CS307 - FALL 23-24 - PROGRAMMING ASSIGNMENT 1

Shell Command Execution Simulation in C

AHMET ALPEREN GÜNGÖR

28847

- In programming assignment 2, the main goal is to read comments from a file, execute the instructions and print the outputs. The task requires to run simultaneously, so that they don't interfere with each other – cause faulty outputs.

  First, the commands.txt file is opened & read using file streams, where each line is read by getline method. Then in order to identify commands, options etc. each word in line is read by a string stream. Since the first word is always the command – which is the program that will be executed - , it is read and inserted into a char array in order to use for execvp arguments. For the next indexes in the char array, the check for words are as follows:

  If the word starts with - , it is an option for the control of the program.
  If the word equals to > or <, then it is a redirection (for input or output)
  If the word equals to &, it is a background process
  If none, then it is an input to be read

  Then, if the command is to wait, everything needs to be finished before moving on. By looping through process ids calling waitpid, putting them into wait state. Then all the threads are looped and joined. For printing to the console, a thread function was used to ensure output do not interfere as stated in the beginning.

  While printing to the file, processes are forked, where parent process checks whether is a background process or not, if so then pushed to the process list, if not waitpid is called for the id of the child process. Child process changes output path using dup2 and calls execvp to execute for given arguments.

  While printing to the console, processes are again forked but this time thread function is used to ensure correct ordering while printing to the console – so they are not intertwined. Parent creates a thread with the function defined in the beginning of the code, in this code a manually implemented guard lock is used in order to ensure everything is written without interruption so that output to the console is as a whole. In case the child hasn't write to the pipe, it implements a spin wait (not the most efficient but could not find optimal values to use something like sleep), if not then another loop is used to print to the console. In the rest of the parent if it is a background pushes to the vector and continues, if it is not waits for child process, joins the threads. Finally at the end, program waits for child processes to complete, closes file streams and clears vectors.