Metin Arda Oral & Tolgay Dülger

COMP 304 Project 1 Report

Part 1 - Basic Commands:

At part 1, we get all the paths defined in the linux by getenv("PATH") function. Then, we tokenize, concatenate the "/command" part and search all the paths. execv immidiately ends the child execution, thus no need to check for anything, we just try to execute the concatenated path. The code "`if(!command->background) wait(NULL);`" suffices to complete the background check and implementation, on the parent process.

Part 2 - Custom Commands:

Filesearch:

Recursive filesearch uses a recursive function, `filesearch(char* dir, char* search,char* rpath),` which checks whether the given search name is contained in the files inside the given directory, and also for every file, if it is a folder, it recursively calls filesearch on that folder, too. Normal filesearch is very simple, we just iterate and compare the file's name with search argument with the strstr function. Filesearch -o is also very simple, the only difference between normal filesearch and filesearch -o is that we use xdg-open command to open matching file(s) in filearch -o. For every matching file, we call xdg-open with every matching file's path. We were not able to call xdg-open directly with execv function, so we used system() function to be able to execute it.

Joker:

We create a txt file which contains the required cron "`fprintf(joketab, "%s", "*/15 * * * * XDG_RUNTIME_DIR=/run/user/$(id -u) notify-send \"HEY A JOKE!\" \"$(curl --silent https://icanhazdadjoke.com/ | cat)\"\n");`" and then call crontab executable file with execvp, and give a txt file argument to directly add the argument to the crontab. Finding the correct cron was quite a pain. Brute force path calculation did not work for other computers, we had to use `XDG_RUNTIME_DIR=/run/user/$(id -u)` to successfully execute the command. We took this idea from stackexchange, before it was discussed in the discussion forum.

Take:

Take iterates through given input (path) with strtok with the delimiter '/' . While iterating, we create directory with mkdir if it does not exist and change the current working directory to the newly created directory with chdir function.

Cdh:

For cdh, we were not able to use directory stack by executing pushd, popd and dirs function, we do not know why. These functions did not work in our shell. So, we stored the recent directories in a file to be able to store them across different shells. After every 'cd' call, we store the current working directory to our 'history' file. When 'cdh' executed we read the history file and store the directories into a string array. After that, we list recent directories and wait for the user input (integer or char) and change the current working directory with chdir().

Our Creative Commands:

Storyteller (Arda):

I basically wrote a short interactive story game, which is inspired from the movie Oxygen. Instead of going for some useful command, I thought that it would be fun to implement a gamewise command. Code of the command is straightforward, I just used several prints and some while loops. It was possible to write the story into a file and then write a cleaner code, however, for the length of the story, I did not think that it was necessary to do so. There will be a significant difficulty increase on implementing the loops, too.

Project Repository: https://github.com/ardaqwe35/COMP-304-Project-1