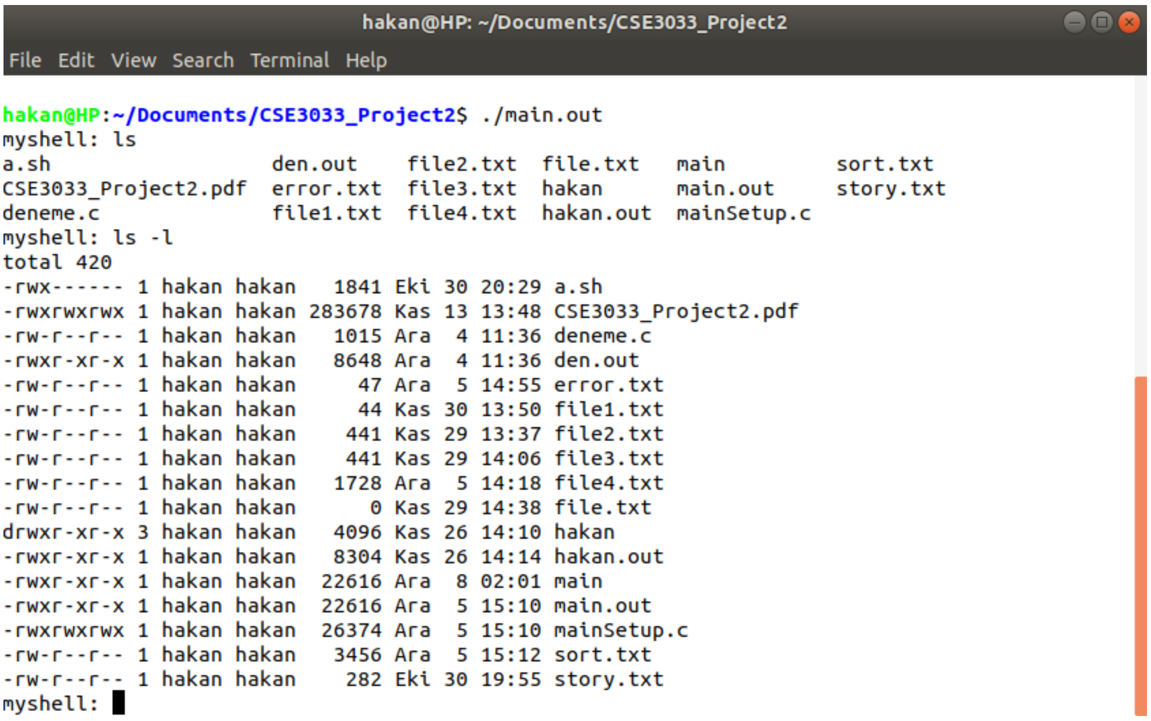
We have completed this project including bonus section.

PART A:

When the program name gets as an argument, we will navigate on files in PATH environment variable until we find the program name in our shell. After the correct directory is found, the program will run with execl(). In Figure A.1, ls and ls -l programs were called respectively.

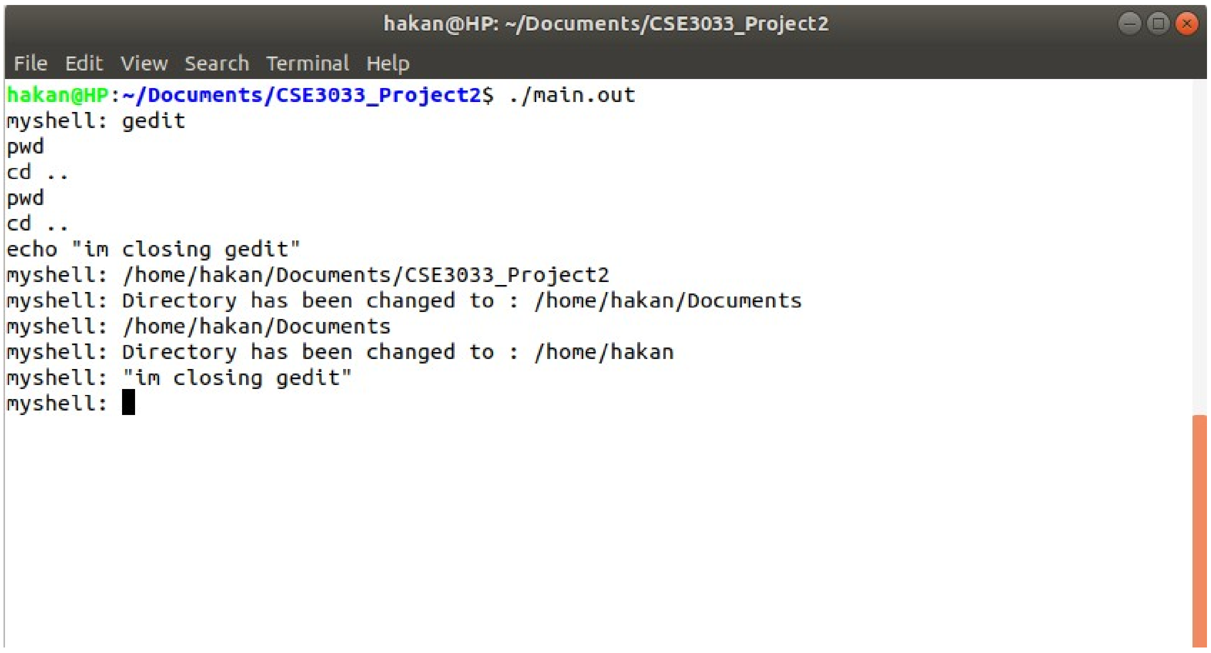

hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help

```
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: ls
a.sh                  den.out    file2.txt  file.txt   main         sort.txt
CSE3033_Project2.pdf  error.txt  file3.txt  hakan      main.out     story.txt
deneme.c              file1.txt  file4.txt  hakan.out  mainSetup.c
myshell: ls -l
total 420
-rwx------ 1 hakan hakan   1841 Eki 30 20:29 a.sh
-rwxrwxrwx 1 hakan hakan 283678 Kas 13 13:48 CSE3033_Project2.pdf
-rw-r--r-- 1 hakan hakan   1015 Ara  4 11:36 deneme.c
-rwxr-xr-x 1 hakan hakan   8648 Ara  4 11:36 den.out
-rw-r--r-- 1 hakan hakan     47 Ara  5 14:55 error.txt
-rw-r--r-- 1 hakan hakan     44 Kas 30 13:50 file1.txt
-rw-r--r-- 1 hakan hakan    441 Kas 29 13:37 file2.txt
-rw-r--r-- 1 hakan hakan    441 Kas 29 14:06 file3.txt
-rw-r--r-- 1 hakan hakan   1728 Ara  5 14:18 file4.txt
-rw-r--r-- 1 hakan hakan      0 Kas 29 14:38 file.txt
drwxr-xr-x 3 hakan hakan   4096 Kas 26 14:10 hakan
-rwxr-xr-x 1 hakan hakan   8304 Kas 26 14:14 hakan.out
-rwxr-xr-x 1 hakan hakan  22616 Ara  8 02:01 main
-rwxr-xr-x 1 hakan hakan  22616 Ara  5 15:10 main.out
-rwxrwxrwx 1 hakan hakan  26374 Ara  5 15:10 mainSetup.c
-rw-r--r-- 1 hakan hakan   3456 Ara  5 15:12 sort.txt
-rw-r--r-- 1 hakan hakan    282 Eki 30 19:55 story.txt
myshell: 
```

Figure A.1

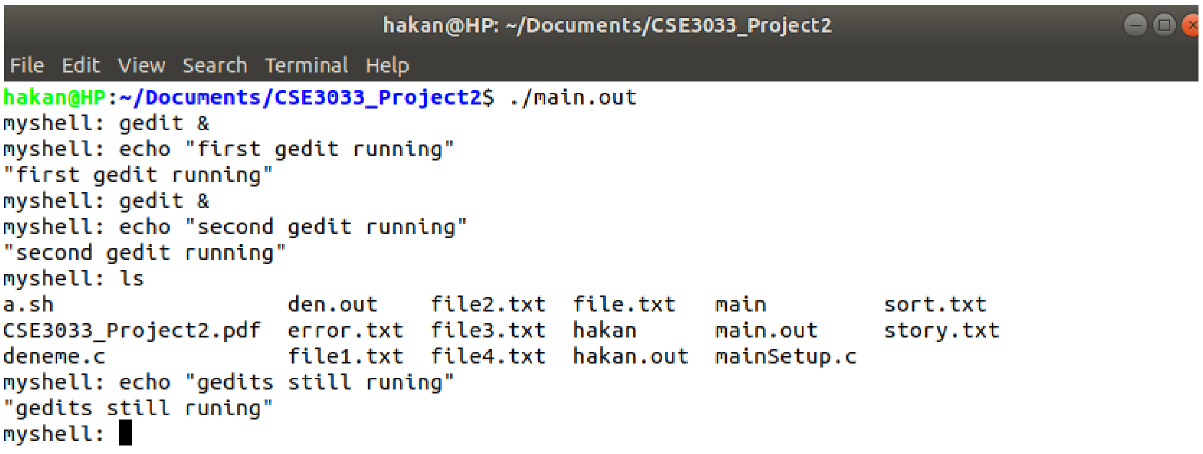
If a process is run in the foreground (without &), the main process will wait for the child process. while the child process is waiting, the main process will not be able to run another process but it will be able to run after the child process is finished. As we can see in Figure A.2, pwd and cd .. process is on hold while gedit running in the background. When we closed the gedit, pwd and cd .. runs respectively.

On the other hand, if a process is run in the background (with &), the main process not need to wait for the child process is finished. We can run processes as much as we want in the background and no one needs to wait for each other. In Figure A.3 our shell can run other process while child processes running in the background.


hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help

```
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: gedit
pwd
cd ..
pwd
cd ..
echo "im closing gedit"
myshell: /home/hakan/Documents/CSE3033_Project2
myshell: Directory has been changed to : /home/hakan/Documents
myshell: /home/hakan/Documents
myshell: Directory has been changed to : /home/hakan
myshell: "im closing gedit"
myshell: 
```

Figure A.2


hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help

```
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: gedit &
myshell: echo "first gedit running"
"first gedit running"
myshell: gedit &
myshell: echo "second gedit running"
"second gedit running"
myshell: ls
a.sh                  den.out    file2.txt  file.txt   main        sort.txt
CSE3033_Project2.pdf  error.txt  file3.txt  hakan      main.out    story.txt
deneme.c              file1.txt  file4.txt  hakan.out  mainSetup.c
myshell: echo "gedits still runing"
"gedits still runing"
myshell: 
```

Figure A.3

PART B:

We need to create built-in command in part B. This built-in command uses linux commands to invoke different names. In short, we will add a nickname to the current command names. Figure B.1 explain of using alias.

In first condition, We checked whether the first argument (args[0]) in argument array is “alias” and the second argument (args[1]) is “-l”.

```
if ( strcmp(args[0],"alias") == 0 && strcmp(args[1],"-l") == 0 )
```

then invoke print() function. This function is print to all alias in the list.

The second conditon only controls args[0] is “alias” or not.

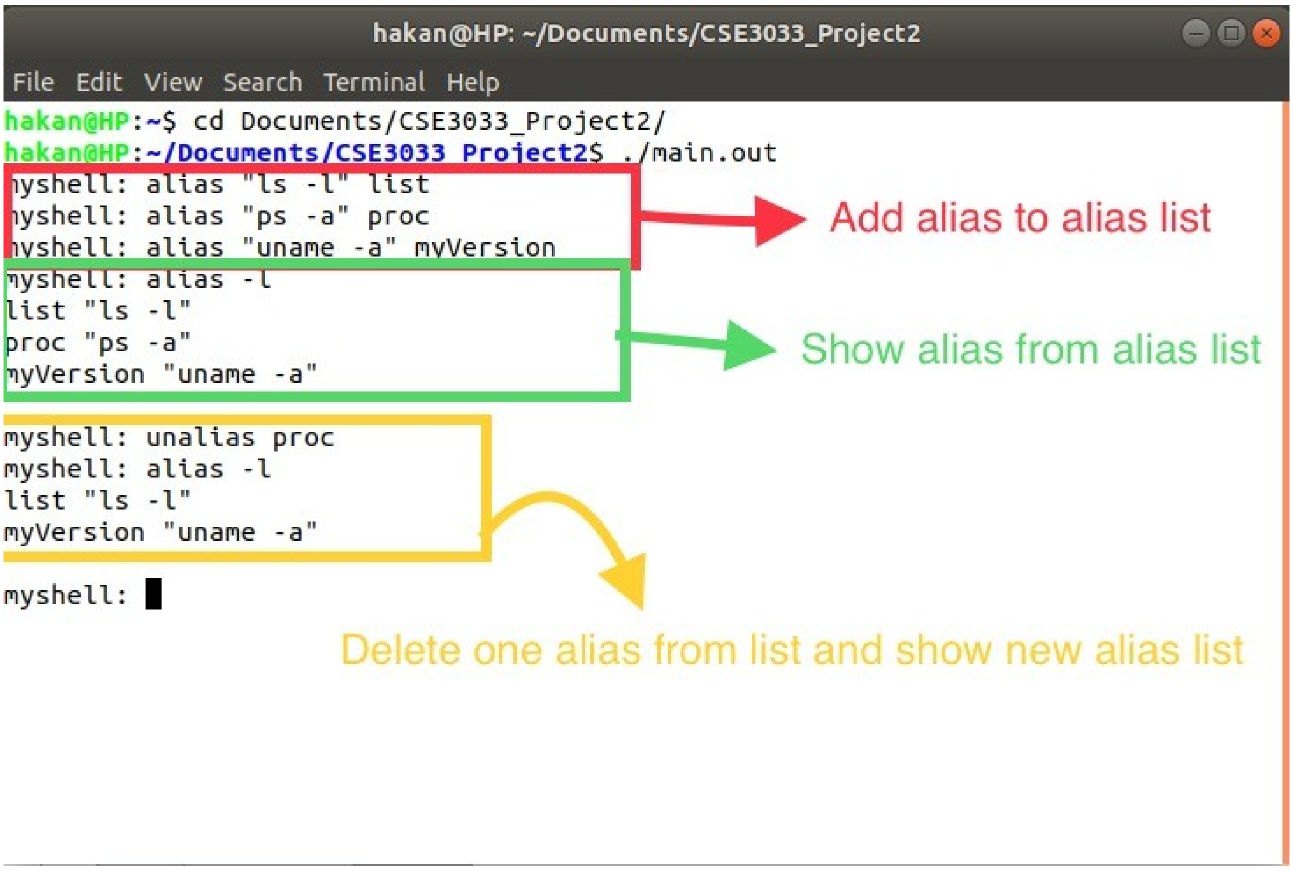
```
else if ( strcmp(args[0],"alias") == 0 )
```

then invoke clearQuoates(args) and insert(args) functions respectively to add alias to alias list.

The third condition controls the args[0] is unalias.

```
else if ( strcmp(args[0],"unalias") == 0 )
```

then invoke delete(args) function.

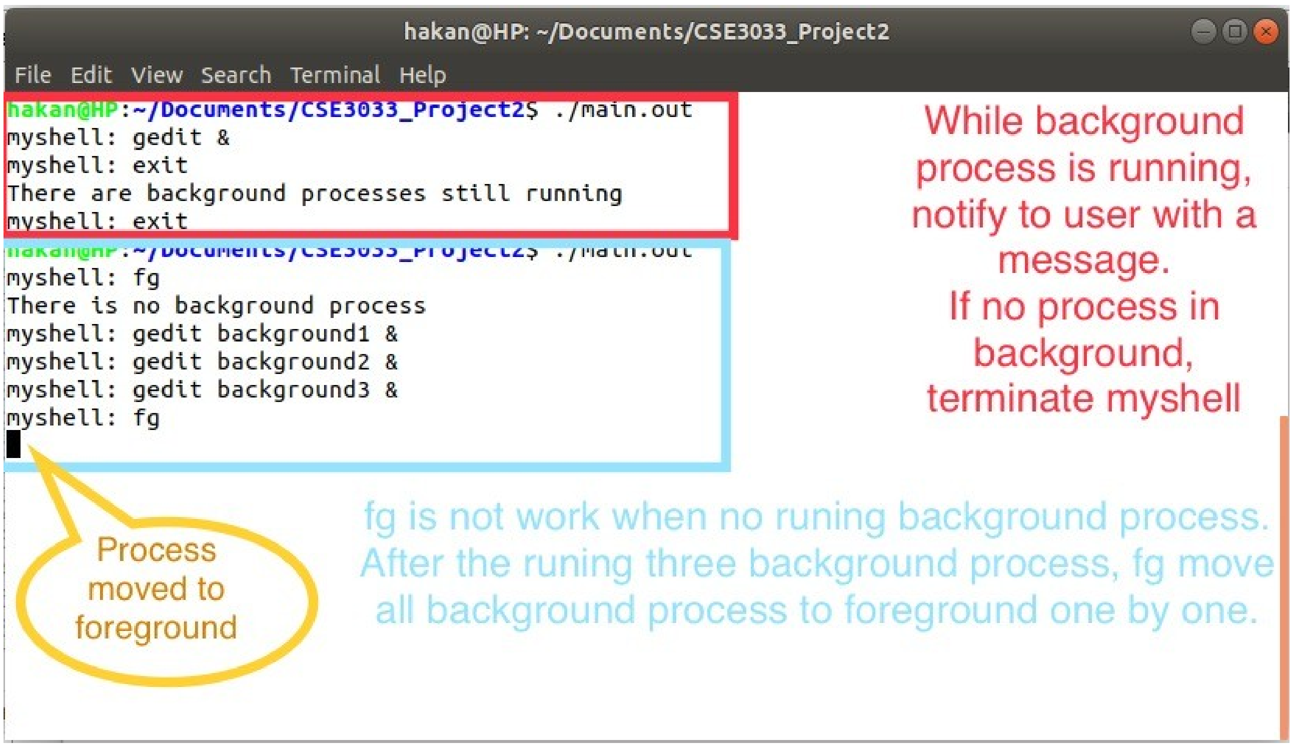


```
hakan@HP:~$ cd Documents/CSE3033_Project2/
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: alias "ls -l" list
myshell: alias "ps -a" proc
myshell: alias "uname -a" myVersion
myshell: alias -l
list "ls -l"
proc "ps -a"
myVersion "uname -a"

myshell: unalias proc
myshell: alias -l
list "ls -l"
myVersion "uname -a"

myshell: 
```

hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help
Add alias to alias list
Show alias from alias list
Delete one alias from list and show new alias list

Figure B.1

Our shell program also runs the fg, exit, control+z and clear commands. In Figure B.2, the terminal output of the fg command and exit command is shown. (we didn't put the screenshot display of the clear and ^Z command because it was unsatisfactory but it works on code)



```
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: gedit &
myshell: exit
There are background processes still running
myshell: exit
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: fg
There is no background process
myshell: gedit background1 &
myshell: gedit background2 &
myshell: gedit background3 &
myshell: fg
```

While background process is running, notify to user with a message.
If no process in background, terminate myshell

fg is not work when no runing background process.
After the runing three background process, fg move all background process to foreground one by one.

hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help
Process moved to foreground

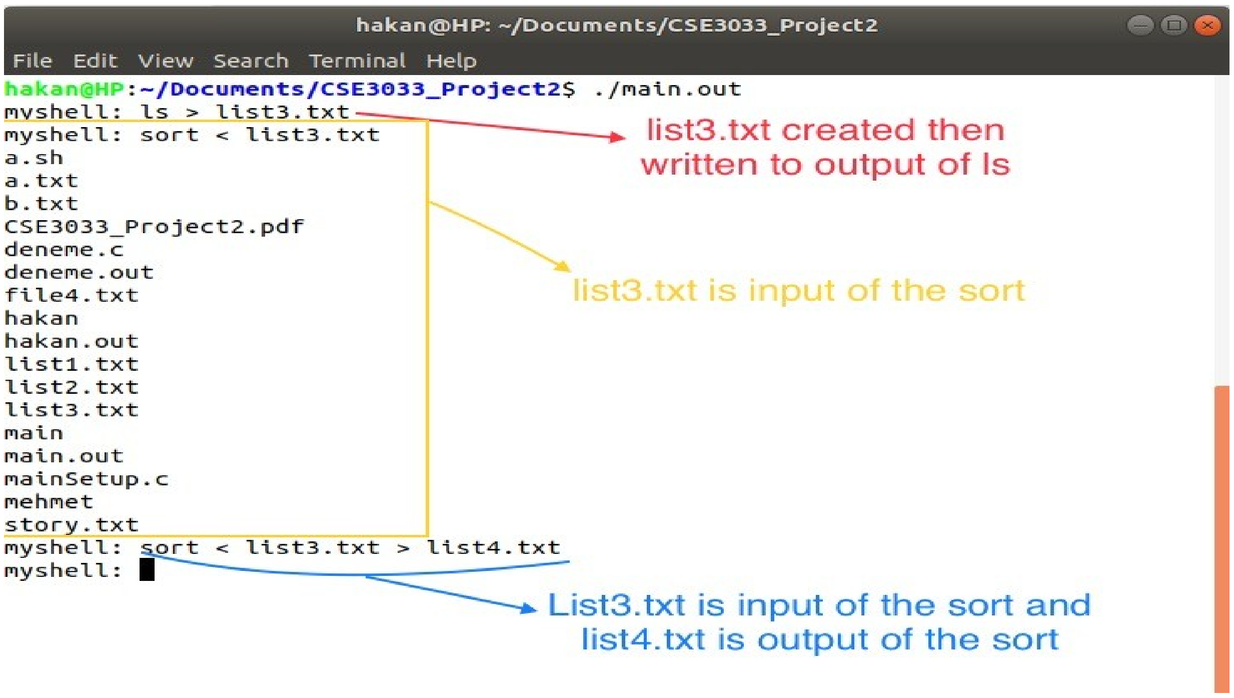
Figure B.2

PART C:

In the part C, we have made the I/O redirection.

hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: ls > list1.txt *// list1.txt is created and output of ls is written to list1.txt*
myshell: ls > list2.txt *// list2.txt is created and output of ls is written to list2.txt*
myshell: ps > list1.txt *// output of ps is overwritten to list1.txt*
myshell: ps >> list2.txt *// oputput of ps is appended to list2.txt*
myshell:

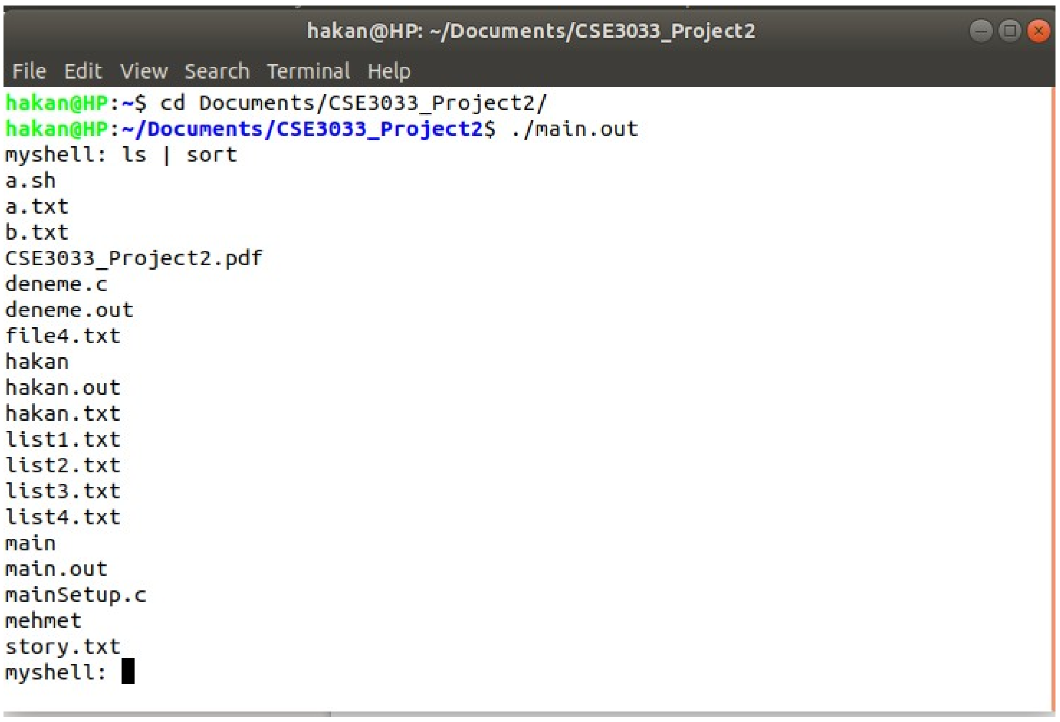
After the run these commands, The list1.txt contains only the output of ps. Otherwise list2.txt contains output of ls and output of ps.


hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help
list3.txt created then written to output of ls
list3.txt is input of the sort
List3.txt is input of the sort and list4.txt is output of the sort

```
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: ls > list3.txt
myshell: sort < list3.txt
a.sh
a.txt
b.txt
CSE3033_Project2.pdf
deneme.c
deneme.out
file4.txt
hakan
hakan.out
list1.txt
list2.txt
list3.txt
main
main.out
mainSetup.c
mehmet
story.txt
myshell: sort < list3.txt > list4.txt
myshell: 
```

After the run these commands, the sort command take input from the list3.txt. The list4.txt will include the output of ps in a sorted.

BONUS PART:
In bonus part, we implement pipe operation to our shell. As you can see ls and sort command runs concurrently.


hakan@HP: ~/Documents/CSE3033_Project2
File Edit View Search Terminal Help

```
hakan@HP:~$ cd Documents/CSE3033_Project2/
hakan@HP:~/Documents/CSE3033_Project2$ ./main.out
myshell: ls | sort
a.sh
a.txt
b.txt
CSE3033_Project2.pdf
deneme.c
deneme.out
file4.txt
hakan
hakan.out
hakan.txt
list1.txt
list2.txt
list3.txt
list4.txt
main
main.out
mainSetup.c
mehmet
story.txt
myshell: 
```