# COMP304 – PROJECT 1

*Murat Güç – 68967* *Yakup Can Karacaoğlu - 69063*

* 1. **Background**

For running each command, shell creates a child process and run it. In the regular shell it waits the command process output. By adding “&” at the end of the command force it run in background. In the shellax-skelaton code, parse\_command function has already handled detecting background process. Briefly, if there is a “&” at the end of the command, it changes boolean variable of the background to true in the command struct. So, in the process\_command function, child process handles the waiting. As mentioned before, if it is not a background process, we must wait it to finish. To wait specific pid, waitpid() function is used.



Figure

In the first line of Figure 1, sleep 10 runs in the background. While it is processing, 2nd line operates sleep 1 and it waits 1 second.

* 1. **execv**



Figure

The instruction expects us to change execvp to execv functions to run commands. Execvp that is implemented before in skeleton code takes the input of command name and arguments. After that, it automatically searches the command in the environment. As can be seen in Figure 2 above, my computers PATH variable includes many env paths splitted by the “:”. Instead of using execvp, implementing execv is assigned. This function only takes the argument of command’s specific path and the command arguments. So, we should iterate over PATH variable by tokenizing them. It basically searches for the command in the local paths until it successfully compiled. We illustrated the paths looked for a command in the Figure 3.

Text

Description automatically generated

Figure

**2.1 Redirection**

In that part of the project, the implementation of the input-output redirection for the commands is completed. There are 3 types of the redirections such as overwrite (“>”), append (”>>”) and the input (“<”). The overwrite operation redirects the standard output of a command from console to a specific file. If the file does not exist, it also creates a file with the given name. The append operator does similar thing with the overwrite, but they have a sharp difference. The overwrite operator deletes the content of the file before writing its own content, append adds its content at the end of the pre-existed content.

The implementations of these operators are completed with the dup2 function of the c language. During the parsing phase, the parse\_command function detects the existence of one of these operators in the given command line, then it modifies the redirects array of the command. In the process\_command function we evaluated the redirect array, and the necessary redirections are done in there. Basically, if the first element of redirects array has a value, which is a file name, the code closes the standard output with “close(1)”. Then, open a file with the given file name and redirects the output to this file with “dup2(filename,1). It also works in the same manner if the second element of the redirects array is not null. The only difference is that the opening flags.

Lastly, the third element of the redirects array can be not null. It means there is an input redirection. The standard input of the command is closed with the line close(0), and the given file is redirected into the command with the line dup2(filename, 0). These are the basic principles of the redirection in the shellax.

Text

Description automatically generated

Figure

**2.2 Piping**

The piping implementation is similar to the redirection. The piping operator “|” is also checked by the parse\_command function, and the following command is put into the “next field” of the current command. By checking the existence of the next in a command, the code calls the pipe\_execute function. As mentioned before, the piping is also based on the redirections of the inputs and outputs. The methodology is creating a pipe, connecting the output of the first command into the writing end of the pipe, and connecting the input of the second command to reading end of the pipe.

In the implementation, firstly a pipe named “fd” is created. Then the first fork is called for the execution of the first command. In the child, which is created by the fork, the mentioned redirections are completed and execv command is called with the necessary arguments. The first child completed its implementation and called the exit to be killed. The main process waits for the child to completes its execution during that time. Secondly, the second fork is called for the execution of the next command. Then, the redirection from reading end of the pipe to the standard input is completed. Lastly, a recursive call is occurred in the second child. The process\_command function is called for the second command here, and the rest are the regular operations for a command.

Text

Description automatically generated

Figure

**3.1 Uniq (Named as myuniq)**

The uniq command is responsible for decreasing the number of elements to one if they exist more than once in the given list. It also has -c and “- -count” flags to return the elements with the number of occurrences. The implementation is fundamental. It takes the input using the fgets method and keeps them into an initial array. Then, it goes over the initial list and puts the unique elements into in a different array. If the flags are not given, it directly prints the unique array. If they are given, we go over the initial array again and check the number of occurrences and print them. The important side note for the uniq command is that it is named as “myuniq” to prevent confusion between the original uniq command and our implementation.

A picture containing text

Description automatically generated

Figure

**3.2 Chatroom**

In shellax cli, user gives input consisting of room name and username to the chatroom program. For instance, “chatroom comp304 murat”. First, program ought to construct room in the /tmp/ folder. For creating another folder in the /tmp/, we must be user. So, “sudo su” command makes us to authorize ourselves in the shell right before the initiating the shellax cli. Then, program makes directory in terms of the path “/tmp/chatroom-ROOMNAME”. If it already exists, program handles that. After creating directory successfully, program creates a child process. In the parent process, named pipe of the registered user is constituted thanks to the mkfifo() function, then program keeps reading this fifo to handle whether a message is gotten or not. When the other users write a message in the chatroom from another shellax, it will be written by iterating over the room folder and writing them into their named pipes. So, duty of the child process is to write messages to the others by iterating named pipes. In the Figure 7, the example dialog is given.

**3.3 Wiseman**

Wiseman function is a basic command to understand and use the crontab of the Linux. The crontab ensures that to execute a command repeatedly in given time intervals. For this application, wiseman command takes a minute value between the 0-59 and it executes the “fortune | espeak” command in each time intervals. The fortune returns a random quote and espeak command vocalizes the given quote.

In the implementation, the wiseman function creates a temporary file to hold the crontab commands and write the “\*/m\* \* \* \* fortune | espeak” line into the temporary file. The “m” is the placeholder for the given minute value. Then, the crontab command is executed with name of the temporary file as argument. The rest will be handled by the crontab, it gives enlightening quotes every given m minutes.

Text

Description automatically generated

Figure 8

**3.4.1 Reflex**

It is the first of our custom commands, it is a test to measure your reflection time. When you enter the reflex command, the introduction page welcomes you and it says, “press enter to start”. After you press the enter, you should wait to see a cat in the screen. When the cat shows up, you should press the enter as soon as possible. In that way we can measure your reflection time and represent it in the end.

For the implementation, the fidget command should be installed in the system. The code represents the first page after execution and waits until the user presses the enter. After someone presses the enter, it gets a random number to determine its sleep time and goes to sleep. Then the sleep is ends and the cat shows up. In that time the user should the press enter again and the system measures the exact time in millisecond during the operation.

Text

Description automatically generated

Figure 9

**3.4.2 Vigenère**

Vigenère cipher method is used for encrypting messages by summing characters of the plaintext and key in terms of the alphabet. It repeatedly uses keys for encrypting message. For instance,

ALACARTE – plaintext (there shouldn’t be any blank character.)

ABCABCAB – key (key is AT, but for fitting it is repeated)

BNDDCUUF – cipher (decryption is implemented in a same way but for this time, we subtract).

This command takes 3 arguments, first one is mode (enc/dec), if user passes another string another than enc/dec, program indicates that. Second argument is plaintext for enc, or ciphertext for dec. It must not contain blank character, but it may include other characters. Lastly, third argument is the key which cannot be longer than the second argument. In Figure 10, encryption and decryption of the arbitrary message is displayed.

