CSE3033 PROJECT-2

Fatih BAŞ-150117048

Ahmet Süleyman Sönmez-150117010

Alperen KÖKER-150117035

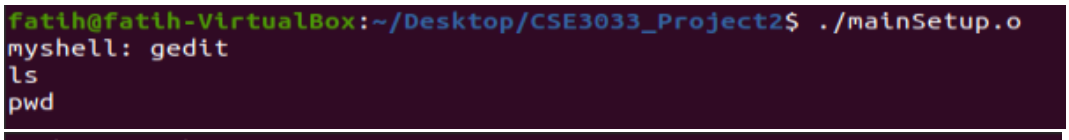
Our goal is execute write basic shell in c in this program.

- Execute every executable stored in variable PATH in foreground or back gorund.
- Implement some command list:
    1. Ps_all
    2. Search
    3. Bookmark
    4. Handle control Z signal for foreground process
    5. Exit
    6. Redirections (<,>,>>,2>,<>)

Part-A)

We used the fork() method to create a new child process. If the child process cannot be created successfully our program print error message with stderr.If the child process can be created succesfully it finds the path of the program to be executed using the findPath() and execute program. If user write program name with & program execute in background and can't block our terminal and it is added to the background process list with the appendRunning() started index 1. If all background process closed after open new process again started index 1 with controlExit(). I user write program name without & program execute in foreground and block our terminal so parent process wait the finish child process(wait() method).

EXAMPLE-OUTPUT:



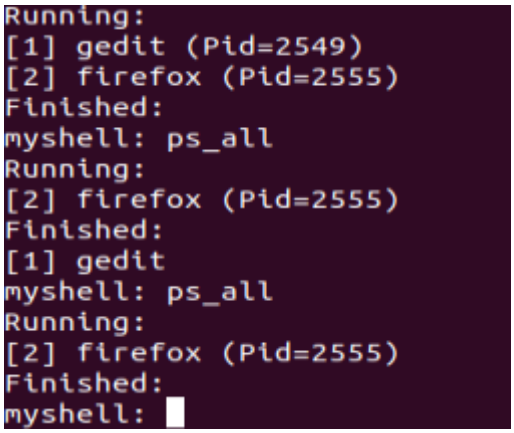
Part-B)

Ps_All

This command print all background process on terminal screen in the following Running: and Finished: format. Finished process name print only one time on the screen.

Our command work like this -> While the program is running, it constantly checks status using the waitpid(). When this command execute, print all background process after closed background process delete from list with deleteNodeBackgroundProcess().

EXAMPLE-OUTPUT:

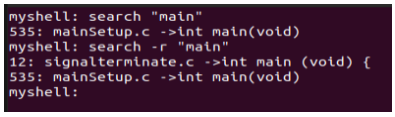


Search

This command find desired word in the files and print line number, file name and line.

Our command work like this -> Firstly find current path and check word type is valid or not. After check argument 1 if argument 1 is -r, travel all files in current directory and subdirectory with listFilesRecusively() and print desired word's line number, file name and line with indexOf().if argument 1 is word, travel all files only in the current directory with listFiles() and print desired word's line number, file name and line with indexOf().

EXAMPLE-OUTPUT:



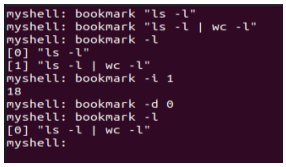
Bookmark

This command is like our dictonary. We can add, delete, execute and list commands.

Our command work like this -> Program checks argument 1. If argument 1 is -l, print all added frequent commands on the terminal screen with printList(). If argument 1 is -d, first checks argument 2 and delete frequent command in index which coming argument 2 with

deleteNode(). If argument 1 is -i, firstly find frequent command in desired index with findSpesificIndex() and call substr() for throw “. Lastly execute command with system(). If argument 1 is frequent command, firstly checks command type valid or not if it is valid add command in the list with append().

EXAMPLE-OUTPUT:



Control Z signal

This command handle control Z signal and stop foreground process with childs.

Our command work like this -> Firstly created signal struct and initialize signal handler(controlZ) and flag. After program initialize and checked signal action and and set. If signal not initialize succesfull program write error message. When send control Z signal, handle it and check foreground process is Running or not. If foreground process is running stop it with childs, If it is not running, signal can’t effect program.

Exit

This command close our process.

Our command work like this -> Firstly checked any background processes is running or not with controlExit(). If any background is not running, our program close. If any background is running print message on the terminal screen.

EXAMPLE-OUTPUT:

Part-C)

Checks input, output files and arguments are valid or not. If they are not valid write error message on terminal screen.

>

This command write output in desired output file.

Our command work like this -> execute input command with execv() and write standart output in output file. If output file is not exist, program open new and write it. If file is exist, program truncated it.

>>

This command write output in desired output file.

Our command work like this -> -> execute input command with execv() and write standart output in output file it. If output file is not exist, program open new and write. If file is exist, program append it.

<

This command read input in desired input file.

Our command work like this -> read standart input from input file and execute with execv(). After print output on terminal screen.

2>

This command write standart error in desired output file.

Our command work like this -> execute input command with execv(). If command is valid, print output on terminal screen. If command is invalid, write standart error in output file. If output file is not exist, program open new and write. If file is exist, program truncated it.

<>

This command read input in desired input file and write output in desired output file.

Our command work like this -> read standart input from input file and execute with execv(). After write standart output in output file. If output file is not exist, program open new and write it. If file is exist, program truncated it.