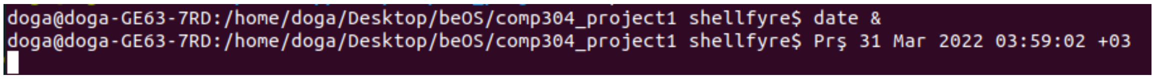
COMP304: Operating Systems
Project 1: Shellfyre

Betül DEMİRTAŞ - 68858
Doğa Ege İNHANLI - 69033

P.s: All the details of our project can be found in the repository at: https://github.com/bdemirtas18/beOS

1. Basic Commands

- For the UNIX command executions, since we know that UNIX commands are located in the "usr/bin" directory, as a first parameter, we give this path by adding the command name at the end of it. As a second parameter, we give the arguments of the command directly. By this way, "execv" function handles these basic commands in the child process.
- For the background execution of the programs, we realized that the existence of the ampersand (&) is already stored in the "command_struct" as "background" boolean. Thus, we check this boolean and wait for the child process accordingly.



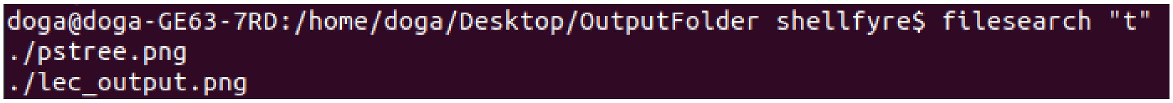
```
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ date &
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ Prş 31 Mar 2022 03:59:02 +03
```

Figure 1: Sample execution of a command with ampersand (&)

2. Custom Commands

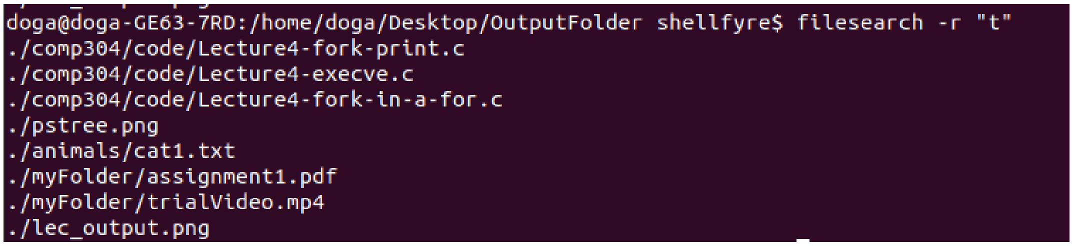
a. filesearch

When the "filesearch" command is typed on the shellfyre, we created a text file named "files.txt". Then, our "my_ls" function writes the names of the files and directories in the current directory to "files.txt". If the flag "-r" is also given, this function goes over the subdirectories recursively. After that, in our "print_files" function, we extract the extensions and paths from the file/directory names and search for the given string in the lines of the "files.txt". If the flag "-o" is also given, this function executes the "xdg-open" command in a child process for each file/directory.



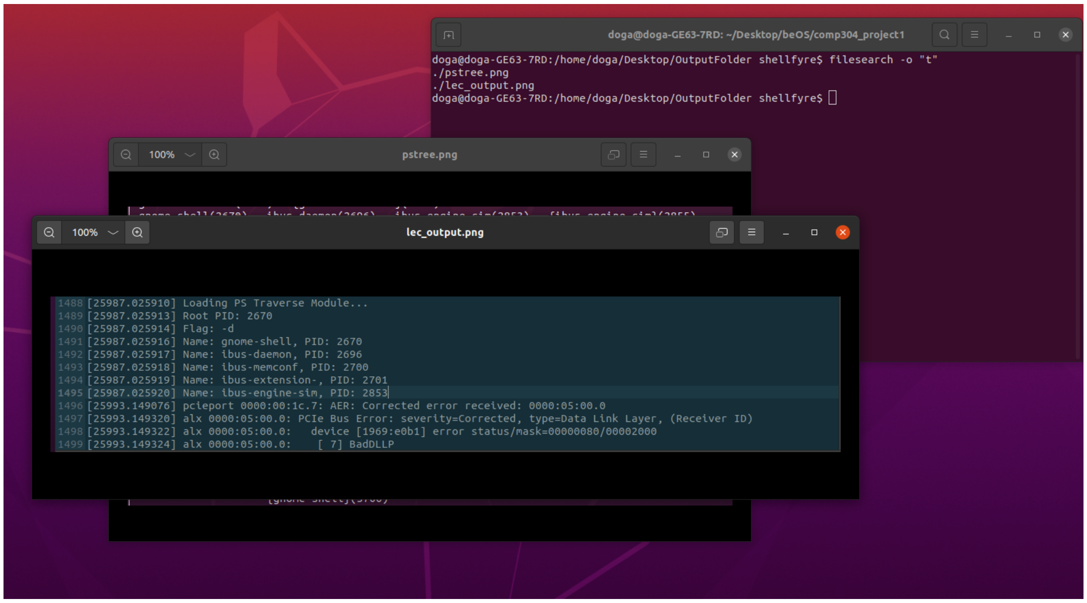
```
doga@doga-GE63-7RD:/home/doga/Desktop/OutputFolder shellfyre$ filesearch "t"
./pstree.png
./lec_output.png
```

Figure 2: Sample execution of "filesearch" command without any flag.



```
doga@doga-GE63-7RD:/home/doga/Desktop/OutputFolder shellfyre$ filesearch -r "t"
./comp304/code/Lecture4-fork-print.c
./comp304/code/Lecture4-execve.c
./comp304/code/Lecture4-fork-in-a-for.c
./pstree.png
./animals/cat1.txt
./myFolder/assignment1.pdf
./myFolder/trialVideo.mp4
./lec_output.png
```

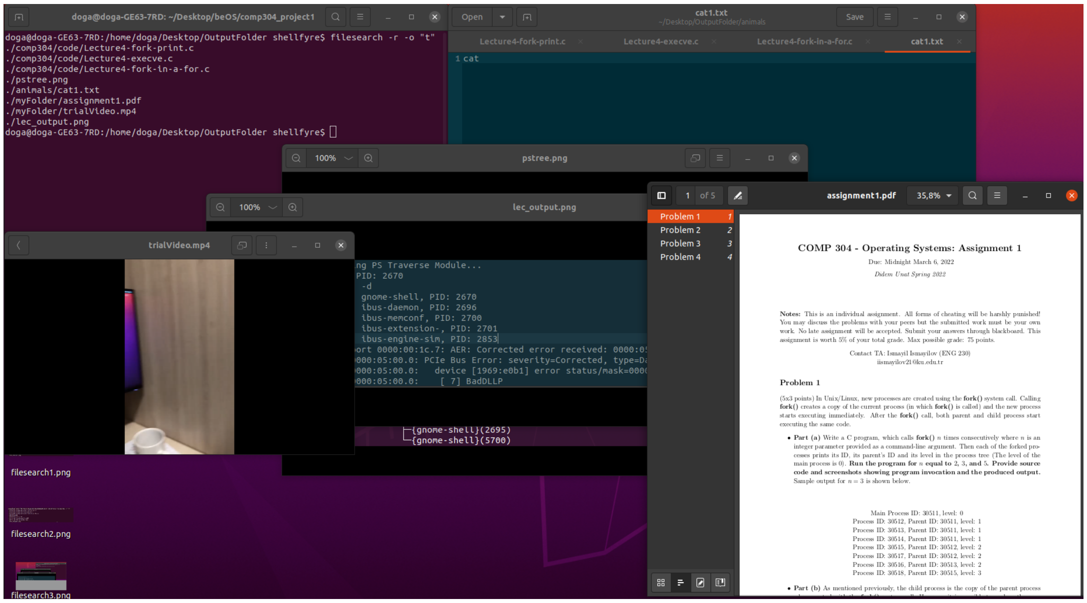
Figure 3: Sample execution of “filesearch” command with the flag “-r”.



```
doga@doga-GE63-7RD:/home/doga/Desktop/OutputFolder shellfyre$ filesearch -o "t"
./pstree.png
./lec_output.png
doga@doga-GE63-7RD:/home/doga/Desktop/OutputFolder shellfyre$
```

```
[25987.025910] Loading PS Traverse Module...
[25987.025913] Root PID: 2670
[25987.025914] Flag: -d
[25987.025916] Name: gnome-shell, PID: 2670
[25987.025917] Name: ibus-daemon, PID: 2696
[25987.025918] Name: ibus-memconf, PID: 2700
[25987.025919] Name: ibus-extension-, PID: 2701
[25987.025920] Name: ibus-engine-sim, PID: 2853
[25993.149076] pcieport 0000:00:1c.7: AER: Corrected error received: 0000:05:00.0
[25993.149320] alx 0000:05:00.0: PCIe Bus Error: severity=Corrected, type=Data Link Layer, (Receiver ID)
[25993.149322] alx 0000:05:00.0:   device [1969:e0b1] error status/mask=00000080/00002000
[25993.149324] alx 0000:05:00.0:    [ 7] BadDLLP
```

Figure 4: Sample execution of “filesearch” command with the flag “-o”.



```
doga@doga-GE63-7RD:/home/doga/Desktop/OutputFolder shellfyre$ filesearch -r -o "t"
./comp304/code/Lecture4-fork-print.c
./comp304/code/Lecture4-execve.c
./comp304/code/Lecture4-fork-in-a-for.c
./pstree.png
./animals/cat1.txt
./myFolder/assignment1.pdf
./myFolder/trialVideo.mp4
./lec_output.png
doga@doga-GE63-7RD:/home/doga/Desktop/OutputFolder shellfyre$
```

```
cat
```

```
ng PS Traverse Module...
PID: 2670
 -d
 gnome-shell, PID: 2670
 ibus-daemon, PID: 2696
 ibus-memconf, PID: 2700
 ibus-extension-, PID: 2701
 ibus-engine-sim, PID: 2853
ort 0000:00:1c.7: AER: Corrected error received: 0000:05
000:05:00.0: PCIe Bus Error: severity=Corrected, type=Da
000:05:00.0:   device [1969:e0b1] error status/mask=0000
000:05:00.0:    [ 7] BadDLLP
```

```
      ├{gnome-shell}(2695)
      └{gnome-shell}(5700)
```

COMP 304 - Operating Systems: Assignment 1

Due: Midnight March 6, 2022

Didem Unat Spring 2022

Notes: This is an individual assignment. All forms of cheating will be harshly punished! You may discuss the problems with your peers but the submitted work must be your own work. No late assignment will be accepted. Submit your answers through blackboard. This assignment is worth 5% of your total grade. Max possible grade: 75 points.

Contact TA: Ismayil Ismayilov (ENG 230)
iismayilov21@ku.edu.tr

Problem 1

(5x3 points) In Unix/Linux, new processes are created using the **fork()** system call. Calling **fork()** creates a copy of the current process (in which **fork()** is called) and the new process starts executing immediately. After the **fork()** call, both parent and child process start executing the same code.

- **Part (a)** Write a C program, which calls **fork()** n times consecutively where n is an integer parameter provided as a command-line argument. Then each of the forked processes prints its ID, its parent's ID and its level in the process tree (The level of the main process is 0). **Run the program for n equal to 2, 3, and 5. Provide source code and screenshots showing program invocation and the produced output.** Sample output for $n = 3$ is shown below.

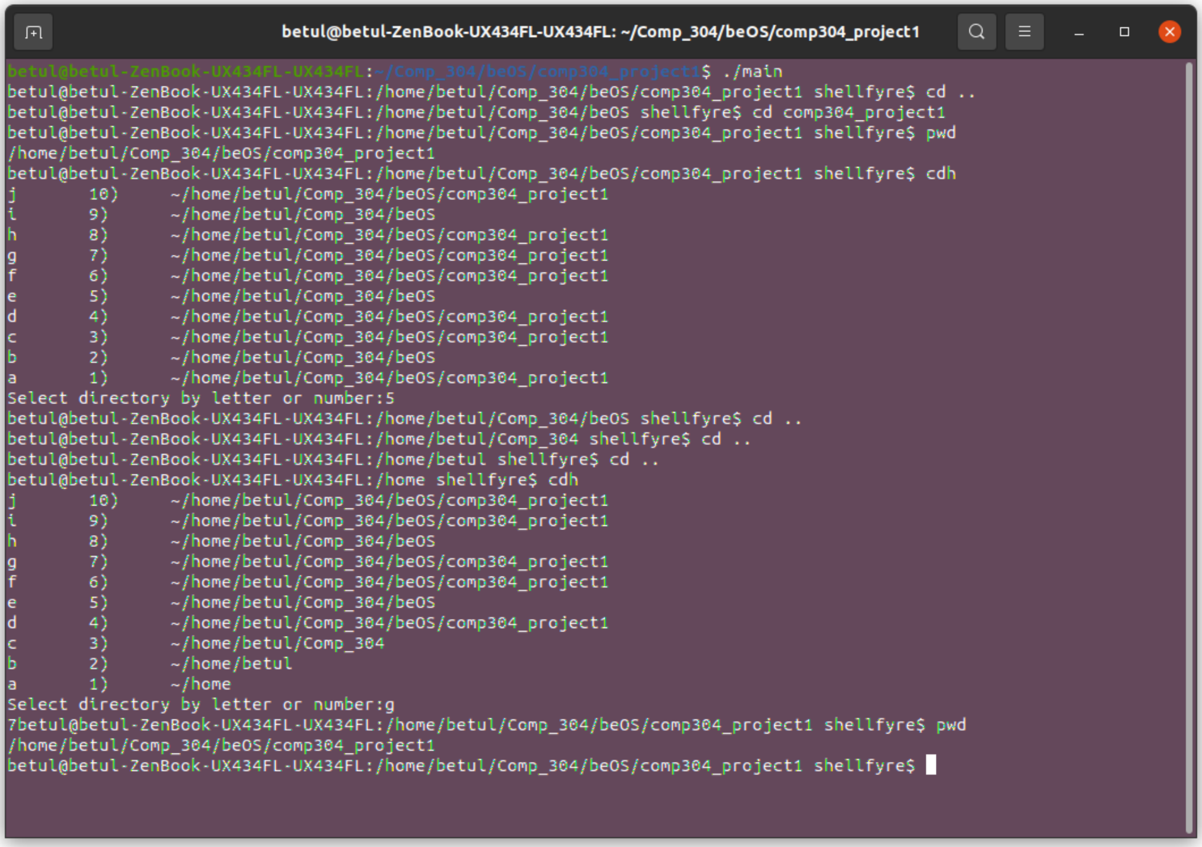
```
Main Process ID: 30511, level: 0
Process ID: 30512, Parent ID: 30511, level: 1
Process ID: 30513, Parent ID: 30511, level: 1
Process ID: 30514, Parent ID: 30511, level: 1
Process ID: 30515, Parent ID: 30512, level: 2
Process ID: 30517, Parent ID: 30512, level: 2
Process ID: 30516, Parent ID: 30513, level: 2
Process ID: 30518, Parent ID: 30515, level: 3
```

- **Part (b)** As mentioned previously, the child process is the copy of the parent process

Figure 5: Sample execution of “filesearch” command with the flag “-o” and “-r”.

b. cdh

In order to be able to keep a directory history we wrote the function “save_directory_history”. This function basically records the current directory path, then goes in to the main directory where we keep the history file of our shell and appends the recorded path to the history file called “beOS_shellfyre_directory_history.txt”, and comes back to the original directory. Our history file has been manually created by us outside of the shell program to ensure that it lives beyond sessions. Finally, we made sure that the every command with the possibility of changing into a new directory (such as cd and take) calls this function so that we can keep a record of the visited directories. When the “cdh” command is typed on the shellfyre, we call our “print_directory_history” function. This function reads the history file line by line, and takes the last 10 (or less if there are not enough entries in the file) entry then prints this information on the terminal with their corresponding numbers and letters. Later, it takes an input from the user and finds the corresponding path of the letter or the number that the user has provided. Finally, by using “chdir” function we go to the directory that the user wanted to go.



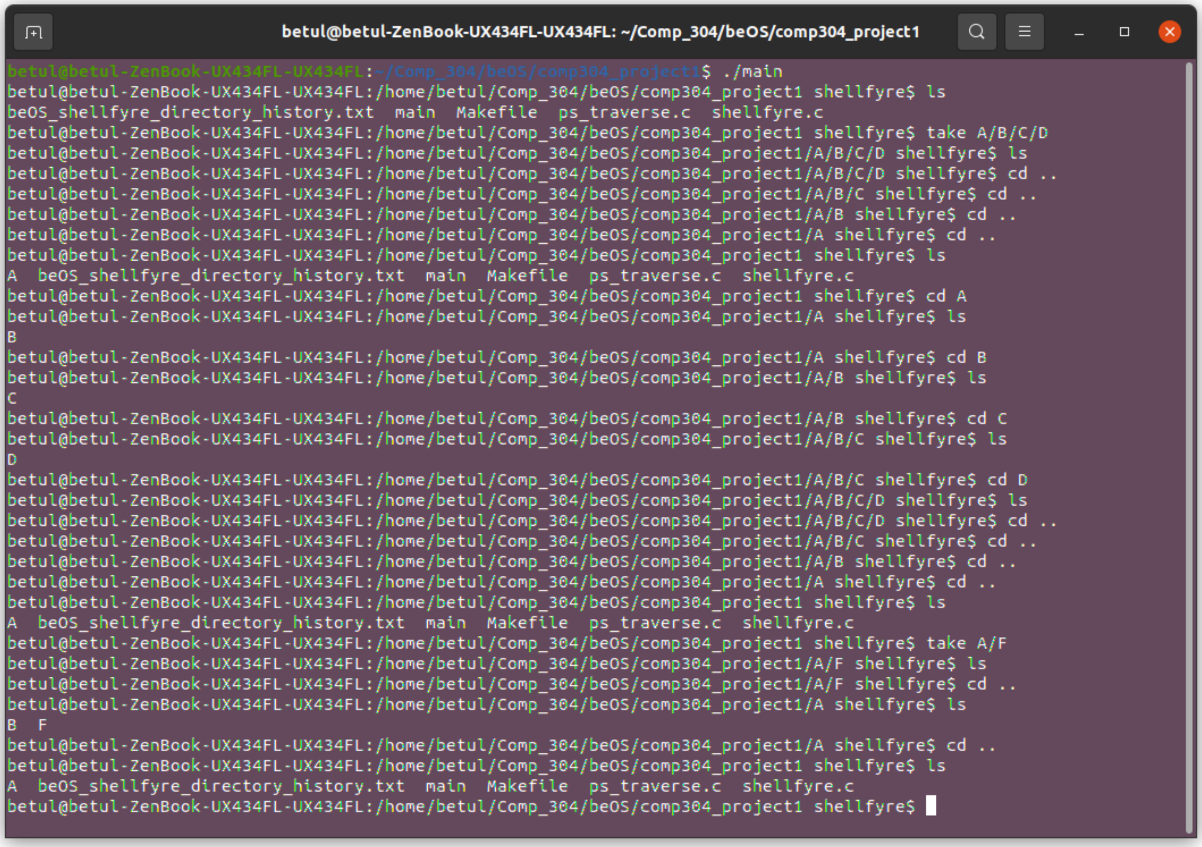
betul@betul-ZenBook-UX434FL-UX434FL: ~/Comp_304/beOS/comp304_project1

```
betul@betul-ZenBook-UX434FL-UX434FL:~/Comp_304/beOS/comp304_project1$ ./main
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS shellfyre$ cd comp304_project1
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ pwd
/home/betul/Comp_304/beOS/comp304_project1
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ cdh
j       10)     ~/home/betul/Comp_304/beOS/comp304_project1
i        9)     ~/home/betul/Comp_304/beOS
h        8)     ~/home/betul/Comp_304/beOS/comp304_project1
g        7)     ~/home/betul/Comp_304/beOS/comp304_project1
f        6)     ~/home/betul/Comp_304/beOS/comp304_project1
e        5)     ~/home/betul/Comp_304/beOS
d        4)     ~/home/betul/Comp_304/beOS/comp304_project1
c        3)     ~/home/betul/Comp_304/beOS/comp304_project1
b        2)     ~/home/betul/Comp_304/beOS
a        1)     ~/home/betul/Comp_304/beOS/comp304_project1
Select directory by letter or number:5
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304 shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home shellfyre$ cdh
j       10)     ~/home/betul/Comp_304/beOS/comp304_project1
i        9)     ~/home/betul/Comp_304/beOS/comp304_project1
h        8)     ~/home/betul/Comp_304/beOS
g        7)     ~/home/betul/Comp_304/beOS/comp304_project1
f        6)     ~/home/betul/Comp_304/beOS/comp304_project1
e        5)     ~/home/betul/Comp_304/beOS
d        4)     ~/home/betul/Comp_304/beOS/comp304_project1
c        3)     ~/home/betul/Comp_304
b        2)     ~/home/betul
a        1)     ~/home
Select directory by letter or number:g
7betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ pwd
/home/betul/Comp_304/beOS/comp304_project1
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ 
```

Figure 6: The working example of cdh command both with a letter and a number entry

c. take

When the “take” command is typed on the shellfyre, we call our “take” function. This function first reads the argument provided by the user and divides it into tokens which are splitted with “/” symbol. It basically writes all those tokens into a file and loops through each one by one. If the current directory name exist, it simply goes into it and checks the next one. Otherwise, it first creates a new directory with the current directory name, goes into it, then checks the next one. Also, as mentioned above, to be able to record this action in our directory history, if the function returns a successful event, we also call our “save_directory_history” function at the end.



betul@betul-ZenBook-UX434FL-UX434FL: ~/Comp_304/beOS/comp304_project1

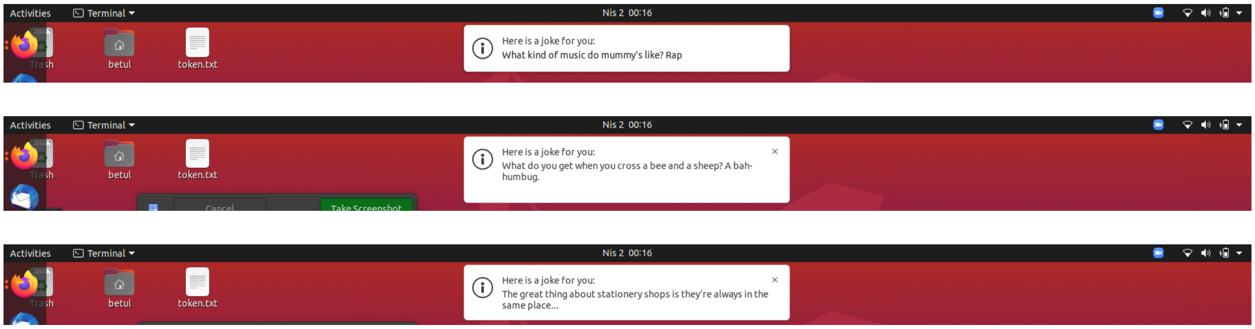
```
betul@betul-ZenBook-UX434FL-UX434FL:~/Comp_304/beOS/comp304_project1$ ./main
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ ls
beOS_shellfyre_directory_history.txt  main  Makefile  ps_traverse.c  shellfyre.c
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ take A/B/C/D
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C/D shellfyre$ ls
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C/D shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ ls
A  beOS_shellfyre_directory_history.txt  main  Makefile  ps_traverse.c  shellfyre.c
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ cd A
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A shellfyre$ ls
B
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A shellfyre$ cd B
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B shellfyre$ ls
C
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B shellfyre$ cd C
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C shellfyre$ ls
D
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C shellfyre$ cd D
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C/D shellfyre$ ls
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C/D shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B/C shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/B shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ ls
A  beOS_shellfyre_directory_history.txt  main  Makefile  ps_traverse.c  shellfyre.c
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ take A/F
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/F shellfyre$ ls
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A/F shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A shellfyre$ ls
B  F
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1/A shellfyre$ cd ..
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ ls
A  beOS_shellfyre_directory_history.txt  main  Makefile  ps_traverse.c  shellfyre.c
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ 
```

Figure 7: The working example of take command both with existing and non-existing directories

d. joker

When the “joker” command is typed on the shellfyre, we call our “schedule_joke" function. This function first opens a file and writes the command that we want to put into the crontab file. We designed the crontab input like “15 * * * *

XDG_RUNTIME_DIR=/run/user/$(id -u) notify-send \"Here is a joke for you:\" \"$(curl https://icanhazdadjoke.com)\"". This argument schedules a notify-send program (sends messages as desktop notifications) with a message that is gotten by using curl command to import a joke message from the site https://icanhazdadjoke.com. This argument is scheduled for every 15 minutes for any hour of any day of any month. Then we give this text file with the command to the crontab by making a system call (crontab joker.txt).



Here is a joke for you:
What kind of music do mummy's like? Rap

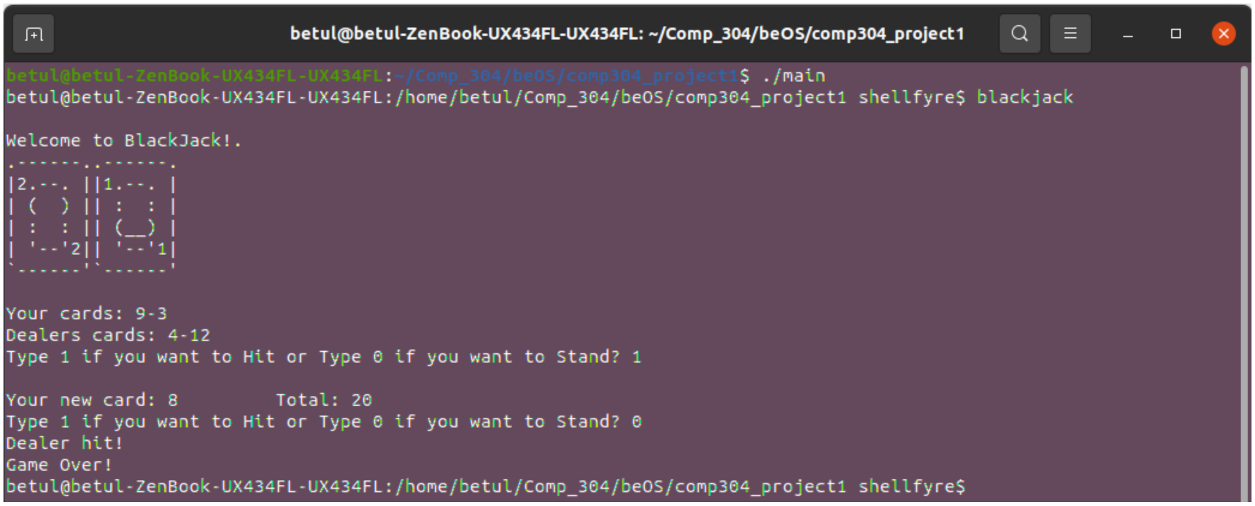
Here is a joke for you:
What do you get when you cross a bee and a sheep? A bah-humbug.

Here is a joke for you:
The great thing about stationery shops is they're always in the same place...

Figure 8: Example joke outputs from when the command is given

e. blackjack

When the “blackjack” command is typed on the shellfyre, the user can play a simple blackjack game on the terminal against a randomized player. The program simply draws cards randomly and takes command line arguments from the user to play the game such as “hit and stand”.


betul@betul-ZenBook-UX434FL-UX434FL: ~/Comp_304/beOS/comp304_project1

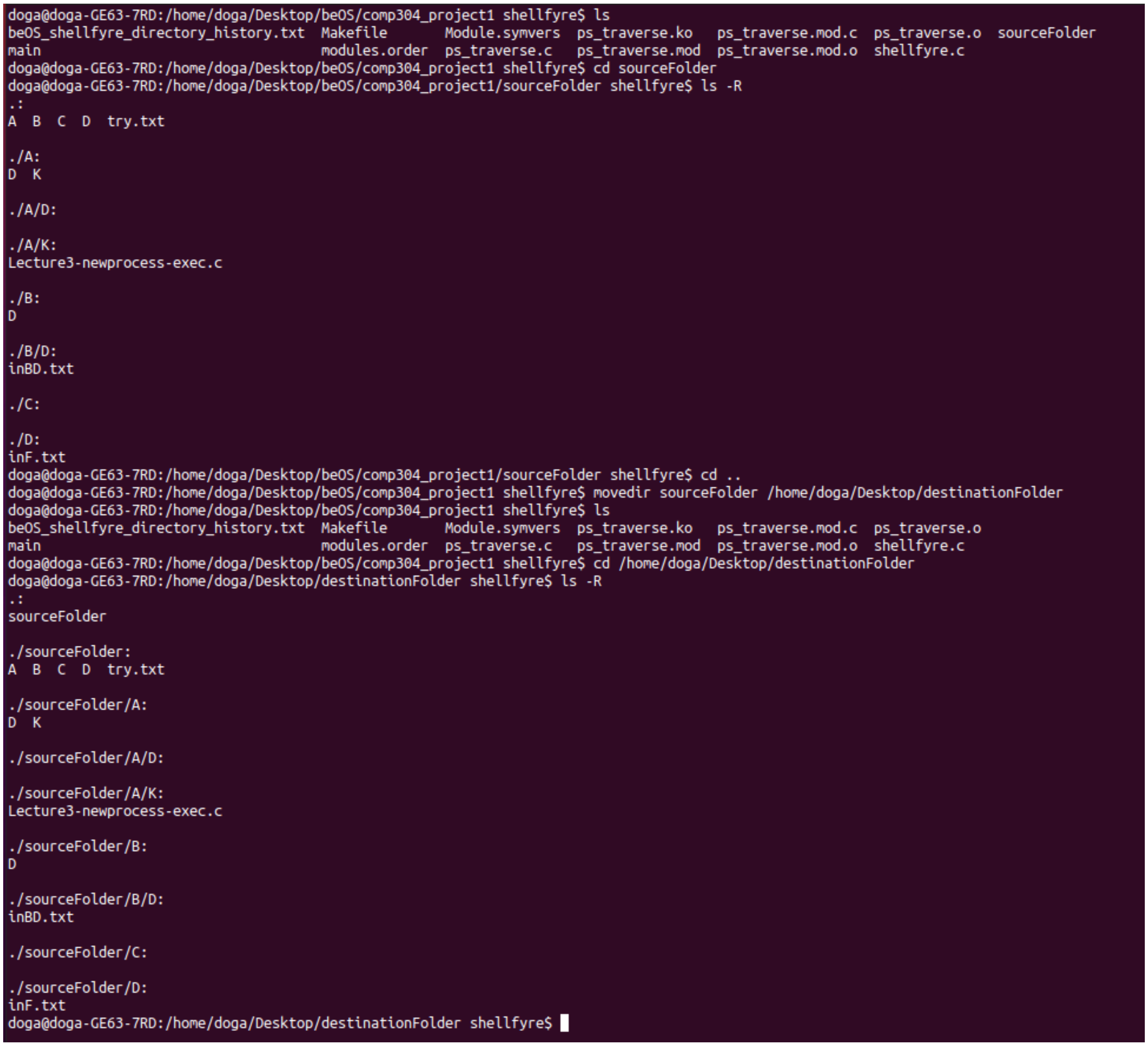
```
betul@betul-ZenBook-UX434FL-UX434FL:~/Comp_304/beOS/comp304_project1$ ./main
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$ blackjack

Welcome to BlackJack!.
.------..------.
|2.--. ||1.--. |
| (  ) || :  : |
| :  : || (__) |
| '--'2|| '--'1|
`------'`------'

Your cards: 9-3
Dealers cards: 4-12
Type 1 if you want to Hit or Type 0 if you want to Stand? 1

Your new card: 8          Total: 20
Type 1 if you want to Hit or Type 0 if you want to Stand? 0
Dealer hit!
Game Over!
betul@betul-ZenBook-UX434FL-UX434FL:/home/betul/Comp_304/beOS/comp304_project1 shellfyre$
```

Figure 9: An example blackjack game play

f. movedir

This command must be called with two arguments, first one of which is the source directory and second one of which is the destination directory. It basically moves the source directory inside the destination directory with all the content inside it. Only thing which must be considered is the file type inside the source directory. The source directory can contain only regular files (.c, .txt, .py etc.) or subdirectories. In other words, there must not be any files with extensions like .pdf, .mp4, .png etc.

Our implementation reads all the contents inside the source directory. If it is a regular file, the file is copied to the destination directory. If it is a directory, the movedir continues copying recursively. After the copying is completed, the original version of the source directory is deleted.

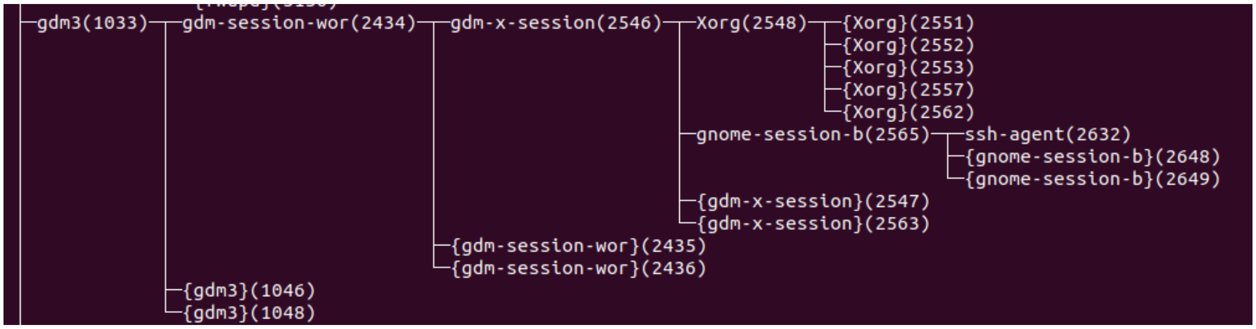


```
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ ls
beOS_shellfyre_directory_history.txt  Makefile       Module.symvers  ps_traverse.ko   ps_traverse.mod.c  ps_traverse.o  sourceFolder
main                                  modules.order  ps_traverse.c   ps_traverse.mod  ps_traverse.mod.o  shellfyre.c
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ cd sourceFolder
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1/sourceFolder shellfyre$ ls -R
.:
A  B  C  D  try.txt

./A:
D  K

./A/D:

./A/K:
Lecture3-newprocess-exec.c

./B:
D

./B/D:
inBD.txt

./C:

./D:
inF.txt
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1/sourceFolder shellfyre$ cd ..
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ movedir sourceFolder /home/doga/Desktop/destinationFolder
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ ls
beOS_shellfyre_directory_history.txt  Makefile       Module.symvers  ps_traverse.ko   ps_traverse.mod.c  ps_traverse.o
main                                  modules.order  ps_traverse.c   ps_traverse.mod  ps_traverse.mod.o  shellfyre.c
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ cd /home/doga/Desktop/destinationFolder
doga@doga-GE63-7RD:/home/doga/Desktop/destinationFolder shellfyre$ ls -R
.:
sourceFolder

./sourceFolder:
A  B  C  D  try.txt

./sourceFolder/A:
D  K

./sourceFolder/A/D:

./sourceFolder/A/K:
Lecture3-newprocess-exec.c

./sourceFolder/B:
D

./sourceFolder/B/D:
inBD.txt

./sourceFolder/C:

./sourceFolder/D:
inF.txt
doga@doga-GE63-7RD:/home/doga/Desktop/destinationFolder shellfyre$ █
```

Figure 10: Sample execution of “movedir” command

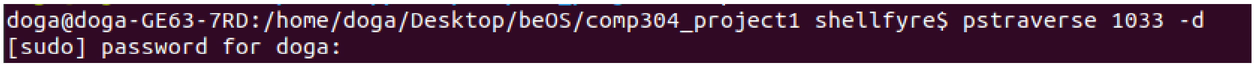
3. Kernel Modules

We created a Kernel Module named ps_traverse.c which contains depth first search and breadth first search algorithms. When the user enters “pstraverse” command with a PID and a flag representing the searching type in the tree, the kernel module is directly installed. Then, our kernel module checks the validity of the given PID and operates the corresponding searching type which is given as a flag. For the implementation of the depth first search and breadth first search, we used the Project 2 - Linux Kernel Module for Listing Tasks from the book [1]. After that, the kernel module is removed from the kernel



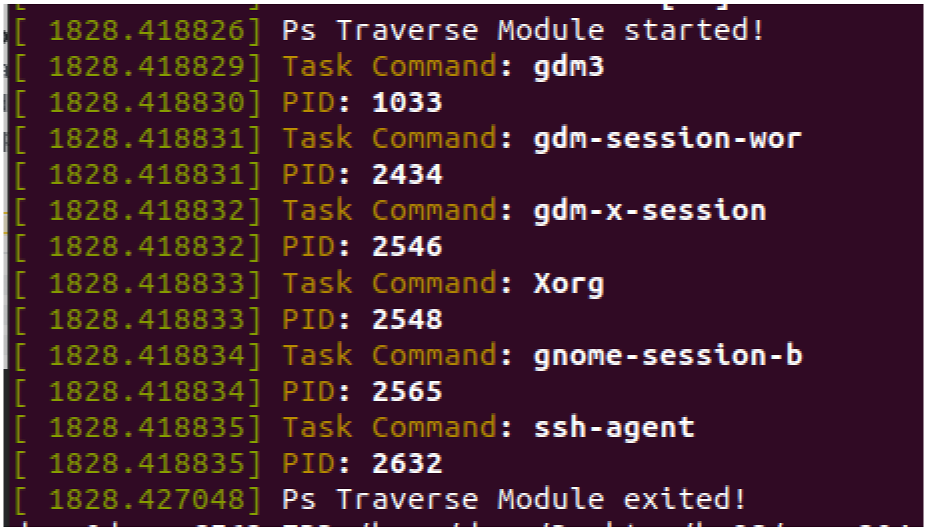
```
├─gdm3(1033)─┬─gdm-session-wor(2434)─┬─gdm-x-session(2546)─┬─Xorg(2548)─┬─{Xorg}(2551)
│            │                       │                     │            ├─{Xorg}(2552)
│            │                       │                     │            ├─{Xorg}(2553)
│            │                       │                     │            ├─{Xorg}(2557)
│            │                       │                     │            └─{Xorg}(2562)
│            │                       │                     ├─gnome-session-b(2565)─┬─ssh-agent(2632)
│            │                       │                     │                       ├─{gnome-session-b}(2648)
│            │                       │                     │                       └─{gnome-session-b}(2649)
│            │                       │                     ├─{gdm-x-session}(2547)
│            │                       │                     └─{gdm-x-session}(2563)
│            │                       ├─{gdm-session-wor}(2435)
│            │                       └─{gdm-session-wor}(2436)
│            ├─{gdm3}(1046)
│            └─{gdm3}(1048)
```

Fİgure 11: pstree -p output of the process with PID = 1033



```
doga@doga-GE63-7RD:/home/doga/Desktop/beOS/comp304_project1 shellfyre$ pstraverse 1033 -d
[sudo] password for doga:
```

Figure 12: Sample execution of “pstraverse” command



```
[ 1828.418826] Ps Traverse Module started!
[ 1828.418829] Task Command: gdm3
[ 1828.418830] PID: 1033
[ 1828.418831] Task Command: gdm-session-wor
[ 1828.418831] PID: 2434
[ 1828.418832] Task Command: gdm-x-session
[ 1828.418832] PID: 2546
[ 1828.418833] Task Command: Xorg
[ 1828.418833] PID: 2548
[ 1828.418834] Task Command: gnome-session-b
[ 1828.418834] PID: 2565
[ 1828.418835] Task Command: ssh-agent
[ 1828.418835] PID: 2632
[ 1828.427048] Ps Traverse Module exited!
```

Fİgure 13: Sample output of the execution

References:

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts*. Hoboken, N.J: Wiley, 2019.

[2] “Crontab and C program that should be executed into a terminal window,” *Ask Ubuntu*, 01-Jul-1965. [Online]. Available: https://askubuntu.com/questions/955582/crontab-and-c-program-that-should-be-executed-into-a-terminal-window. [Accessed: 03-Apr-2022].

[3] “Curl,” *curl*. [Online]. Available: https://curl.se/. [Accessed: 03-Apr-2022].

[4] *Executing a file (the GNU C library)*. [Online]. Available: https://www.gnu.org/software/libc/manual/html_node/Executing-a-File.html. [Accessed: 03-Apr-2022].

[5] “File I/O,” *C*. [Online]. Available: https://www.tutorialspoint.com/cprogramming/c_file_io.htm. [Accessed: 03-Apr-2022].

[6] “How to open a folder (with GUI) from terminal?,” *Ask Ubuntu*, 01-Jan-1961. [Online]. Available: https://askubuntu.com/questions/265641/how-to-open-a-folder-with-gui-from-terminal. [Accessed: 03-Apr-2022].

[7] “Kernel module that iterates over the Tree of children of a task using depth first search,” *Stack Overflow*, 01-Jul-1961. [Online]. Available: https://stackoverflow.com/questions/19208487/kernel-module-that-iterates-over-the-tree-of-children-of-a-task-using-depth-firs. [Accessed: 03-Apr-2022].

[8] “Kernel modules¶,” *Kernel modules - The Linux Kernel documentation*. [Online]. Available: https://linux-kernel-labs.github.io/refs/heads/master/labs/kernel_modules.html. [Accessed: 03-Apr-2022].

[9] R. Love, *Linux kernel development: A practical guide to the design and implementation of the linux kernel*. .

[10] “Strtok() and strtok_r() functions in C with examples,” *GeeksforGeeks*, 13-Dec-2021. [Online]. Available: https://www.geeksforgeeks.org/strtok-strtok_r-functions-c-examples/. [Accessed: 03-Apr-2022].

[11] https://askubuntu.com/questions/265641/how-to-open-a-folder-with-gui-from-terminal. [Accessed: 03-Apr-2022].

[12] *Whereis Command*. [Online]. Available: https://www.ibm.com/docs/en/aix/7.1?topic=w-whereis-command. [Accessed: 03-Apr-2022].