But after I fixed my string parsing method it work perfectly.
* Requirements I have Finished:
  + Make piping works
  + Make background processor work
  + Implemented all the built-in methods
  + Make input/output redirection works
  + Path name will be updated each time
  + Shell will execute batch files
  + Input/output redirection work has append and trunc options.
  + Create user manual
  + Create make file

**Fourth:**

* Conclusion and Thoughts
  + The project help me to understand how does a shell work. What is the importance of the fork () function in the shell operation. Also it help me to understand the functionality of dup and dup2. Also learn how to create a man page using groff format.