CS 307 PA 2 REPORT

Name and Surname: Görkem Yar ID: 27970

IMPORTANT NOTE!!!!: I used sabanci.flow to run my programming assignment with the command “g++ -lpthread -o cli cli.cpp -std=c++11”. Please, use this command and flow while grading my homework.

IMPORTANT NOTE2!!!: Since I used c++, execvp function can handle options such as -a without any additional parameter.

This report will examine the logic and reasoning behind my programming assignment.

In the first part of my code. I created the respective command.txt input file and parse.txt output file objects to do necessary operations after.

After that I started to reading from the file line by line. Firstly, I parse the line to find command, input, option redirection, filename and background functionalities. At this point, I am adding these keywords to parse.txt. Then, checking whether the command is wait or not.

Option-1: Command != Wait

Since command is not wait, I need to execute the command. To do that I will use execvp() function. Firstly, I am creating a pipe to pass data between child and parent processes. Then, I use fork() command to create child process. When the fork command executed, there are two branches in my program, one of them is child and the other one is parent process.

Child Process:

Child Process firstly checks for redirection. If the redirection is input redirection “<”, then child process will change the STDIN_FILENO with the input file and read the data from that file.

Then, It will create an args array for commands. After creating the array it will again look the redirection to check whether it is output redirection or not ”>”. If it is output redirection, by using dup2() function STDOUT_FILENO will be changed by the output file.

Finally, It will execute execvp() and terminate.

Parent Process:

Parent process has 3 options.

Option-1: If the process is not a background process and there is an output redirection:

Then we need to wait for the child process to die.

Option-2: If the process is not a background process and there is not any output redirection:

Then we need to create a thread to print results. And need to wait for thread to join. Since it is not a background process.

Option-3: If the process is a background process and there is not any output redirection:

Then we need to create a thread to print results. Also, we need to keep this thread id to join it later.

Option-4(Implicit option): If the process is a background process and there is an output redirection:

Do nothing since child process already redirected the output.

Option-1: Command == Wait

We already stored the thread Ids' of the running threads. Now, if there is a wait command, all of the running threads need to join and terminate.

Termination:

Like the wait function all of the running process need to be terminated then shell will terminate as well.

PrintPipe function (Function of threads):

It will get the read and of the pipe as an input. Then, lock the print statements by using the global mutex. It will print the results of the commands to the console. Then sync the console and read-end of the pipe. Finally, unlock the global mutex and terminate.

Global Mutex: It is used for blocking the threads when a thread is printing the output of a command. No, print interleaving due to this mutex.