# CS 307 Programming Assigment - 2

İsmail Berat Düzenli

December 7, 2022

## 1 Parsing

I created 7 element array to organize commands in commands txt which is created from line read and sent to parse function.

1. Command
2. Input
3. Option
4. Redirection direction
5. Redirection File
6. Is background job
7. NULL

After parsing the array, it is given to the printParse() function that prints arguments to parse.txt in desired format.

While giving these arguments to execvp later, I loop on this array and only give input which have a value.

NOTE: As oppose what is claimed in assigment paper "-" character doesn't cause any unexpected behaviour, as a result I took no action. But "" characters are cleared from tokens.

## 2 Overall Flow of Program

After arguments are parsed if command is "wait" all processes started previously are waited and and their pipeline outputs are fed to different threads to be printed concurrently (but handled with locks) and these threads are also waited to join (due to wait command).

The way I bookkeep the processes are is a linked list. I created two linked list for processes and threads. whenever a process starts it is added to head of processes linked list, same for thread and whenever termination of a process is noticed then its node is deleted from the lists.

If command is not wait, command is added to commands linked list with piped file descriptors, arguments and process Id, and a child process started to execute command with execvp() function. If process is not a background process termination of the process and its output display thread is waited, otherwise not and moved on the next step. After one of these actions the processes in the linked list are checked to see if there is any process finished, if it is then its pipeline results are fed to a thread to be printed on console.

For synchronization threads are locked and unlocked before and after print statements

NOTE: if output is redirected to text file it does not fed its output to pipe but instead output of process is changed to file from standard output.

After all commands are read from the commands.txt all processes in the process linked list are waited and their results are sent to display thread.

After all processes terminated all threads are waited.

# 3 Questions in The Assigment paper

1. I used one pipe per process and keep them in a linked list with commands arguments and process id

2. if there is an output redirection to txt file pipe is not used instead of standard output replaced with txt file using close and open functions. if there is an input redirection same is done for input but pipe is used to give output to main process. if there is no redirection output directly given to the pipe and nothing done for input

3. threads are created after a processes termination is noticed, and locked by mutex at the start of thread and end to not allow any interruption in a specific console display