Nov, 10th, 2019

Assignment 4 - Chapter 3 Programming Project

Report

This project consists of designing a C program to serve as a shell interface that accepts user commands and then executes each command in a separate process. The project is meant to support creating the child process and executing the command in the child, providing a history feature, adding support of input and output redirection, and allowing the parent and child processes to communicate via a pipe.

- **Part II: Creating the child process and executing the command in the child**

  For this part, first I created a commandParser function, to take the input of the user for the command, and then tokenize it into arguments. Inside this function exists another function checkForFlags, to set the values of the flags according to the existing arguments in the arguments array. Then, after parsing the command. Another function to remove the flags is called, and then fork() is called to create the child process, and executes it either alone or concurrent with its parent depending on the value of the concurrency flag that is set based on the existence of the & symbol at the end of the command.

- **Part III: Providing a history feature**

  For this part, I added another part at the beginning of the commandParser function, to check if the command is "!!", and if so it prints back the last entered command and executes it again. For this part, I added the number of argument as a parameter to the commandParser function to handle the loop over the previous arguments that get printed.

- **Part IV: Adding support of input and output redirection**

For this part, I used dup2() to change the value of STDOUT_FILENO or STDIN_FILENO from its default value to the fileno value of the file that the user wants to write to or read from. This part only gets executed if the input or output redirection flags are not equal to zero.

- **Part V: Allowing the parent and child processes to communicate via a pipe**

  For this part, I created a new function to handle pipe communication to split the arguments between those before the "|" symbols and those after it, and then by calling fork(), a child process is created to handle the process before the pipe symbol, while the parent process handle the process after the symbol.