**Program description:**

To accomplish the task of creating a simple shell I started of with making a main while loop that prompt the user repeatedly for input. Also, the while loop looks to see if a batch file was sent at the start. When a batch file is found then the program will start reading each command from the file and executing them. Once typed-in user input is acquired it is sent to a parsing method that would take each word delimited by a space and put it in an index of an array with the last index as a null terminator. The parsing method has check after it parses the command which is seeing if there is a “&’ indicating the program will be executed in the background. When this symbol is found at the end of the command then a flag will be set so the parent will not wait for the children executing the command and the output will not be displayed on the screen. A couple other checks for the parsed command are in the while loop. The first check is to see if the command has redirection such as a pipe or I/O redirection. If the command has redirection then a condition flag will be set accordingly depending on what type and the redirection symbol will be removed from the parsed command. In the case it was I/O redirection, the command will be sent to a method with various if-statements to check against to see what type of I/O occurred. when found, the file being overwritten, appending, or output taken from will be opened and duplicated. Then the new parsed input will be executed so, the output will be written or the input will be taken in by the command. Now if the command had a pipe, the command will be sent to a method to parse the users input again into two different arrays. One array containing the command that needs the output and the other array with the command that needs the input. Afterwards the new arrays will be sent to another method that uses pipe() to create the pipes. Next step is to fork 2 children, one for each array and then the appropriate file descriptors for each child are duplicated allowing for the output of one command to be used as input in another. Both commands are then executed in their respective child. The second check in the while loop is if the command is internal or external. To check if the command is one of these two, it is put through a loop that compares the string typed with a herded coded array of internal command key words. Whenever the compare finds a match, it will call an array of method calls that will execute the proper method for that command. When the command is not internal, it will be sent directly to external method which will fork and execute the command.

**Program issues:**

The first issue that I came across was using my method readline() that reads in the user input. The method used fgets() that has a problem with adding a null terminator or new line at the at the end or if you didn’t enter anything. So, I had some up with a couple checks to remove these characters when need

Another issue is with background execution with not getting output to be shown. The one way that worked that I really don’t know is that I closed all the file descriptors 0,1, and 2. Then opened a new output paths with /dev/null.

There was an issue with the piping method. Whenever 2 correct commands where put in the pipe format the wrong way like if the command that needs the input is output place and vice verses. The program would crash. I tried putting error cases in the pipe method, but nothing seemed to worked.

Last issues is when running a batch file my cuts of the last character in the command . so to run with a batch file you have to put a space after the command in the batch file for it to run properly.

**Test cases:**

**Internal commands:**

* run commands with short correct input
* entered incorrect and correct to see how the error cases would handle both

**Parsing and reading in user input:**

* entered a ton of symbols, characters, and spaces
* had print stamen to view output

**Piping:**

* piped commands with small output
* piped commands with larger output
* entering the command wrong
* switching the input and output orientation of the commands (like the command that needs the input is now on the side of the pipe that sends the input)

**I/O redirection:**

* wrote to files with small output
* wrote to files with large output
* append to files multiple times
* tried entering bad file names

**Batch mode:**

* ran with one simple command at a time
* ran slightly harder commands with arguments
* ran a ton of commands

**External commands:**

* entered simple commands like ls
* then entered commands with arguments add to them like “ls -l -h -a”
* entered bad simple commands and commands with arguments

**Background execution:**

* tried to see if my check that looks for the “&’ at the end and delete will delete that symbol when its anywhere else
* executed background execution with external, piping, and redirecting multiple times