Ege Berk Süzgen
53714
Emir Atışay
53590

COMP 304 SHELLDON: PROJECT 1

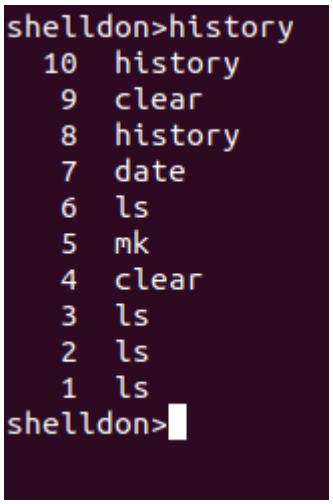
Part 1-

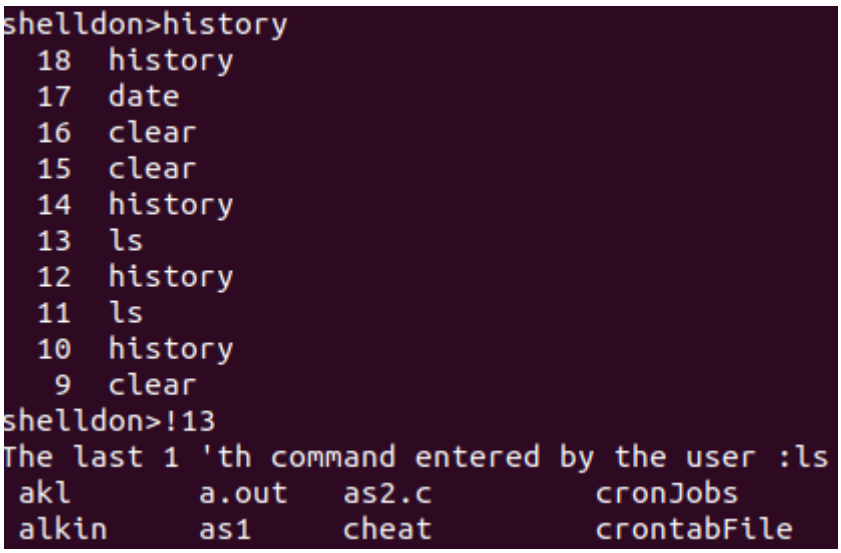
In this part of the project, we implemented a child-parent structure as we learned in the lecture. For each process demanded by the user, parent only forks a child and does nothing until child finishes the process. For the background processes, the only difference is parent waits. Also, child executes functions with execv() function which takes the command's directory as the first input and an array which holds what user entered.

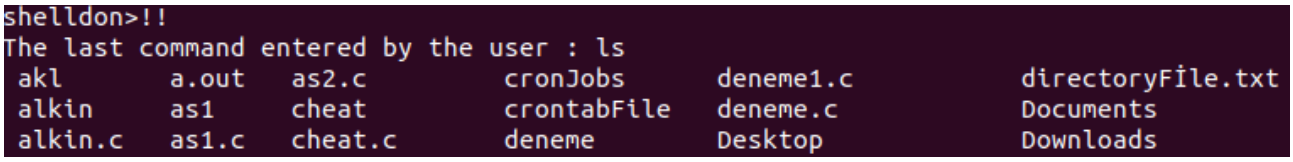
Part 2-

In the I/O redirection part, operating a executing a command and holding in outfile was requested with two modes. One mode indicates that shelldon should create the output file and refresh it with new outputs if it exists. Respectively, other mode appends the output file if necessary. To do this, we open the file and call for dup2 method to track every footsteps of the child. And also, the given process is executed with execvp command. The difference between the modes are defined by the opening type of the file, O_TRUNC and O_APPEND, respectively.

Then in the second case we implemented a function called history which returns the last 10 commands entered by the user. There is a global array which stores all commands entered by the user except '!' ones. '!' commands task is to execute a function in the history array according to the second character of the '!' function.

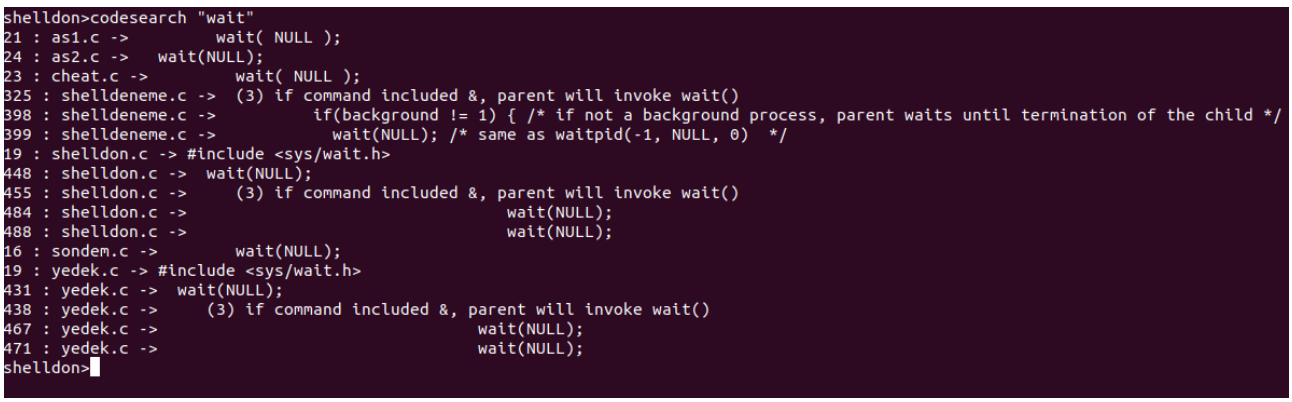






Part 3-

In this part, we first tried to implement a new function called “codesearch”. Basically, codesearch has 3 commands, which enable user to either search in the whole source, or in specific file or all subdirectories. First, we implement targeted search with specific file by user. That is because we had a file searcher method and all the other parts are based on this function. Then, in the directory search, we fork another child and this child holds the corresponding subfiles of the current directory. By parsing the output file of that child, now we would be able to search on the proper files, such as files with c, cpp or h extension. However, in the recursive function part, things did not go well for us. We partially implemented and decided not to include in our submission. On the other hand, in the recursive search, we basically iterate through the output file that is mentioned and if that file is directory, it would call itself again. However, if it is proper file to search, it will again act as directory search and call for that method.



```
shelldon>codesearch "wait" -f shelldon.c
19 : shelldon.c -> #include <sys/wait.h>
448 : shelldon.c ->  wait(NULL);
455 : shelldon.c ->     (3) if command included &, parent will invoke wait()
484 : shelldon.c ->                                  wait(NULL);
488 : shelldon.c ->                                  wait(NULL);
shelldon>
```

The second function is “birdakika” which takes time and music file from the user. At first, we parsed the args[1] because user enters the time like 23.45. However, we need it as 45 23. After the parse operation, we opened a new custom file for crontab -l because we are not allowed modify the default directory of crontab file. With the echo function,we wrote it to the crontab file. Then to terminate it after 1 minute, we followed the same process just incremented the minute part of the input and called the “pkill” function.

```
56 17  * * * mpg321 /home/kingest/music.mp3
57 17  * * * pkill mpg321
```

The third function is custom made. We created a basic calculator which supports summation, division, multiplication and subtraction.

```
shelldon>calc
Please enter the operator = *
Please enter the first operand = 23
Please enter the second operator = 67
The result is 1541.000000
shelldon>
```

Part 4-

In the last part of the project, we added an extra function to the kernel module which simply does depth first search, because the oldest child of the each process is the first entered one. At first we find the process (according to the code it is task) which matches with the input PID and then put it into the dfs function to display its oldest children till the leaf node.