PROJECT – 2
<<Terminal>>

Overview :

The purpose of project 2 is designing our own terminal which runs on UNIX platform. C language is used to create this shell. This shell has its own unique operations as well as other ordinary operations. All bugs are tried to be handled and all errors are printed in stderr.

Our shell supports all of these features :

- Running program ,which is already loaded into our system, as foreground and background process.
- Listing all the background processes by using “ps_all” command.
- Stopping all the foreground processes and its children by using “^Z”.
- Searching specific keyword by using “search “ command.
- Keeping frequently used commands by using “bookmark “ command.
- Shutting down our shell by using “exit” command.
- I/O redirections :
    1.) “<” this is for getting input from external file.
    2.) “>” this is for printing output to external file.
    3.) “>>” this is for printing output to external file ‘append’.
    4.) “2>” this is for printing errors to external file.
    5.) “2>>” this is for printing errors to external file. ‘append’

These are the detailed explanations of each command :

Part A :

The shell is needed to support to run program which is already loaded into our system like gedit , ls , xterm and so on. In our shell , this is handled by creating new child by using fork() function and running corresponding program by using execv() function . Firstly, path of that program is found by using “findpathof()” function . This function takes 2 parameter . These are “path” and “exe”. If the path of program is determined , then path is loaded into “path” variable. Then this path is given to the execv() function.

If user enters “&” sign at the end , then program should be run in the background . If background variable is 1, our child is run in the background thanks to waitpid() function . This function is called at each new child creation because of signal handler.

Example Usage –> myshell : gedit
–> myshell : gedit &

Part B :

The shell is needed to support extra commands which are not already loaded into our system. Their internal design is written by coder. These commands are “ps_all”, “search” , “bookmark” and “exit” . The shell also should be able to catch “^Z” signal . There are the details explanations of each command :

ps_all :

The aim of this command is listing all the backgrounds processes as “running” and “finished”. Whenever new background process is created , this process is added into “linkedlist” structure. If “ps_all” command is entered , our shell prints all of them . While printing operation , status of each child is checked by using waitpid() and child is printed according to that information as “running” or “finished”.

In the first call of this command , all dead processes are listed whereas in the second call of this command , they are not printed.

Example Usage –> myshell : ps_all

search :

The aim of this command is searching specific pattern in the current directory or the all subdirectories. This is handled by getting help from “grep” program which is already loaded into our program. Thanks to grep program , our shell catchs pattern in the correct file . Output of the grep is formatted and printed in the correct order.

This command also supports “-r” option to traverse all the subdirectories to find the corresponding pattern. This is handled by simple recursive function and again grep program is used.

Pattern should be given in the double quotes. This is important.

Example Usage –> myshell : search “main”
–> myshell : search -r “main”

bookmark :

The aim of this command is to keep the frequently used commands as a list. Name of the each program is added to a list . Linkedlist is used as a list. If user enters the unknown program name , this will not be added into bookmark list. There command has different options. These are “-h” , “-l” , “-d” and “-i”.

“-h” option is used to list the usage of bookmark command.
“-l” option is used to list the all programs in the bookmark list.
“-d” option is used to delete item from bookmark list.
“-i” option is used to run the program in the corresponding item.

Example Usage –> myshell : bookmark -h
–> myshell : bookmark “ls -la”
–> myshell : bookmark -l
–> myshell : bookmark -d <index>
–> myshell : bookmark -i <index>

exit :

The aim of this command is to shut down our shell. If there is not background process at all , exit command simply close our shell. Otherwise , error message is printed and our shell keeps running.

Example Usage –> myshell : exit

^Z :

The aim of this signal is to stop foreground process and its all childeren . signal(SIGTSTOP,funcName) is used to catch that signal. Thanks to “ps” command , all the pids of processes running in our system are found and stopped if it is child of our foreground process. This part is handled recursively.

If there is no foreground process , then the signal does nothing.

Part C :

The shell is needed to support I/O redirections. This is the purpose of that part. Our shell supports 5 different symbol to make I/O redirections. There are “>” , “>>” , “2>” , “2>>” and “<” . When user enters his/her command , shell looks into that command. If all of the symbols are placed correctly with correct number of arguments in the correct order , then operation type , input file name and output file name is assigned to the corresponding variables. After that , shell cut the command and take just program name with arguments. Then shell runs the program same as part A . Before running execv() function in the child part , shell checks if there will be any redirections or not. Redirection flags are used in that part. According to the situation , inputRedirection() - outputRedirection() functions are called .

In the inputRedirection() function , open() function is used with the inputFileName and corresponding flags . If file is opened , then dup2() function is used to make redirection .

In the outputRedirection() function , type of the output handling is determined by formatOutputSymbol() function. According to the output symbol type , flags are changed.

1- >

CREATE_FLAGS and CREATE_MODES are used for that symbol because new file is needed to be created for printing outputs. In the dup2() function , STDOUT_FILENO is used.

2- >>

APPEND_FLAGS and CREATE_MODES are used for that symbol because output is needed to append to existing file for printing outputs. In the dup2() function , STDOUT_FILENO is used.

3- 2>

CREATE_FLAGS and CREATE_MODES are used for that symbol because new file is needed to be created for printing errors. In the dup2() function , STDERR_FILENO is used.

4- 2>>

APPEND_FLAGS and CREATE_MODES are used for that symbol because new file is needed to be created for printing errors. In the dup2() function , STDERR_FILENO is used.

5- <

READ_FLAGS and READ_MODES are used for that symbol because input should be got from that input file to run program . In the dup2() function , STDIN_FILENO is used.

History Part :

In that part, we are expected to show previous commands in the screen by using arrow keys.

Linkedlist implementation of this part was ready. But we could not manage to detect pressed keys. As we understood from all researches , “termios” , “ncurses” and “curses” libraries can be used to detect pressed key. “getch()” function is used in ncurses to detect key whereas terminal mode is changed in the termios library .These are library dependant solution methods. Another solution method is that file descriptor can be used to get stdin info and this can be used to detect the key.

All of them has its own problems. So the history part is not completed . Our efforts on that part are added into the zip file as screenshots.

As an Extra :

→ Usage of bookmark program can be seen by typing “bookmark -h”
→ Usage of redirections can be seen by typing “io -h”

Problems & Solutions :

These are the most complicated parts and solutions :

- Main part runs for either parent and child. This causes call of setup() function twice

Solution : We changed the condition of while loop as a pid of parent process.

- Detecting if the background process is still alive or not

Solution : Return value of waitpid() is used to handle that problem.

- Killing all the children and relation of foreground process with ^Z

Solution : Thanks to output of ps program and a recursive function , we handled that problem.

- Formatting output of search command

Solution : We divided all the components of grep output and then brought together again.

Github : https://github.com/MrYunusEmre/CSE3033_Project_2_Terminal

--Engineers –
Yunus Emre Ertunç
Muhammed Enes Aktürk