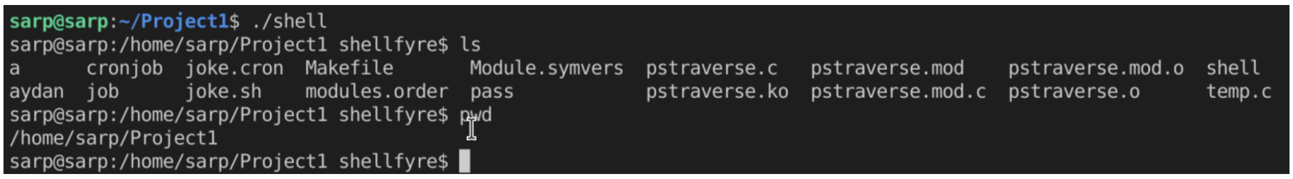
Sarp Göl
ID: 72368

Asu Tutku Gökçek
ID: 71766

Part 1 Basic Commands :

For this part I implemented my own exec function. The starter code was already parsed in a struct. I used the command struct to get the data. Command->args contained args and command->name contains the main command which will be executed. I gave the arguments and command name to a function and used strcat function with "/bin/". Then I executed the command with its arguments using execv(). For the background part I checked whether the command is in background or not using the command structure. If it



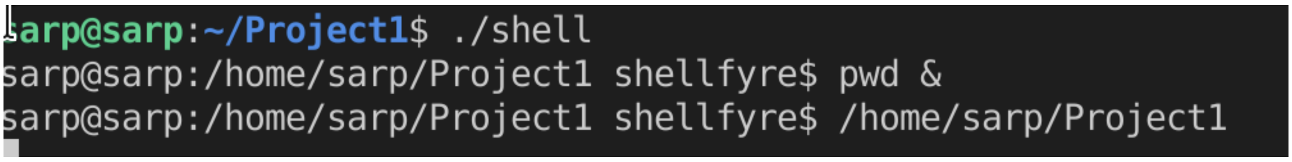
```
sarp@sarp:~/Project1$ ./shell
sarp@sarp:/home/sarp/Project1 shellfyre$ ls
a      cronjob  joke.cron  Makefile       Module.symvers  pstraverse.c   pstraverse.mod    pstraverse.mod.o  shell
aydan  job      joke.sh    modules.order  pass            pstraverse.ko  pstraverse.mod.c  pstraverse.o      temp.c
sarp@sarp:/home/sarp/Project1 shellfyre$ pwd
/home/sarp/Project1
sarp@sarp:/home/sarp/Project1 shellfyre$ 
```

is in the background I made the parent process wait.

Part 2 Custom Commands :

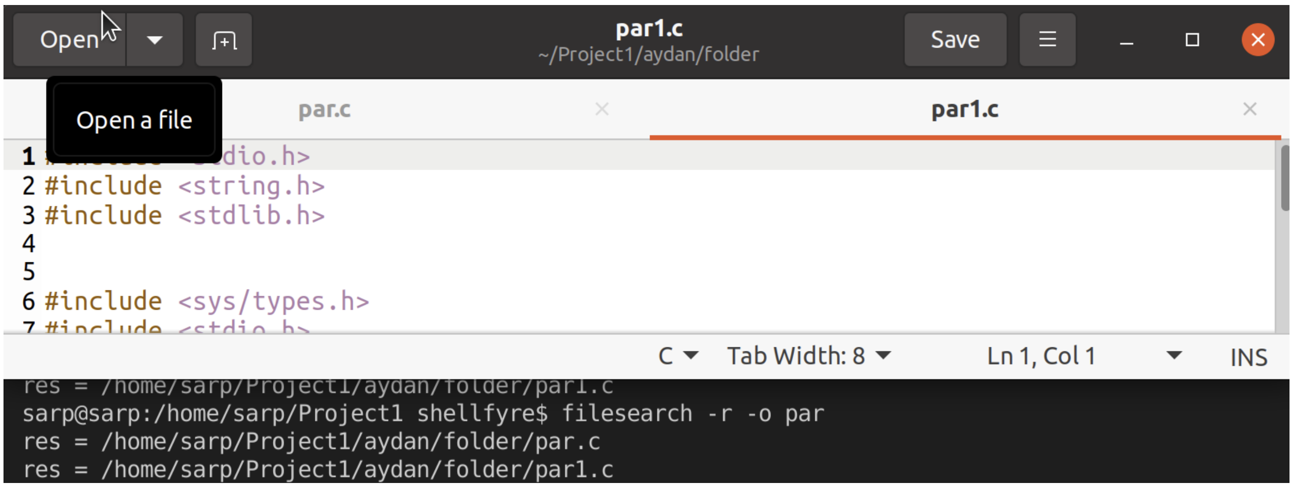
Filesearch:

For the filesearch command I checked the arguments. If there is "-r" option ı called my recursive



```
sarp@sarp:~/Project1$ ./shell
sarp@sarp:/home/sarp/Project1 shellfyre$ pwd &
sarp@sarp:/home/sarp/Project1 shellfyre$ /home/sarp/Project1
```

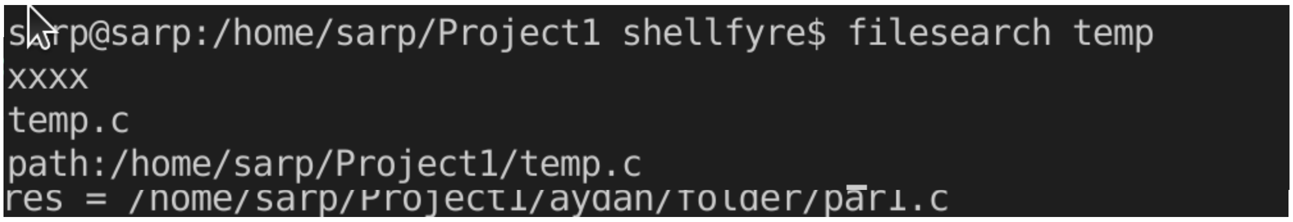
function. If there is "-o" option I opened the file using xdg-open. If there are no options then it checks the current directory. To find all matching file names I used regex expressions



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


#include <sys/types.h>
#include <stdio.h>
```

```
res = /home/sarp/Project1/aydan/folder/par1.c
sarp@sarp:/home/sarp/Project1 shellfyre$ filesearch -r -o par
res = /home/sarp/Project1/aydan/folder/par.c
res = /home/sarp/Project1/aydan/folder/par1.c
```



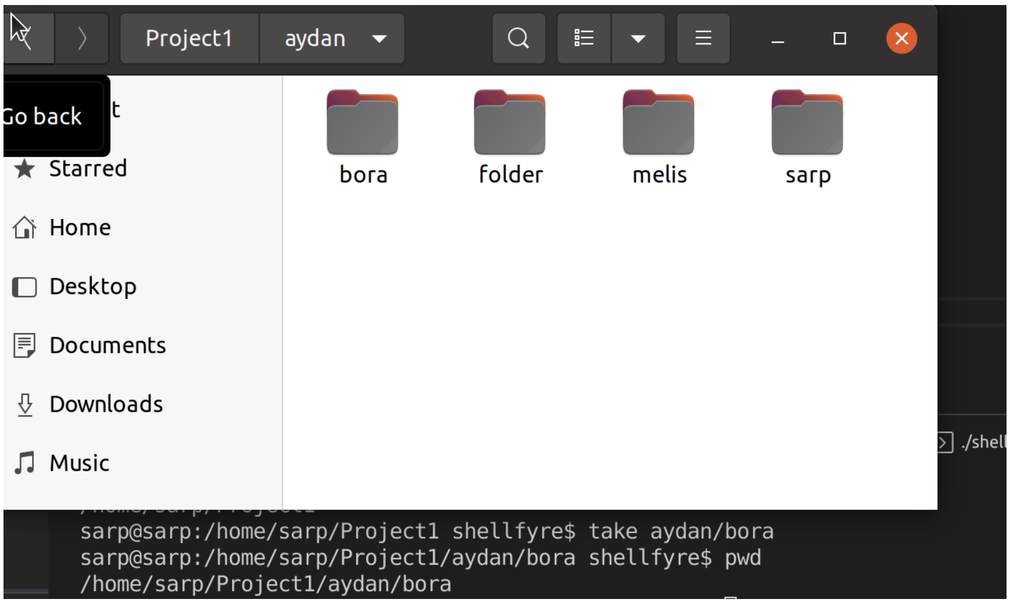
```
sarp@sarp:/home/sarp/Project1 shellfyre$ filesearch temp
xxxx
temp.c
path:/home/sarp/Project1/temp.c
res = /home/sarp/Project1/aydan/folder/par1.c
```

Cdh command:

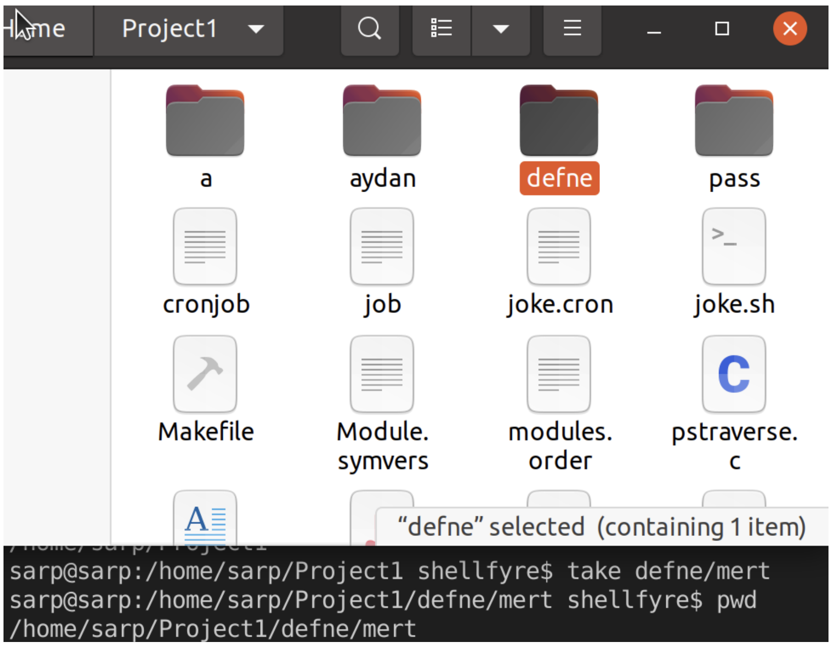
For this command we ran out of time. However, we worked on saving the visited directories on a queue and listing them from there. The implementation of queue can be seen in the code.

Take command:

To implement take I command first I took the current directory using getcwd function. I tokenized the user input with "/" delimiter. If the current directory combined with first token does not exists I create a new directory. After last token I switch to the newly created directory.



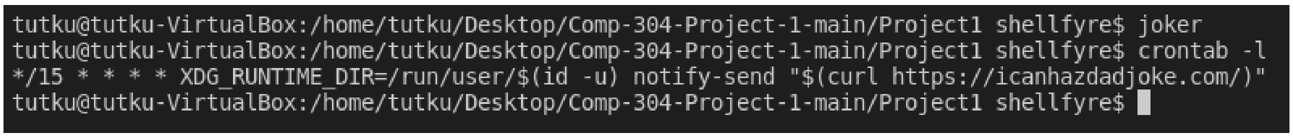
```
sarp@sarp:/home/sarp/Project1 shellfyre$ take aydan/bora
sarp@sarp:/home/sarp/Project1/aydan/bora shellfyre$ pwd
/home/sarp/Project1/aydan/bora
```



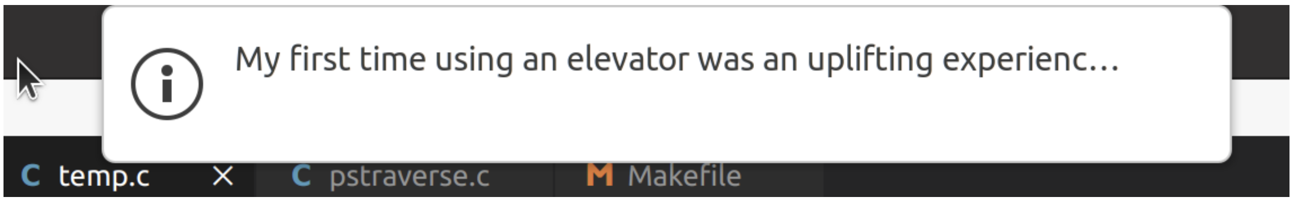
```
sarp@sarp:/home/sarp/Project1 shellfyre$ take defne/mert
sarp@sarp:/home/sarp/Project1/defne/mert shellfyre$ pwd
/home/sarp/Project1/defne/mert
```

Joker command :

For this command I first tried different ways of getting the crontab line from shell directly but I couldn't manage to get it to work. The system call was not doing the job either. Therefore, I changed the idea to creating a file and using that to make crontab work. In this version of the command, the crontab line is getting written to a file. I also had to enclose the curl command to get the correct input for the crontab line.



```
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$ joker
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$ crontab -l
*/15 * * * * XDG_RUNTIME_DIR=/run/user/$(id -u) notify-send "$(curl https://icanhazdadjoke.com/)"
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$
```

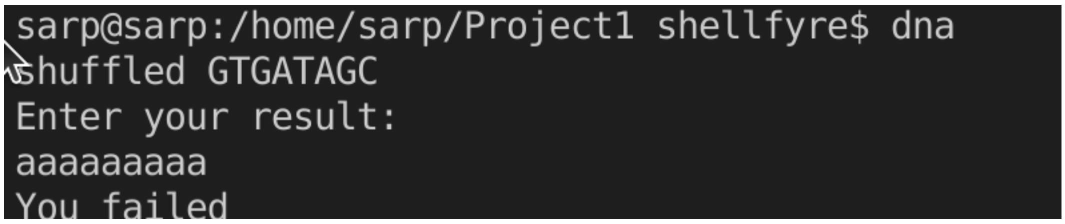


My first time using an elevator was an uplifting experienc…

Your awesome command:

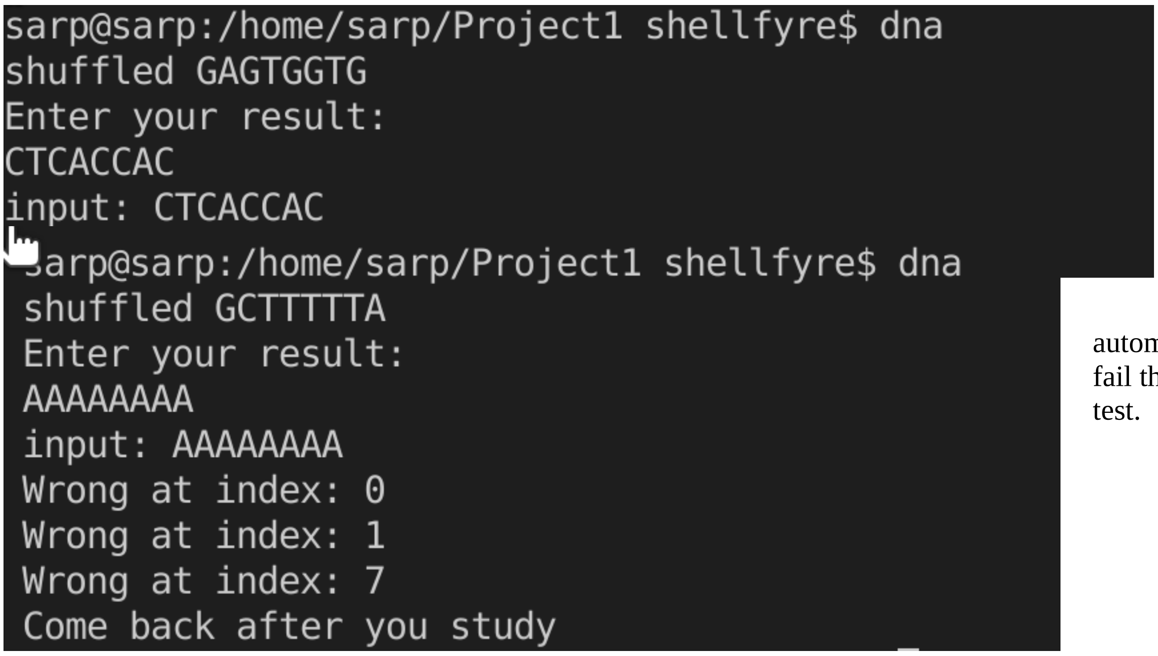
Dna command (sarp) :

This command shuffles a dna string. Whenever you run the dna command it gives you a new string of nucleotides. After it gives you a random dna string , it waits for user to enter the other sde of the dna string. For example program gives you string AGGCTGC. User should enter TCCGACG because of basic biology. Adenin combines with Thymin and Guanin combines with Cytosin. If you mismatch some parts



```
sarp@sarp:/home/sarp/Project1 shellfyre$ dna
shuffled GTGATAGC
Enter your result:
aaaaaaaaa
You failed
```

program give you the wrong indexes. If you mismatch the length of the dna string you

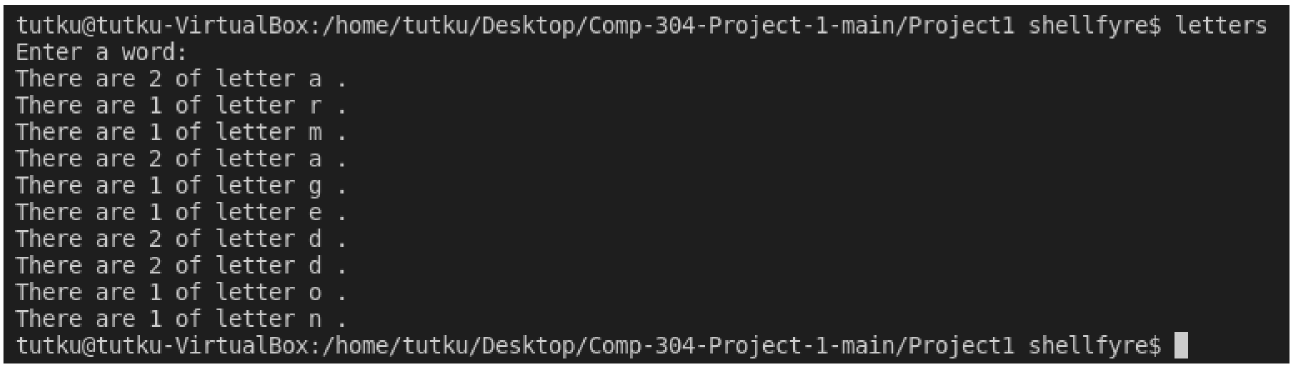


```
sarp@sarp:/home/sarp/Project1 shellfyre$ dna
shuffled GAGTGGTG
Enter your result:
CTCACCAC
input: CTCACCAC
sarp@sarp:/home/sarp/Project1 shellfyre$ dna
shuffled GCTTTTTA
Enter your result:
AAAAAAAA
input: AAAAAAAA
Wrong at index: 0
Wrong at index: 1
Wrong at index: 7
Come back after you study
```

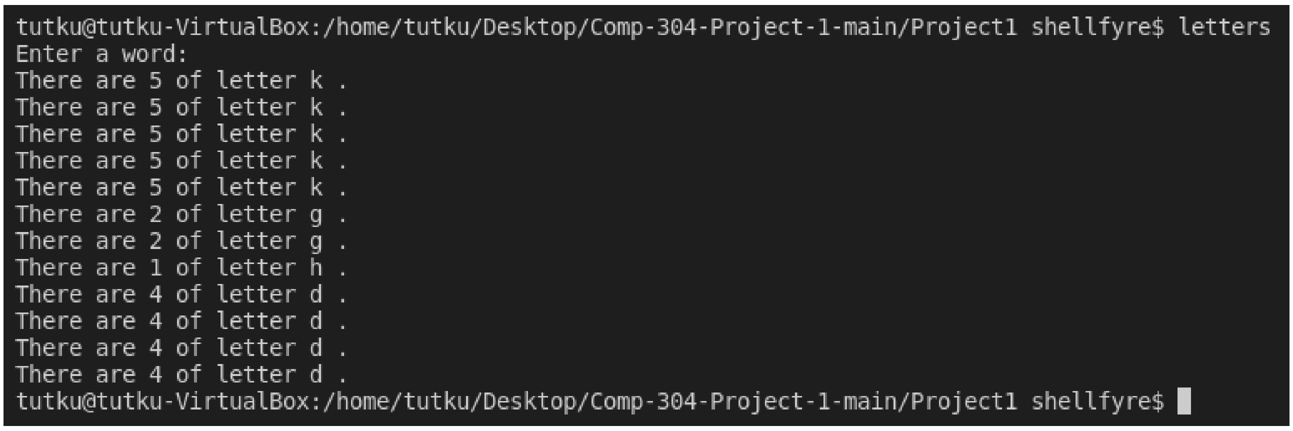
automatically fail the dna test.

Letters command (Tutku) :

For this command I implemented a function that takes a string as input from the user. After taking the input, the function takes every character one by one and counts how many of that character is in the entire string, and prints the amount at the end for every different character. For example: if the first letter is “a”, the code keeps 1 as the amount for character “a” and checks the rest of the word to see if there is any other “a”s. If not, code prints the amount of letter “a” and moves onto the second letter. The code does this operation until all the letters are checked individually.



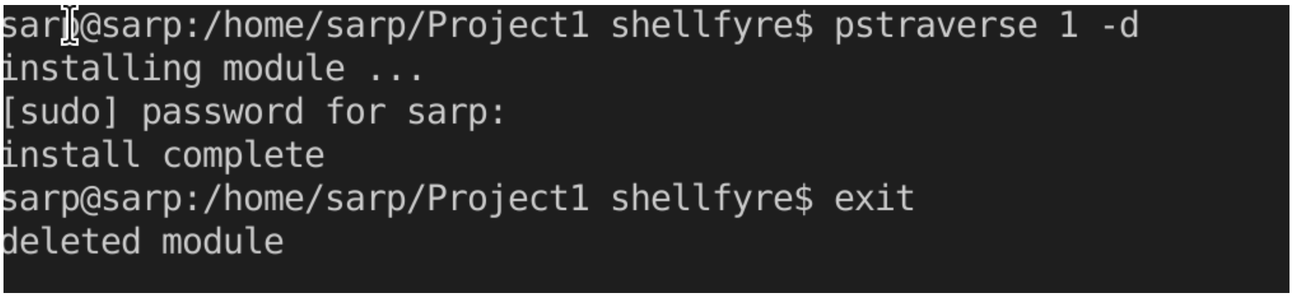
```
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$ letters
Enter a word:
There are 2 of letter a .
There are 1 of letter r .
There are 1 of letter m .
There are 2 of letter a .
There are 1 of letter g .
There are 1 of letter e .
There are 2 of letter d .
There are 2 of letter d .
There are 1 of letter o .
There are 1 of letter n .
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$
```



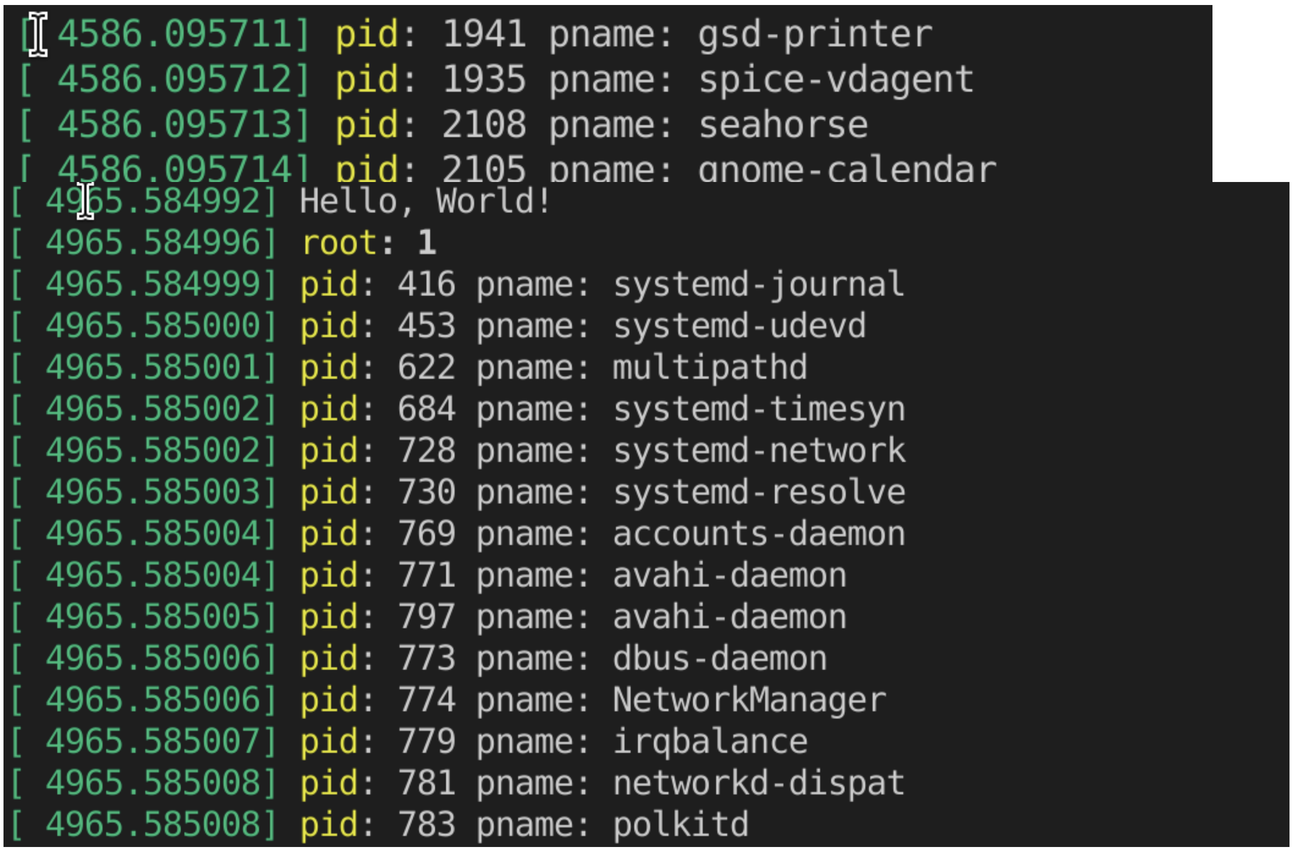
```
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$ letters
Enter a word:
There are 5 of letter k .
There are 5 of letter k .
There are 5 of letter k .
There are 5 of letter k .
There are 5 of letter k .
There are 2 of letter g .
There are 2 of letter g .
There are 1 of letter h .
There are 4 of letter d .
There are 4 of letter d .
There are 4 of letter d .
There are 4 of letter d .
tutku@tutku-VirtualBox:/home/tutku/Desktop/Comp-304-Project-1-main/Project1 shellfyre$ 
```

Part 3 Kernel Module:

In order to load the kernel module I used the exec function and insmod. It runs the bfs or dfs algorithm according to the given input in kernel space. Program deletes the module when "exit" command is typed in to the shell.



```
sarp@sarp:/home/sarp/Project1 shellfyre$ pstraverse 1 -d
installing module ...
[sudo] password for sarp:
install complete
sarp@sarp:/home/sarp/Project1 shellfyre$ exit
deleted module
```



```
[ 4586.095711] pid: 1941 pname: gsd-printer
[ 4586.095712] pid: 1935 pname: spice-vdagent
[ 4586.095713] pid: 2108 pname: seahorse
[ 4586.095714] pid: 2105 pname: gnome-calendar
[ 4965.584992] Hello, World!
[ 4965.584996] root: 1
[ 4965.584999] pid: 416 pname: systemd-journal
[ 4965.585000] pid: 453 pname: systemd-udevd
[ 4965.585001] pid: 622 pname: multipathd
[ 4965.585002] pid: 684 pname: systemd-timesyn
[ 4965.585002] pid: 728 pname: systemd-network
[ 4965.585003] pid: 730 pname: systemd-resolve
[ 4965.585004] pid: 769 pname: accounts-daemon
[ 4965.585004] pid: 771 pname: avahi-daemon
[ 4965.585005] pid: 797 pname: avahi-daemon
[ 4965.585006] pid: 773 pname: dbus-daemon
[ 4965.585006] pid: 774 pname: NetworkManager
[ 4965.585007] pid: 779 pname: irqbalance
[ 4965.585008] pid: 781 pname: networkd-dispat
[ 4965.585008] pid: 783 pname: polkitd
```