Faculty of Engineering
Alexandria University
CSED 2024
Operating Systems

OS
Lab 1

Bassant Yasser Salah 19017262

<u>**1-Simple Shell:**</u>

<u>**Problem Statement:**</u>

It is required to implement a Unix shell program. A shell is simply a program that conveniently allows you to run other programs. Read up on your favorite shell to see what it does.

<u>**Explanation:**</u>

<u>**Split**</u>:

is a function that is used to parse given arguments from the user to be able to distinguish which function will be executed and which one won't. Splitting happen be strtok function (spaces as a parameter);

<u>**removqu():**</u>

which used to see to remove any double quotes from the argument after the command itself.

<u>**execution():**</u>

The method that contains the fork that help in execution all other methods is far from the main method itself. First, create a child with it's id that represents its state whether it's failure (negative), (0 represent the existence of the child class), (otherwise means that parent class is executed). Those states help us to determine which class should wait (parent class). In case of the succession of creating the child, there is a condition to compare the first command given in order to determine which method will be executed.

Not built in function will be executed using execvp method which is used to search for the given command in another built in function if it found return 1 and it is a successful operation other than that it is considered a failure.

At the end of the method there is a flag that helps in determining the waiting period for the parent until the child finishes his operations. This flag is determined by the presence of & symbol at the end.

<u>Shell()</u>:

it contains only a loop that contains taking of the commands for users and parsing it. This loop terminates only in case the user entered exit as a command or there is an error in execution either wrong command that doesn't exist or failure in creating a child class. It is responsible for determining which case will be executed according to the first parsed command.

<u>cdd():</u>

method responsible for execution of one of the built in functions which is cd(). We enter the path of the directory or the name of the file wanted into chdir() which returns 0 if it is a succession in finding the directory or it is a valid command, otherwise it returns -1;

<u>expo():</u>

Method for exporting function. Before storing any variables we have to make sure that there are no quotes to store it correctly and to make sure that we separate values from the variables using strtok() method and " =",command [1] is a parameter for it. After making sure that we have fulfilled previous conditions. The value will be stored using **setenv(variable,value,1)**, 1 is representing writing or storing the value.

<u>echo():</u>

Method for printing or evaluating given parameters.if there is $, the method will remove this sign and return the value of the given argument by getev(argument), if there is no presence of $ sign then the argument will be printed as it is.

log-termination():

Opening a file **fopen()** method to save that succession of the termination of the child. The file closes after exiting from the console (shell).

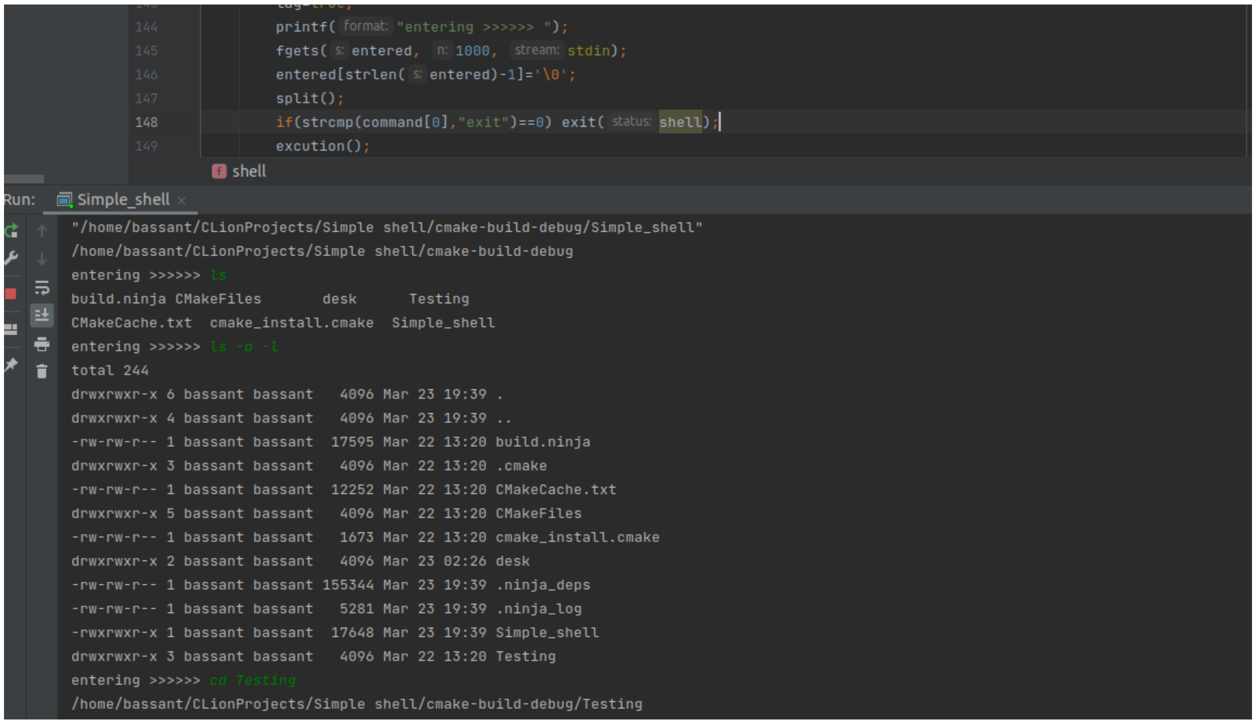
Signal handler():

Method to hand signals of the child as it is sent to the parent to make sure that it is terminated to avoid the zombie process.

eval():

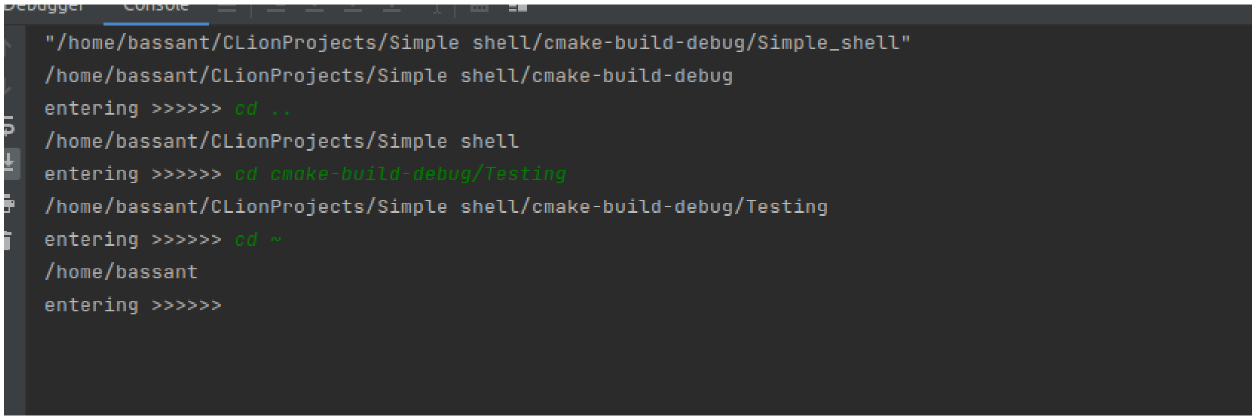
This method used to evaluate $x in arguments by taking the name of the variable and get back the value variable.

Test Cases:

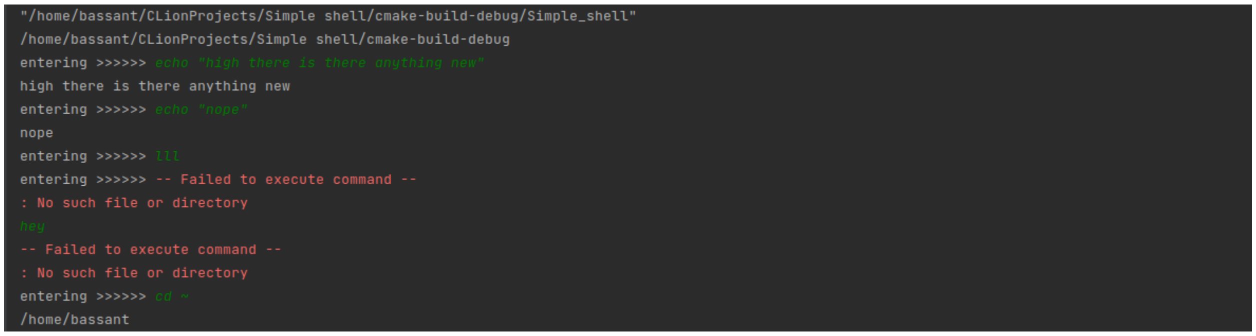


```
        printf("entering >>>>>> ");
        fgets(entered, 1000, stdin);
        entered[strlen(entered)-1]='\0';
        split();
        if(strcmp(command[0],"exit")==0) exit(shell);
        excution();
```

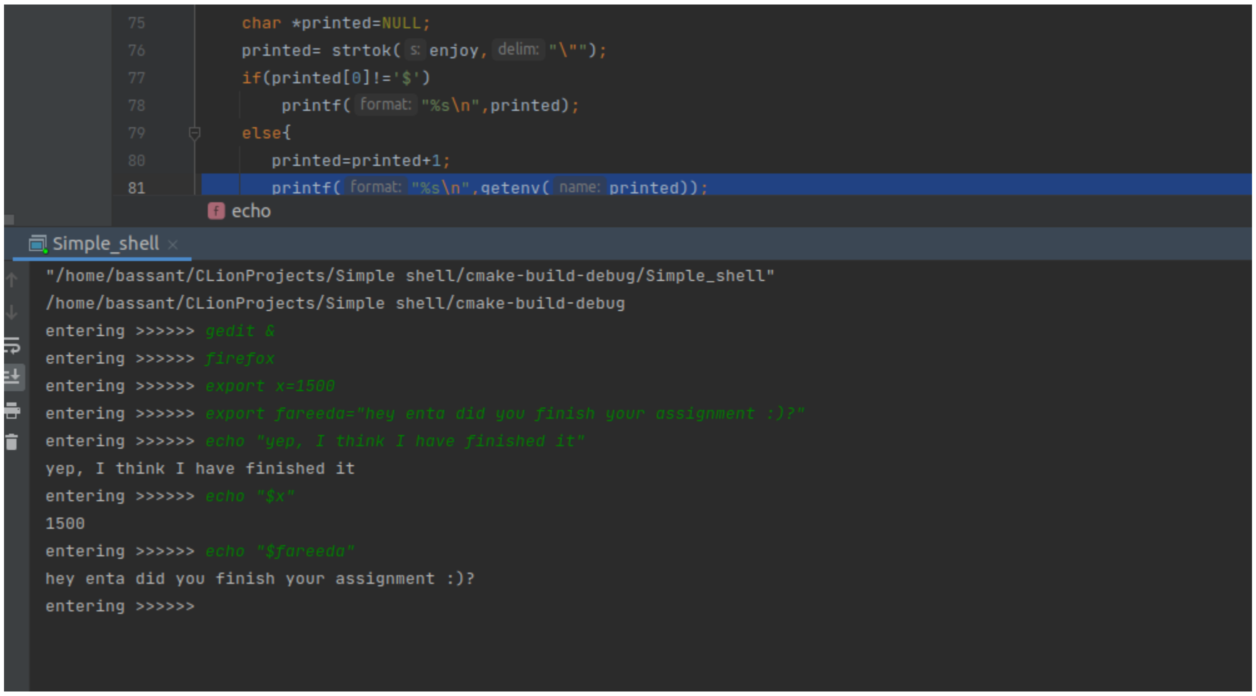
```
"/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Simple_shell"
/home/bassant/CLionProjects/Simple shell/cmake-build-debug
entering >>>>>> ls
build.ninja CMakeFiles       desk      Testing
CMakeCache.txt  cmake_install.cmake  Simple_shell
entering >>>>>> ls -a -l
total 244
drwxrwxr-x 6 bassant bassant   4096 Mar 23 19:39 .
drwxrwxr-x 4 bassant bassant   4096 Mar 23 19:39 ..
-rw-rw-r-- 1 bassant bassant  17595 Mar 22 13:20 build.ninja
drwxrwxr-x 3 bassant bassant   4096 Mar 22 13:20 .cmake
-rw-rw-r-- 1 bassant bassant  12252 Mar 22 13:20 CMakeCache.txt
drwxrwxr-x 5 bassant bassant   4096 Mar 22 13:20 CMakeFiles
-rw-rw-r-- 1 bassant bassant   1673 Mar 22 13:20 cmake_install.cmake
drwxrwxr-x 2 bassant bassant   4096 Mar 23 02:26 desk
-rw-rw-r-- 1 bassant bassant 155344 Mar 23 19:39 .ninja_deps
-rw-rw-r-- 1 bassant bassant   5281 Mar 23 19:39 .ninja_log
-rwxrwxr-x 1 bassant bassant  17648 Mar 23 19:39 Simple_shell
drwxrwxr-x 3 bassant bassant   4096 Mar 22 13:20 Testing
entering >>>>>> cd Testing
/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Testing
```



```
"/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Simple_shell"
/home/bassant/CLionProjects/Simple shell/cmake-build-debug
entering >>>>>> cd ..
/home/bassant/CLionProjects/Simple shell
entering >>>>>> cd cmake-build-debug/Testing
/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Testing
entering >>>>>> cd ~
/home/bassant
entering >>>>>>
```

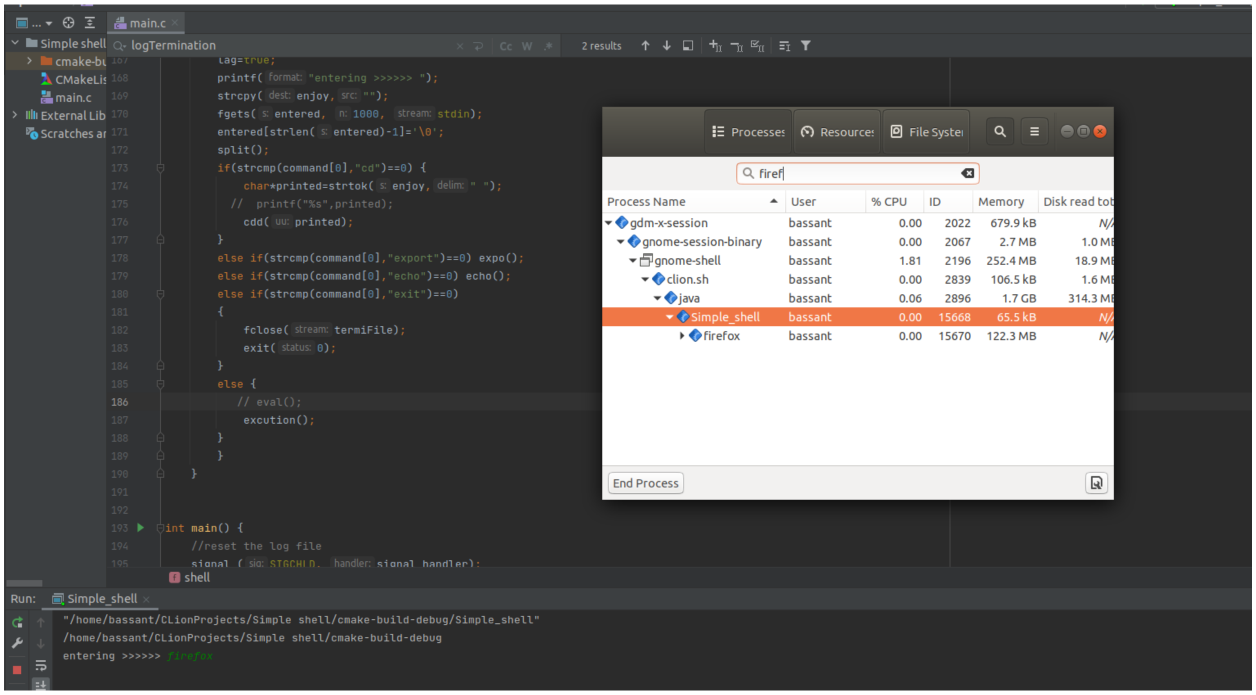


```
"/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Simple_shell"
/home/bassant/CLionProjects/Simple shell/cmake-build-debug
entering >>>>>> echo "high there is there anything new"
high there is there anything new
entering >>>>>> echo "nope"
nope
entering >>>>>> lll
entering >>>>>> -- Failed to execute command --
: No such file or directory
hey
-- Failed to execute command --
: No such file or directory
entering >>>>>> cd ~
/home/bassant
```



```
    char *printed=NULL;
    printed= strtok(enjoy,"\"");
    if(printed[0]!='$')
        printf("%s\n",printed);
    else{
        printed=printed+1;
        printf("%s\n",getenv(printed));
```

```
"/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Simple_shell"
/home/bassant/CLionProjects/Simple shell/cmake-build-debug
entering >>>>>> gedit &
entering >>>>>> firefox
entering >>>>>> export x=1500
entering >>>>>> export fareeda="hey enta did you finish your assignment :)?"
entering >>>>>> echo "yep, I think I have finished it"
yep, I think I have finished it
entering >>>>>> echo "$x"
1500
entering >>>>>> echo "$fareeda"
hey enta did you finish your assignment :)?
entering >>>>>>
```



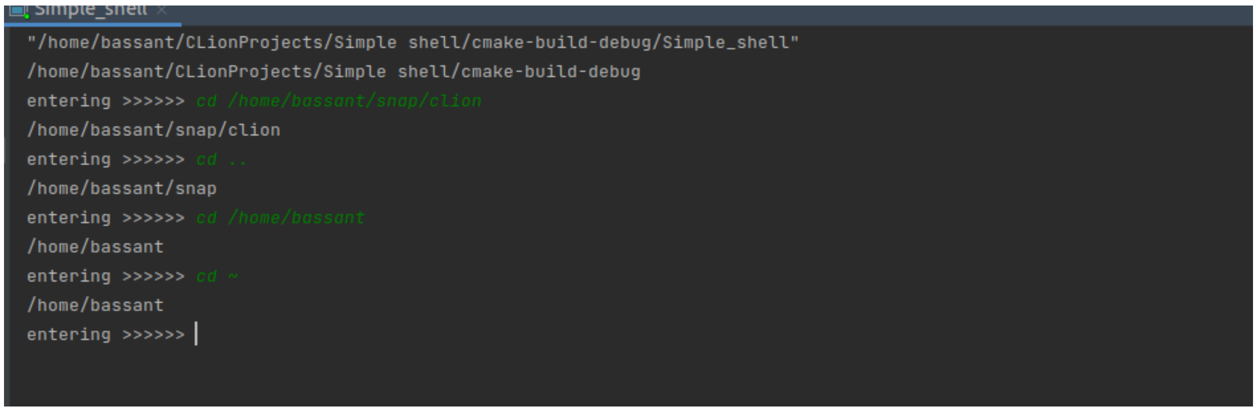
```c
        lag=true;
        printf("entering >>>>>> ");
        strcpy(enjoy,"");
        fgets(entered, 1000, stdin);
        entered[strlen(entered)-1]='\0';
        split();
        if(strcmp(command[0],"cd")==0) {
            char*printed=strtok(enjoy," ");
          //  printf("%s",printed);
            cdd(printed);
        }
        else if(strcmp(command[0],"export")==0) expo();
        else if(strcmp(command[0],"echo")==0) echo();
        else if(strcmp(command[0],"exit")==0)
        {
            fclose(termiFile);
            exit(0);
        }
        else {
           // eval();
            excution();
        }
        }
    }


int main() {
    //reset the log file
    signal (SIGCHLD, signal_handler);
```

firef

Process Name	User	% CPU	ID	Memory	Disk read tot
gdm-x-session	bassant	0.00	2022	679.9 kB	N/
gnome-session-binary	bassant	0.00	2067	2.7 MB	1.0 MB
gnome-shell	bassant	1.81	2196	252.4 MB	18.9 MB
clion.sh	bassant	0.00	2839	106.5 kB	1.6 MB
java	bassant	0.06	2896	1.7 GB	314.3 MB
Simple_shell	bassant	0.00	15668	65.5 kB	N/
firefox	bassant	0.00	15670	122.3 MB	N/

```
"/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Simple_shell"
/home/bassant/CLionProjects/Simple shell/cmake-build-debug
entering >>>>>> firefox
```



```
"/home/bassant/CLionProjects/Simple shell/cmake-build-debug/Simple_shell"
/home/bassant/CLionProjects/Simple shell/cmake-build-debug
entering >>>>>> cd /home/bassant/snap/clion
/home/bassant/snap/clion
entering >>>>>> cd ..
/home/bassant/snap
entering >>>>>> cd /home/bassant
/home/bassant
entering >>>>>> cd ~
/home/bassant
entering >>>>>> 
```