**Program Summary**

The purpose of this program was to simulate the operations of a linux shell. A command is provided as user input to the program at runtime. The program tokenizes each command and loops through each token determining its type. The type is determined by following a simple line of logic. The first token of the command line and the first token after a pipe is assumed a program. The first token after a redirect token is assumed an output/input file. If the token is not a pipe or redirect, it is assumed a command line argument to a program. The program uses global flags to set when a pipe, redirect, or command is found. After the type is determined, a statement describing the type is printed to the screen.

The most difficult thing about this program was determining what to do if a pipe is connected to a token without a space. The original command could only be tokenized using a space delimiter as the pointers to the delimiters are lost after the strtok function. Since the strchr function was used to determine if a pipe was contained within a token, the pointer that is returned was used to split the token into smaller tokens. The pointer location is replaced with a null terminator and the token is printed. Then, a pipe statement is printed and the pipe flag is set. The token pointer is set to the pipe pointer (+1) and the pipe pointer attempts to find the next pipe.

This program could have been simplified by using a pointer to loop each character in the command. When a space is found, replace the space with a null terminator and print the command. When a pipe is found, print a pipe statement. The same globals could be used to determine each tokens type.

**Program ToDo List**

This program will later need to implement a way for each token to be allocated into a string array. The print functions will also be simplified by accepting in a second parameter as the token type and then printing the corresponding statement.

**Test Cases**

All of the test cases provided in the assignment description were applied and the program output the correct response. Test cases with multiple commands were given as runtime arguments were also applied (for example, './a.out "ls -l" "ls -a | grep project") were also applied and gave the correct response. Some test cases were used that contained several pipes connected to tokens without spaces and the program appeared to give the correct response (ls|wc|grep|ls -a dir|pro| cmd >>file). However, when a redirect does not follow a command, the program assumes the redirect is itself a command. This is a possible bug given that the user knows a redirect is not a command, but this program assumes the first token must be a command or program.