COMP 304 Shellfyre: Project 1

Alp Özaslan - aozaslan18, Onur Eren Arpacı - oarpaci18

In this project we modified some of the shell commands and created new commands for Linux.

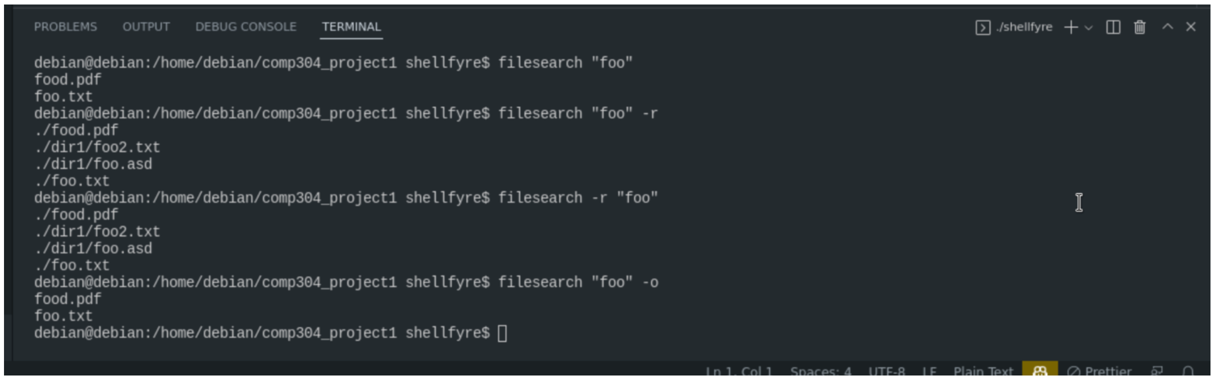
Part 1:

In this part since we didn't use execvp we needed to find and construct the full path of the commands. To do this we first check if the command starts with "./" which means it is an executable file in the current working directory. If it does not start with "./" then we get the PATH environment and try to find the command in the path and then we execute the command. Meanwhile the parent process waits if the command->background is false.

Part 2:

FILESEARH

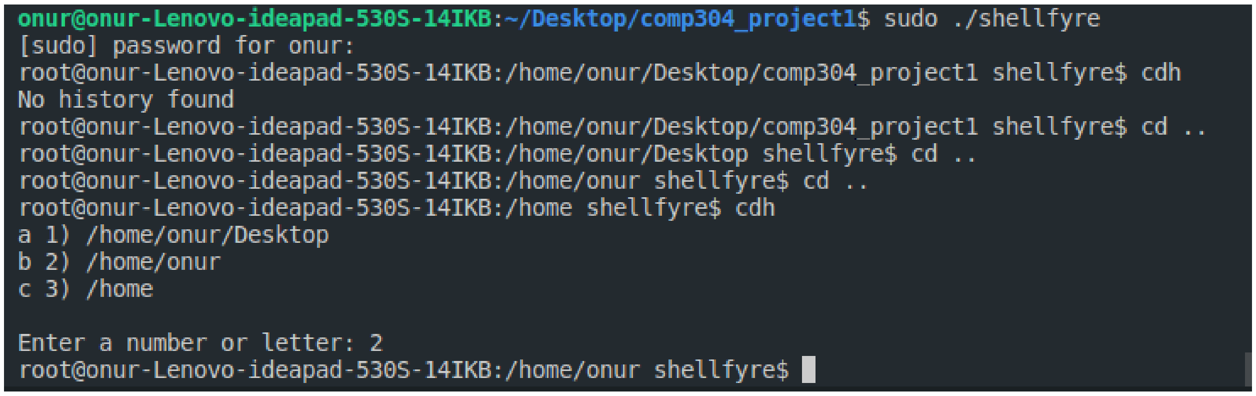
Using this command, the user can find and if s/he wants to open the file(s) using keyword. In the code, first we checked if there are any arguments (-r to look inside directories and -o to open the files that were found.) Then we look at file names using the "readdir" command if they include the keyword or not. After that If the -r argument is given we go to subdirectories and so on. After finding all the files that include the keyword, we list them and if the -o argument is given we open them using the "xdg-open" command.



```
debian@debian:/home/debian/comp304_project1 shellfyre$ filesearch "foo"
food.pdf
foo.txt
debian@debian:/home/debian/comp304_project1 shellfyre$ filesearch "foo" -r
./food.pdf
./dir1/foo2.txt
./dir1/foo.asd
./foo.txt
debian@debian:/home/debian/comp304_project1 shellfyre$ filesearch -r "foo"
./food.pdf
./dir1/foo2.txt
./dir1/foo.asd
./foo.txt
debian@debian:/home/debian/comp304_project1 shellfyre$ filesearch "foo" -o
food.pdf
foo.txt
debian@debian:/home/debian/comp304_project1 shellfyre$ 
```

CDH:

We implemented an "add_directory_to_history" function to keep a FIFO queue in a file. We call this function whenever the user changes directory, it adds the current directory to the file. When the user enters the cdh command, we open the history file, read the contents into an array and print them as choices to the console. When the user makes a choice we modify the command->name to cd and add an argument for the selected path. Then we call the process_command function with the modified command.

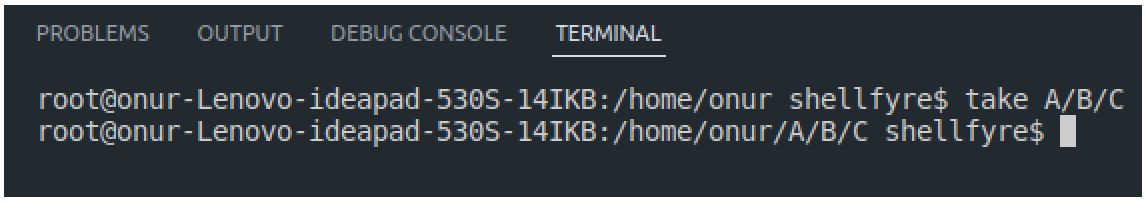


```
onur@onur-Lenovo-ideapad-530S-14IKB:~/Desktop/comp304_project1$ sudo ./shellfyre
[sudo] password for onur:
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop/comp304_project1 shellfyre$ cdh
No history found
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop/comp304_project1 shellfyre$ cd ..
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop shellfyre$ cd ..
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur shellfyre$ cd ..
root@onur-Lenovo-ideapad-530S-14IKB:/home shellfyre$ cdh
a 1) /home/onur/Desktop
b 2) /home/onur
c 3) /home

Enter a number or letter: 2
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur shellfyre$
```

TAKE:

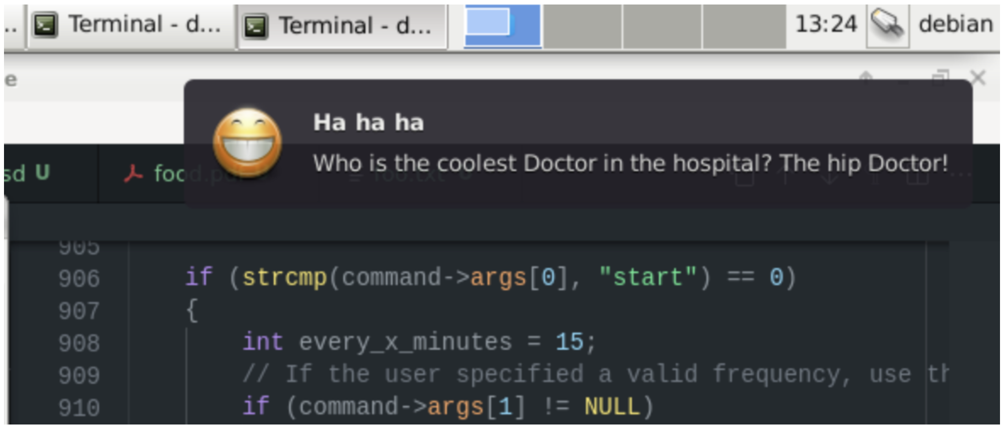
To implement take, we first run the mkdir command with the -p option and the given directory from the user, in a child process. We then modify the command->name to be cd and we then call the process_command function with the modified command.



```
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur shellfyre$ take A/B/C
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/A/B/C shellfyre$
```

Joker:

Using this command the user can start fetching and displaying hilarious jokes from https://icanhazdadjoke.com. Once the joker starts the run it does not stop until the user stops it manually. To implement this we used crontab, curl and notify-send commands. When the user calls the “joker start” command, the code creates an sh file named as joker.sh that includes fetching and notifying commands. After that this file's path is added to the crontab file. And cron runs the joker.sh at specified intervals, 15 minutes unless the user indicates otherwise. The user can specify the time interval using the “joker start [time_interval]” command. When the user is bored with the jokes, can end the jokes using the “joker stop” command. To stop the joker, the code replaces the line in the crontab with empty string.



Terminal - d... Terminal - d... 13:24 debian

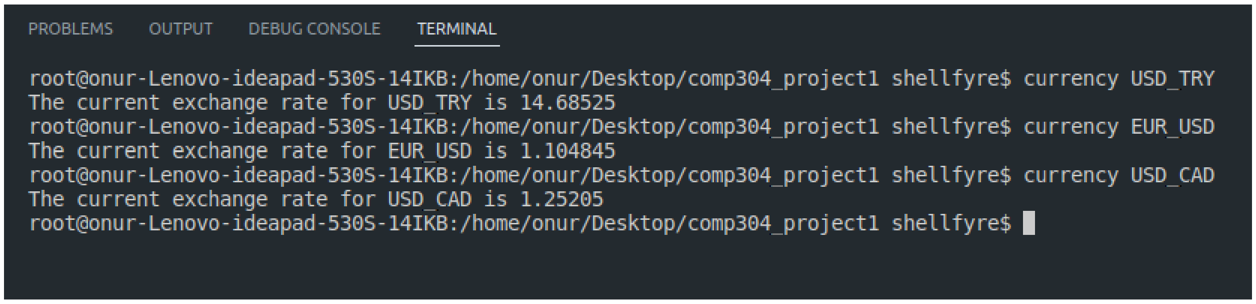
Ha ha ha

Who is the coolest Doctor in the hospital? The hip Doctor!

```
    if (strcmp(command->args[0], "start") == 0)
    {
        int every_x_minutes = 15;
        // If the user specified a valid frequency, use th
        if (command->args[1] != NULL)
```

CURRENCY:

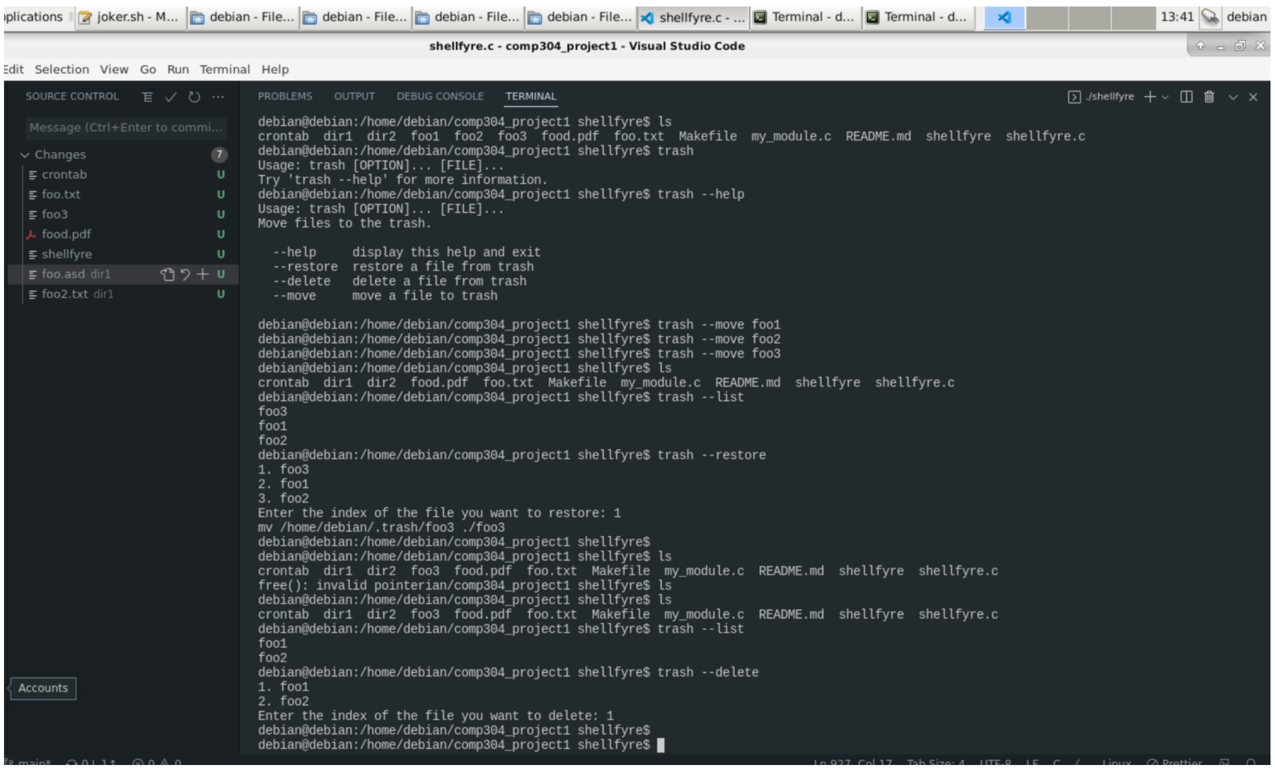
I (Onur) implemented a currency function to retrieve current currency rate information from the command line. I used the "free.currconv.com" api to implement this. I first construct the curl command string with the link, currency pair argument received from the user, and my API key. Then I open a pipe using the popen function to retrieve the result of the curl request. Then I parse the json string to get the currency rate and finally print it to the console.



```
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop/comp304_project1 shellfyre$ currency USD_TRY
The current exchange rate for USD_TRY is 14.68525
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop/comp304_project1 shellfyre$ currency EUR_USD
The current exchange rate for EUR_USD is 1.104845
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop/comp304_project1 shellfyre$ currency USD_CAD
The current exchange rate for USD_CAD is 1.25205
root@onur-Lenovo-ideapad-530S-14IKB:/home/onur/Desktop/comp304_project1 shellfyre$ 
```

TRASH:

I (Alp) implemented a trash command as an alternative to native rm command. This command, unlike the rm, does not permanently delete the selected file but move it to ./trash directory. Users can list the files in the trash, and if s/he wants s/he can restore from the trash to the current directory or delete it permanently. Also they can empty the trash (that deletes all files in the trash folder). I coded it in a modular way with a main function (trash(command)) and helper functions (move_to_trash(), delete_from_trash(), empty_trash(), list_trash()) And I added a kind of helper page to the function to tell how to use it.



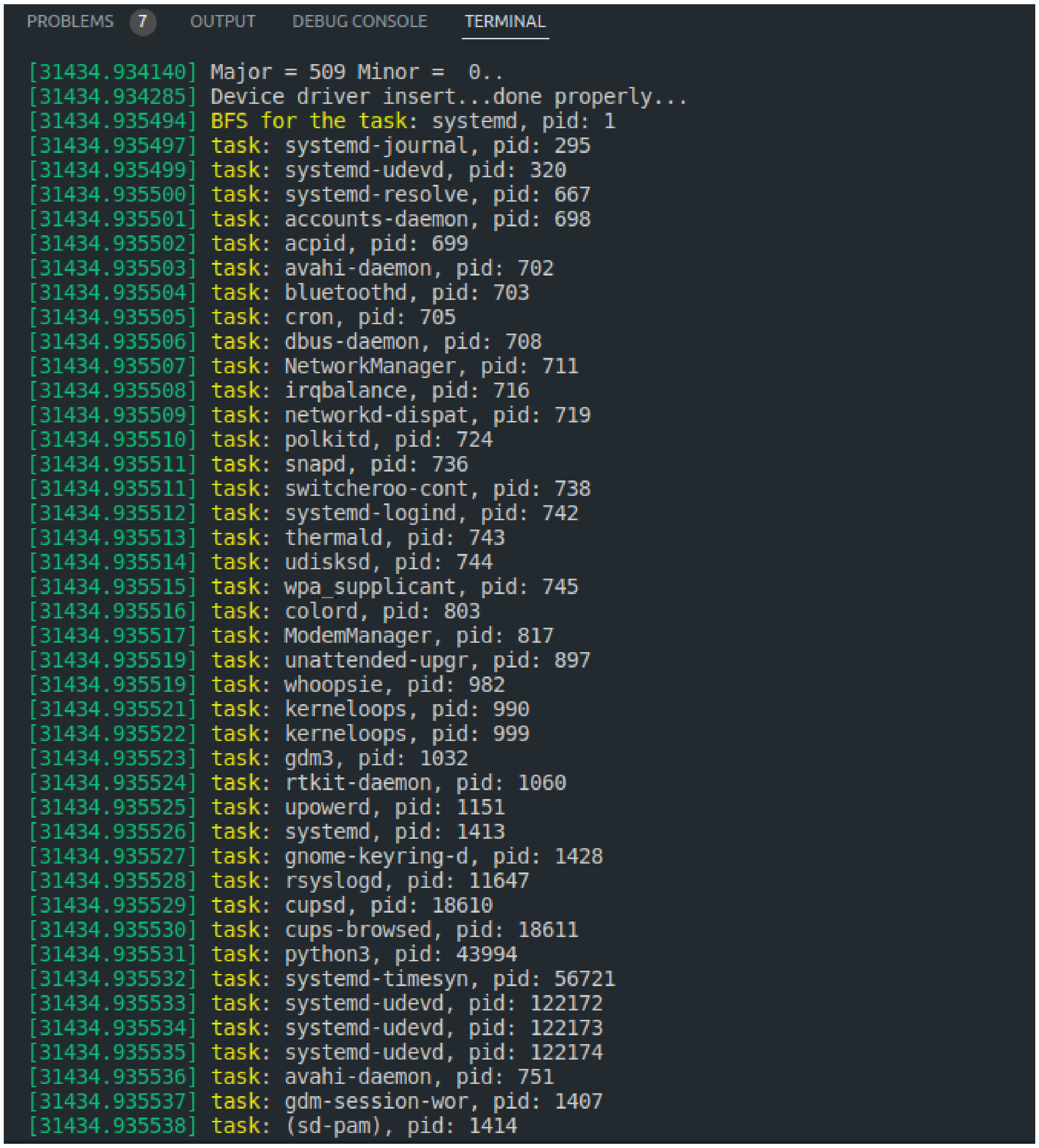
```
debian@debian:/home/debian/comp304_project1 shellfyre$ ls
crontab  dir1  dir2  foo1  foo2  foo3  food.pdf  foo.txt  Makefile  my_module.c  README.md  shellfyre  shellfyre.c
debian@debian:/home/debian/comp304_project1 shellfyre$ trash
Usage: trash [OPTION]... [FILE]...
Try 'trash --help' for more information.
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --help
Usage: trash [OPTION]... [FILE]...
Move files to the trash.

  --help     display this help and exit
  --restore  restore a file from trash
  --delete   delete a file from trash
  --move     move a file to trash

debian@debian:/home/debian/comp304_project1 shellfyre$ trash --move foo1
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --move foo2
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --move foo3
debian@debian:/home/debian/comp304_project1 shellfyre$ ls
crontab  dir1  dir2  food.pdf  foo.txt  Makefile  my_module.c  README.md  shellfyre  shellfyre.c
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --list
foo3
foo1
foo2
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --restore
1. foo3
2. foo1
3. foo2
Enter the index of the file you want to restore: 1
mv /home/debian/.trash/foo3 ./foo3
debian@debian:/home/debian/comp304_project1 shellfyre$
debian@debian:/home/debian/comp304_project1 shellfyre$ ls
crontab  dir1  dir2  foo3  food.pdf  foo.txt  Makefile  my_module.c  README.md  shellfyre  shellfyre.c
free(): invalid pointerian/comp304_project1 shellfyre$ ls
debian@debian:/home/debian/comp304_project1 shellfyre$ ls
crontab  dir1  dir2  foo3  food.pdf  foo.txt  Makefile  my_module.c  README.md  shellfyre  shellfyre.c
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --list
foo1
foo2
debian@debian:/home/debian/comp304_project1 shellfyre$ trash --delete
1. foo1
2. foo2
Enter the index of the file you want to delete: 1
debian@debian:/home/debian/comp304_project1 shellfyre$
debian@debian:/home/debian/comp304_project1 shellfyre$ 
```

Part 3, Kernel Modules:

In this part we started with the kernel module template code from the first assignment. We implemented a ioctl function to communicate with the user applications, the second argument indicates whether the traverse will be DFS or BFS. The third argument is the pid of the process that will be the root of the traversal. We then implemented the bfs function. For that we first needed to implement a FIFO queue for the struct task_struct. Then we implemented an iterative BFS algorithm. We then implemented the DFS recursively, for both searches we printed the process name and the pid to the kernel log. Finally we integrated this module with the shellfyre. (picture in the next page)



```
[31434.934140] Major = 509 Minor =  0..
[31434.934285] Device driver insert...done properly...
[31434.935494] BFS for the task: systemd, pid: 1
[31434.935497] task: systemd-journal, pid: 295
[31434.935499] task: systemd-udevd, pid: 320
[31434.935500] task: systemd-resolve, pid: 667
[31434.935501] task: accounts-daemon, pid: 698
[31434.935502] task: acpid, pid: 699
[31434.935503] task: avahi-daemon, pid: 702
[31434.935504] task: bluetoothd, pid: 703
[31434.935505] task: cron, pid: 705
[31434.935506] task: dbus-daemon, pid: 708
[31434.935507] task: NetworkManager, pid: 711
[31434.935508] task: irqbalance, pid: 716
[31434.935509] task: networkd-dispat, pid: 719
[31434.935510] task: polkitd, pid: 724
[31434.935511] task: snapd, pid: 736
[31434.935511] task: switcheroo-cont, pid: 738
[31434.935512] task: systemd-logind, pid: 742
[31434.935513] task: thermald, pid: 743
[31434.935514] task: udisksd, pid: 744
[31434.935515] task: wpa_supplicant, pid: 745
[31434.935516] task: colord, pid: 803
[31434.935517] task: ModemManager, pid: 817
[31434.935519] task: unattended-upgr, pid: 897
[31434.935519] task: whoopsie, pid: 982
[31434.935521] task: kerneloops, pid: 990
[31434.935522] task: kerneloops, pid: 999
[31434.935523] task: gdm3, pid: 1032
[31434.935524] task: rtkit-daemon, pid: 1060
[31434.935525] task: upowerd, pid: 1151
[31434.935526] task: systemd, pid: 1413
[31434.935527] task: gnome-keyring-d, pid: 1428
[31434.935528] task: rsyslogd, pid: 11647
[31434.935529] task: cupsd, pid: 18610
[31434.935530] task: cups-browsed, pid: 18611
[31434.935531] task: python3, pid: 43994
[31434.935532] task: systemd-timesyn, pid: 56721
[31434.935533] task: systemd-udevd, pid: 122172
[31434.935534] task: systemd-udevd, pid: 122173
[31434.935535] task: systemd-udevd, pid: 122174
[31434.935536] task: avahi-daemon, pid: 751
[31434.935537] task: gdm-session-wor, pid: 1407
[31434.935538] task: (sd-pam), pid: 1414
```