Ege Aktemur 28080

07/12/2022

Programming Assignment 2 Report

In this report I will firstly explain my program flow. Then I will go over some edge cases/Important details and how they are handled.

Program Flow

First read a command from the commands.txt. Then check if it is a wait command. If it is a wait command, wait all previous commands to finish by joining their threads. However if it is not a wait command, get redirection, inputs etc. details. After that create the myargs array. Then open input/output files according to command. Finally open a fork to manage the i/o of the process, fork and execute the command in the child process.

There are 3 possibilities:

1. We should redirect output file
    - Redirect STDOUT to output file
2. We should read from input file
    - Redirect input file to input file
    - Redirect STDOUT to pipe to print later
3. No redirection
    - Redirect STDOUT to pipe to print later

While executing the operation read from the read end of the pipe from a thread we opened in the main process.

There are 2 possibilities:

1. It was a background operation
    - Print thread ids, pipe read end if it's a non output redirection command.
2. It was a foreground operation
    - Wait during the printing period before executing a new line then Print thread ids, pipe read end if it's a non output redirection command.

In the end wait for unfinished threads.

Edge Cases or Important operations

- Special Characters
    - I did not implement anything for this issue since I was implementing using C++ not C which solves this issue. However while doing the first programming assignment in C I had to include an option in the command.
- Redirection (pipes)
    - As I explained in the program flow I solve this issue by holding the outputs in the pipe before printing and printing them in a mutex in a thread created by the command process.

- Bookkeep
    - I do bookkeeping without keeping the process ids but by keeping the threads list.
    - By doing this I do not lock mutexes while executing but only while printing.
    - Additionally, creating a thread for printing lets me execute a line and print it while fetching/executing other commands.
- Threads
    - Again as I explained I create threads at the end of each command to get the output of the child process and print them without any concurrency problems in the output.
    - I used the thread library in c++ (https://en.cppreference.com/w/cpp/thread/thread) for threads.
    - I used the mutex library in c++ (https://en.cppreference.com/w/cpp/thread/mutex) for mutexes.