Introduction :

This project consists of designing a C program to serve as a shell interface that accepts user commands and then executes each command in a separate process. Your implementation will support input and output redirection, as well as pipes between a pair of commands. Completing this project will involve using the lunix fork(), exec(), wait(), dup2(), and pipe() ...

1. Installation

1-installing VM VirtualBox and installing ubuntu

2- create a virtual machine

3- open the terminal and write "**sudo apt update**". The « **sudo apt update** » command is used to download package information from all configured sources.

4- The default Ubuntu repositories contain a meta-package named "**build-essential**" that includes the GNU compiler collection, GNU debugger, and other development libraries and tools required for compiling software.

Write "**sudo apt build-essentia**l": The command installs a lot of packages, including gcc, g++ and make.

5- To validate that the GCC compiler is successfully installed, use the

"**gcc –version**" command which prints the GCC version.

2. Programming:

1- To create a folder, write "mkdir folder-name"

2- Create a file with extension c "touch filename.c".

3- First, we include the library like "#include <studio.h>" …

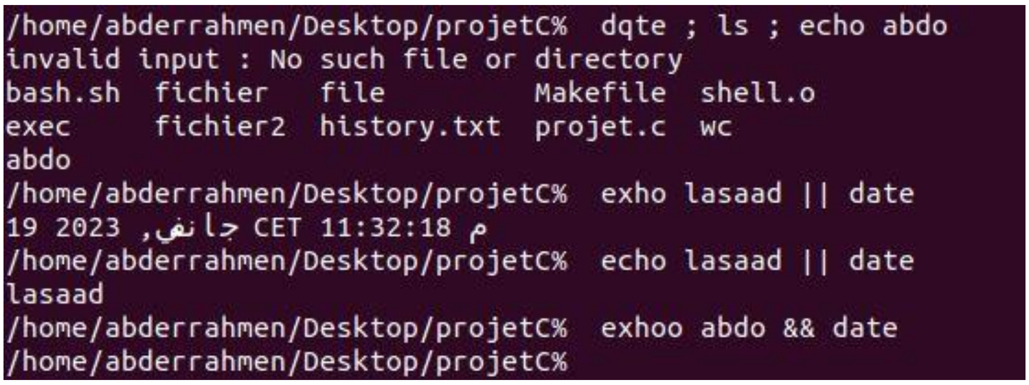
4- We start with our main by declaring a bunch of variables like strings, pointers …

5- to execute makeFile

7- Now we will choose interactive mode

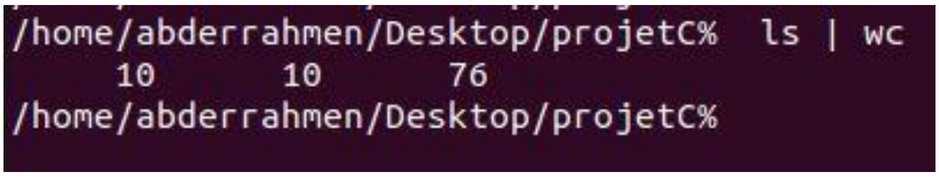
8- In addition, we present our composite function:

when you enter your command the program will take it and put it in a series of characters that will be analyse it one by one first using the “line” variable we test for specific characters like “:”, “&&”, ”||”, “|”, “>” after we separate them from the space with the help of strtok with delimiter being" " . if we found the “;” we increment and we send each character before and after it to be executed



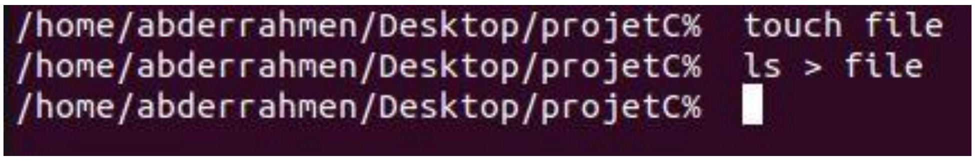
```
/home/abderrahmen/Desktop/projetC%  dqte ; ls ; echo abdo
invalid input : No such file or directory
bash.sh  fichier   file         Makefile  shell.o
exec     fichier2  history.txt  projet.c  wc
abdo
/home/abderrahmen/Desktop/projetC%  exho lasaad || date
19 2023 ,جانفي CET 11:32:18 م
/home/abderrahmen/Desktop/projetC%  echo lasaad || date
lasaad
/home/abderrahmen/Desktop/projetC%  exhoo abdo && date
/home/abderrahmen/Desktop/projetC%
```

Additionally, the pipe | command allows you to pipe the output of one command to another as mentioned in the screenshot below.



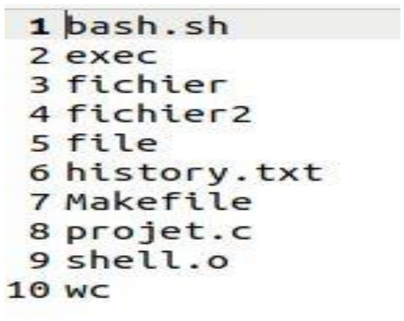
```
/home/abderrahmen/Desktop/projetC%  ls | wc
     10      10      76
/home/abderrahmen/Desktop/projetC%
```

In comparison to the last command is The > operator is used to change the source of the standard input. This method can be useful to take user input from file contents and store it in program buffer with the help of the function fork and dup and dup2 where the out STDOUT will be put in a file with the name is given by the user after the “>” as shown in the screenshot of the output below.



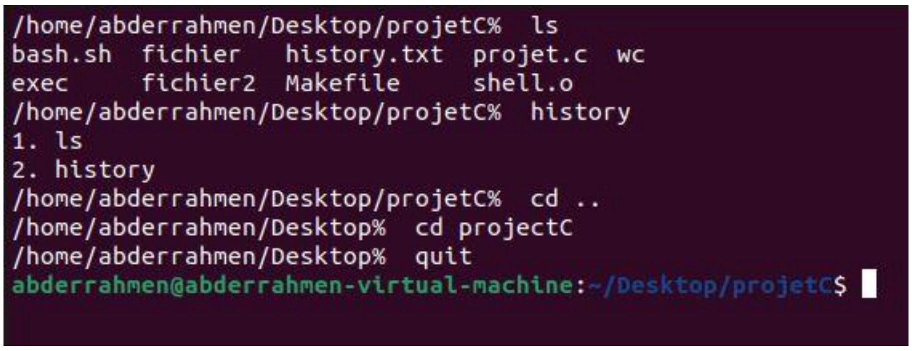
```
/home/abderrahmen/Desktop/projetC%  touch file
/home/abderrahmen/Desktop/projetC%  ls > file
/home/abderrahmen/Desktop/projetC%
```

*



```
bash.sh
exec
fichier
fichier2
file
history.txt
Makefile
projet.c
shell.o
wc
```

At last the quit function terminates the program by specifying a return code. This return code, passed as a parameter of the quit   function, is used to specify how the program ends. A null value (0 or constantTHANKS_FOR_USING_THE_SHELL!)



```
/home/abderrahmen/Desktop/projetC%  ls
bash.sh  fichier   history.txt  projet.c  wc
exec     fichier2  Makefile     shell.o
/home/abderrahmen/Desktop/projetC%  history
1. ls
2. history
/home/abderrahmen/Desktop/projetC%  cd ..
/home/abderrahmen/Desktop%  cd projectC
/home/abderrahmen/Desktop%  quit
abderrahmen@abderrahmen-virtual-machine:~/Desktop/projetC$
```

setup guide:

BEFORE YOU DO ANY OF THE STEPS BELOW YOU SHOULD CREATE A FOLDER AND ACCESS IT IN THE TERMINAL

Using cd folder name

1. We should download the code in file with extension.c

2. Debug the program with the help of the command gcc filename.c -o filename

3. Run the code with the help of the command ./filename

4. Type some command simple complex one

5. For the batch file you download the file with the command in it and save it if you want with extension ".sh"

6. To run it type ./filename.extension ("the extension you chose")

7. For the make file download the script then save it as "Makefile"

8. To run it use the command "make" for it to generate a file with extension ".o"

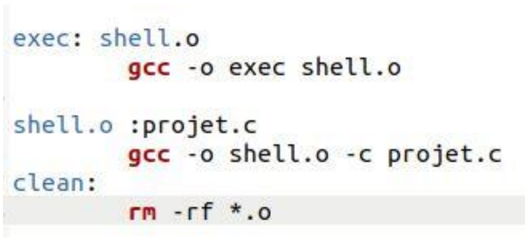
9. To remove the file you generated use the command "make clean".

10. To run the file.c type "make exec".

Makefile

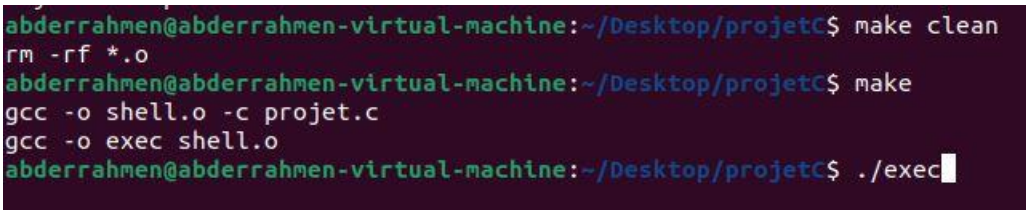
A makefile can be used to control the build process of an object file such as "file.o" in Linux. The makefile would typically contain a rule that specifies how the object file should be built from its corresponding source file, for example:

- “make clean” it will remove the file specified
- “make ” it will compile the file specified
- “./exec” it will execute the file specified



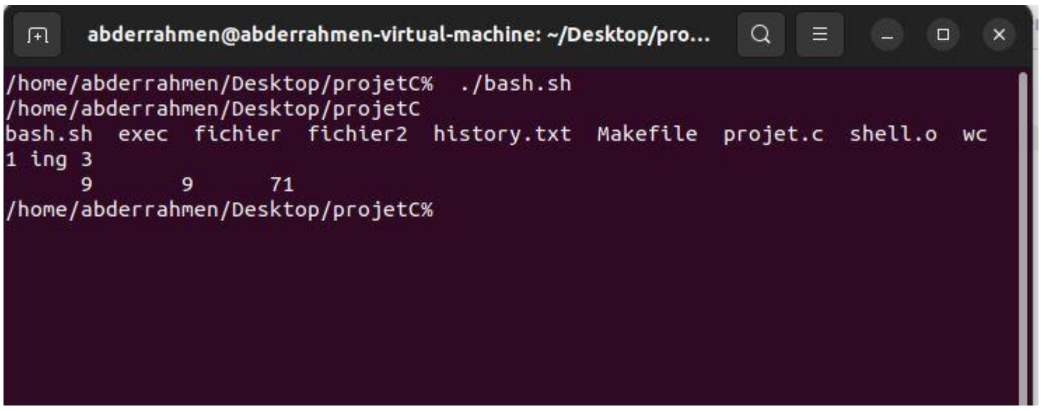
```
exec: shell.o
	gcc -o exec shell.o

shell.o :projet.c
	gcc -o shell.o -c projet.c
clean:
	rm -rf *.o
```

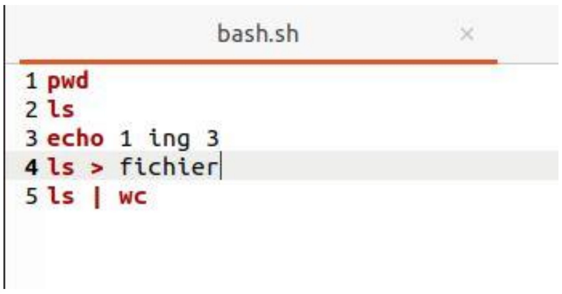


```
abderrahmen@abderrahmen-virtual-machine:~/Desktop/projetC$ make clean
rm -rf *.o
abderrahmen@abderrahmen-virtual-machine:~/Desktop/projetC$ make
gcc -o shell.o -c projet.c
gcc -o exec shell.o
abderrahmen@abderrahmen-virtual-machine:~/Desktop/projetC$ ./exec
```

BACH MODE:

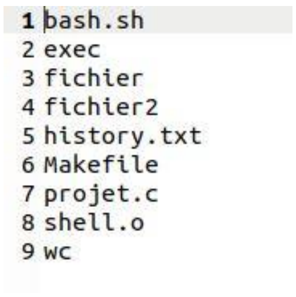


```
/home/abderrahmen/Desktop/projetC%  ./bash.sh
/home/abderrahmen/Desktop/projetC
bash.sh   exec   fichier   fichier2   history.txt   Makefile   projet.c   shell.o   wc
1 ing 3
      9       9      71
/home/abderrahmen/Desktop/projetC%
```



bash.sh

```
pwd
ls
echo 1 ing 3
ls > fichier
ls | wc
```



```
bash.sh
exec
fichier
fichier2
history.txt
Makefile
projet.c
shell.o
wc
```