Ahmet Bilal Yildiz 27925

CS307 PA2

In the assignment PA2, I wrote code in C++ that reads the "command.txt" file line by line and simulates the process in which Unix Shell runs those lines (commands).

Program Flow:

In my program, first all the lines of the command.txt is read and then these lines are parsed and they are stored in a vector called parse, then parse.txt is created which contains all the commands with their parsed versions.

After the parse vector is created the the program continues with reading this vector line by line and do the necessary operations with respect to the commands.

If the command is not a wait command, and if the command has output redirection operator:

New child is created,

if the command do not have background operator (&):

Shell process waites the child and child process run the command and redirect the output to the desired file.

İf the command have background operator (&):

Shell process adds the id of the process to the background processes list (a vector of integers), and continues with the next line. Whenever wait command occurs, or before the shell terminates shell waits for all these background processes in the list.

Else if the command again is not a wait command, and if the command has not output redirector operator(if the output will be printed to the console):

New pipe is created which dynomically allocates memory from heap.

New child is created,

In the child process, necessary execution done and the output is written to the write-end of the pipe.

In the shell process,

İf the command has a background operator(&):

New threas is created and join immediately.

İf the command do not have a background operator(&):

New thread is created and added to the thread list. When wait command occurs, or before the shell terminates shell joins all these threads.

else (if the command is a wait command):

    Shell process waits for all background processes and threads.

Design Choices

Pipes:

If the process have output redirector since output is not directed to the console, pipe is not created and used, the output is directed to the desired file.

If the process do not have output redirector, then a pipe is created (so for all the commands which do not output director operator, there is a uniqe dynamically allocated pipe).

Threads:

If the process have output director operator, since output is not directed to console, thread is not created. For the background output director commands the waitpid() is used to ensure concurrency.

If the process do not have output redirector operator, a thread is created and join immediately if it is not a background thread. If the command is background operator, then the thread joins later when the wait command calls or before the program terminates.

Mutex:

There is a single mutex created, and it is given to the thread function as a reference parameter. It is locked before the critical section and unlocked after the critical section.

Bookkeeping:

For bookkeeping, as explained above:

My program uses waitpid() for the commands that has output redirectioning.

For the command that has not output redirectioning, my program uses join() for the threads and by using this it is also ensured that the processes which are related to thread is also finished.

Redirectioning:

For input and output redirectioning:

First the needed std is closed, then the desired file is opened with necessary command(read/write), and then the command is executed.