Comp304 Project 1 Report

Mehmet Üstek - 64782
Irmak Ilgaz - 64626

Part 1

Part 1 implementation with execv searches the related files (bin file) on the given command. If the given command is matched with any command that is in the bin file, it is executed.

Part 2

Shortdir.h file is responsible for the implementation.
The code creates a directories.txt file in the home directory of the user. For every new short directory association, it appends the new association to the file. The rest of the commands are implemented based on the file manipulation.

Part 3

highl.h file is responsible for the implementation.
The file is scanned line by line. If that line contains the desired string, then it appends it into the to-be printed values. A “flag” variable is responsible for these checks. The code highlights the desired string.

Part 4

good_morning.h file is responsible for the implementation.
The code creates a file named crontab.txt and this file overwrites the crontab file stored in the user directory.

Part 5

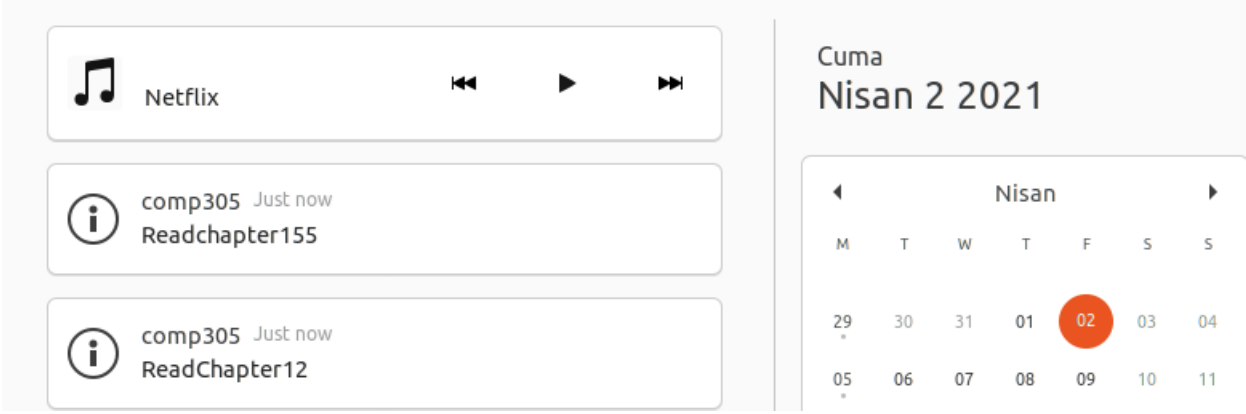
kdiff.h file is responsible for the implementation.
Normal implementation with -a, compares two files and gives output with only file manipulation.
The implementation with -b, uses fseek and ftell() functions which are I/O operations.
SEEK_END variable returns the all bytes restored in the file and comparisons between those two files are made at the end.

Part 6

todo.h file is responsible for the implementation.

(There are sometimes errors that occur when commands ran consecutively, in that case we need to exit seashell, but the data and changes are secured.)

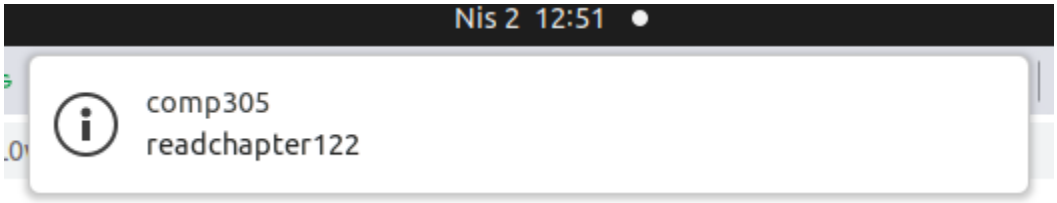
The description of this part:
We created a todo list, which is similar to what we are using in our daily life. It is fractioned into projects. Inside that projects, you can list, delete and set an alarm for that event. The lists are all sorted with bubble-sort using a linked list. As such, all todos will be sorted based on their dates before prompting it to the user. The commands for these operations are given below;
Commands:

- Todo add <project-name> <todo-description> <todo-duetime>
    - It adds the project name, description and due time to the todo.txt, which will be used in the other operations, such as listproject, listall and del. It also creates appends to crontab.txt file and executes a child process that will send a notification in every minute when the due date has came. For example;
    - 

    - As you can see, comp305 notification is reached.
    - (However, there is a slight issue, for a command to work in crontab, it needs to be assigned to times, as such;
    - ```
mehmet@mehmet-X580VD:/home/mehmet/comp304/Project1 seashell$ todo add comp305 readchapter122 02.04
mehmet@mehmet-X580VD:/home/mehmet/comp304/Project1 seashell$ todo add comp305 readchapter122 02.04
```
    - This problem probably stems from the memory allocation issue. However, we could not solve it. So it is best to enter the argument two times in that sense.
    - Also, the description of the todo must be typed without white spaces. As such;
        - ReadChapter12
        - **Not** Read Chapter 12
        - Since, the parser understands the 2nd argument as another argument of the command.
- Todo del <todo-description>
    - It deletes the todo that matches that description.
- Todo delproject <project-name>
    - It deletes the entire todos assigned to that project.
- Todo listproject <project-name>
    - It sorts and displays all todos related to that project name.
- Todo listall
```

- It sorts and displays all todos.

Another crontab notify-send visual which operates every minute on that day;



For this part, we used queue and linked list data structures.
To add todos, we used a queue.
To list, sort and display the todos we used linkedlist.
Both queue and linked list implementations are reused from the below links;
In that sense, we acknowledge using another's code and accept the code of conduct. The codes are slightly changed regardings some parameters. The intention of using an open source code in here to move faster. No intention of plagiarism is shown, as sourced the materials we received from open source articles or sites. We acknowledge that this project has sole purpose of learning materials in operating systems, hence we did not use any other code that has been retrieved from the web other than queue and linked list data structures.
The code for queue and linked list are retrieved from:
https://www.geeksforgeeks.org/c-program-bubble-sort-linked-list/
https://www.learn-c.org/en/Linked_lists
https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/

And used in queue.h and linkedlist.h
These links and acknowledgements are also stated in the code.