OS write up

To implement the shell, we basically design a three stage pipeline. In the first stage, our program breaks the line of command into token groups. Then in the second stage, the program breaks up the token group into an array of words, files or operators. In the meantime, the program parses the tokens to see if they violate any of the rules specified in the project description. Finally, in the third stage, each token group is implemented.

To be more specific, in the first stage, we first check whether the read-in command is ‘exit’. We then check to see if the command is more than 80 characters. After that, we break up the whole command into token groups using strtok() command. We also handle the special case like “ls|uniq” which is not allowed according to the project description.

Before diving into the second stage, we first back-up stdin and stdout. The reason for doing this is to create a way to handle error message easily. For example, if in the child process, an error occurs when the stdout is connected to a pipe, then the error message will be printed out to the pipe instead of to the console. To avoid this behavior, we will recover stdin and stdout before the error message is printed. Then in the second stage, we parse the tokens in each token group. We first parse the command part of the token group. We use strtok() with a while loop to break up the command and its arguments and store them in an array called tokens. Each token is checked against the rule for a valid word. We stop parsing when an “>” or “<” is reached.