Hillari Denny
CSCE A320 – Operating Systems
Assignment 3 – Shell implementation in C

The basic concept here is to get user input, tokenize, and then create a child process to execute the commands and pass the information back to the parent. This exercise is designed to teach the fundamentals of inter-process communication.

First, we will take the user input and store that into a buffer. There are several different methods for this (fgets(), read()). In this implementation we will use the library function getchar. This is so it can return any valid character, including meta and EOF. We use this function to store the use input into a buffer.

Next we tokenize the buffer using the function strtok to obtain the commands and the associated arguments, and store these into an array. We must then determine whether or not there are any redirection or piping symbols. In the case that there is a redirect symbol, we store the file, then remove the character and shift the array. For example, if we have the command “ls -l > foo.txt”, we need to store “foo.txt” as the output file, and both “ls” and “-l” as the elements in the array. If there is a piping symbol we need just to remove the “|” and indicate that we have a pipe so we can perform execution correctly.

Once we have our argument array, we must now create a pipe so that we can use the file descriptors to redirect input and/or output, thus giving us the medium with which to perform our IPC. We will create a child process by calling fork(). Now both of our processes, the parent and a child, have a copy of the file descriptors we created with pipe(). The child process will execute the commands and it’s arguments, and pass the output of these commands back to the parent process.

Therefore we must close the appropriate descriptors in order to pass information correctly. If we have a pipe symbol in our command, we must fork a second child to handle the execution of the other file. This is accomplished in the same manner as creating the first child.

The parent process will receive the command results from the child and print it to stdout. We must then ensure that we wait for all children to die to avoid creating zombies.