CS307 PA2

Alpay Naçar

31133

Used one grace day for extension. Submitted at 12/05/2023, 23.00.

Arguments of Execvp: At first, I parsed each command one by one. For each line of command, created an array with 4 NULL to use it as arguments. The maximum elements I need to put inside arguments is 3 (command, input, option), and to pass it as arguments to execvp, last element should be NULL, therefore size 4 is enough.

Output redirectioning: In my run_command function, when there is output redirection, I didn't used any listener or mutexes because In my implementation I have only one mutex, and if I use that also in output redirections, commands that will output to terminal are going to wait unnecessarily. I used dup2 to change output direction in my child process. If there is no output redirectioning, I created a pipe for communication with process. Changed processes output to this pipe, so that our listener can get what this process sends.

Input redirectioning: In child process, if there is input redirectioning, I changed default input descriptor using dup2 function and replacing it with the given file.

Listener: I used listeners to handle communication between processes, and output synchronization. Every process which is going to output to terminal (not to a file), has a listener. When a listener can read something that a process wrote, It gets the lock of mutex (therefore no other listener thread can write while this listener is writing to terminal), and when the writing finishes, the listener thread unlocks the mutex. Therefore, output is synchronized. Listener cannot finish before the process it listens to ends.

Pipe Structure: For all listeners, I used different pipes.

Bookkeeping of process ids and threads: If there is output redirectioning, I only create a process and store its process id in a vector. If there is no output redirectioning, I will create a listener thread and store this thread in another vector.

Wait function: I wrote a wait_all function which waits until all processes and threads finish using vector of process ids and vector of threads. I used this when wait command is encountered and at the end of shell process, therefore shell process ends after every other process and thread finishes.

Non-Background commands: If a command is not a background command, I wait for that specific command to finish before moving on.