The design of project2

By Shuaiqi Cao

This project aims to achieve some shell functions, inlcuding command phrasing, I/O direction, pipes and exit command. Thus, six major parts are designed in this project:

1. Shell prompt
2. Read input
3. Command phrasing
4. Single command execution
5. Redirection
6. Pipes

In the main function, a while loop is used to accept user commands circularly. In the loop, the shell prompt is called firstly, which will provide the current working directory. This function can be realize by using function getcwd().

Then, use a read input function to get the input string and store it as a char*. getline() can do this function and return an char* for later using.

Once this input string is read in, use the command phrasing function to determine its type. The type of this function is int, because the function can return different integer number to represent different operation type. The parameter of this function is designed as a char*, since the simple interface will be easy to implement. This function only does the work of checking validation and classifying operation type. The splitting of command will be put in later stage. In this function, it

checks input length, checks invalid character by comparing Ascii code. Also, it check the number and position of special operators "|", ">", "<". By comparing their number, the cases of repeating "<" or ">", "|" and "<"/">" in same sentence can be filtered. By using the position of operator, it can check that wether the output, input files are valid. For the return value of this function, 0 means invalid input, 1 means single command; 2 means pipe; 3 means redirection.

The return value of phrasing function can be checked in main function. A switch structure is used to choose different cases.

For the single command, the execute function firstly split command to some arguments. then fork a child process to handle the command with its arguments by calling execvp() function. If the input is "exit", change the number of status, which will stop the loop, and exit the shell program.

For the pipes, the execute function creates a 2d array to store the commands and arguments. The size of the array is determined by the number of pipes. Then it build necessary pipes, use dup() to transmit data, and make the pipes be one direction to realize multiple pipes. Fork child processes according to the number of pipes, and call execvp() in each process.

For the redirection the execute function firstly split the input according to ">", which can separate the output file name. Then split the rest part according to "<", which can separate the input file name. Fork a chile process to execute the command.