CS307 2022 SPRING PA2 – Synchronizing the CLI Simulator REPORT
UĞUR ÖZTUNÇ 28176

At first, global variables for mutex lock and pipes array are decleared. Mutex lock will be used in the void *display(void* args) function. As for the pipes array, in the main while loop, which the file is iterated line by line, after parsing part piping occurs with pipe(fds[no]) statement. This will create a unique pipe for every command line, as well as for every child process to communicate with parent process.

Before the main while loop; a thread array, an int array named threadStatus, which keeps the status of the thread corresponding for threads array (0 not created, 1 active, -1 created and joined), and an integer named thrCount for counting how many threads used in order to use in threads array.

In wait command part, first shell process waits for all children to be terminated with while(wait(NULL) != -1); and after that a for loop iterates in threadStatus array for checking which threads are active and running right now, and when it encounters one, waits for it to be terminated, and joins.

After forking:

In child process, dup2 function is used to referance PIPE[1] with STDOUT, (if redirection is not '>' of course).

In parent process, if redirection is not '>', the thread corresponding for thrCount in threads array is created for executing display function with the parameter no, which is the line counter declared just before the main loop starts. This integer will be used to access proper pipe file descriptor from pipes array (fds[]). After that, thrCount th thread in threadStatus array is set to 1 which means it is active now. And if current command is not background job, shell process waits for the child to be terminated, and if redirection is not '>' again, waits the thread to join and after it joins, sets the status of that thread to -1. At the end of parent process part, thrCount is incremented.

In *display function, first global mutex lock is locked. The argument passed to function, which is an integer addresses the proper pipe in pipes array, is casted to int and pipe is read. The output of command printed, of course with thread id before and after, fflush is used and mutex lock is unlocked.

After all stuff is done and while loop ends (file ends), the same operations in "wait" part is performed one time in order to wait all current processes and threads to be terminated before main function returns 0.