COMP 304 Project1: ShellGibi

Ayça Avcı, Ece Azizoğlu

15th March 2020

1 Completion

Part I and **Part II** work properly.
In **Part III**, there are some issues in autocomplete. Job management part is done partially. Custom commands and alarm work properly. However, we couldn't implement the psvis part.

2 Part I

We implemented execute() function to searched command file in 4 main source/binary files. They are /bin, /usr/bin, usr/local/bin and /sbin. We concatenated typed command to these 4 source/binary files and tried to execute it. If it could not be executed in one of the given source/binary file, we printed "No such command found". If command starts with ".", for example ./a.out, it should execute in the directory which is ./a.out located. To manage to do that, we get the current directory and concatenated with the executable file that starts with".". It works only if the user is in the directory that where executable file is located.

3 Part II

We implemented redirection and recursive piping functionality. To understand whether it is redirection command, first we checked whether the command includes at least one of the "¿", "¿", "¿¿". We used popen() to initiate pipe streams and execute the command. We used fopen() to open a file in read, write or append mode. Using fgets(), we read the characters from a given file stream source into an array of characters and writes them into a file using fprintf(). We printed output to the terminal using printf(). For recursive piping, we implemented execute_pipeline() command. To implement this, we used dup() and pipe(). If command-¿next has next (next is not NULL), we called execute(), otherwise execute_pipeline() function to make it recursive.

4 Part III

auto-complete: We are able to list all possible commands, newly introduces commands, and files that are in the current directory when the tab button is pressed. If there is one command possible command, it completes it and executes.

myjobs: We give a snapshot of a system's current processes by the current user.

pause: We pause the process given its process id.

alarm: We first parse user's command regarding the crontab storing format and edit the directory /var/spool/cron/crontabs accordingly, which executes the command in the system background for us at designated time.

wikipedia search: In our first custom command, if the user commands wiki, homepage of wikipedia opens in user's default browser. If they give another input besides wiki (i.e. wiki coronavirus), it searches for the second input in Wkipedia.

volume handling: In our second custom command, user is now able to change the volume levels without using settings. See the following commands for further explanation.
volume up: volume increases by 5%.
volume down: volume increases by 5%.
volume mute: mutes volume
volume unmute: unmutes volume to its 50% capacityy.