Simple Shell

Name:mohamed aly ahmed mohamed
Id:19016450

1.overall organization

Shell Implementation Overview

In this lab, we focus on creating a simple Linux shell capable of executing user commands. We streamline the code by following these steps:

1. Setting Up the Environment

Begin by establishing the shell's starting directory. This step ensures clarity regarding the shell's initial location.

2. Super Loop Operation

The core of our shell resides within a super loop. Here, we continuously accept user input and scrutinize it for validity. If the input indicates an exit command, the program terminates. Otherwise, it evaluates whether the user has issued a recognizable command.

3. Command Classification

Commands within our shell fall into two categories:

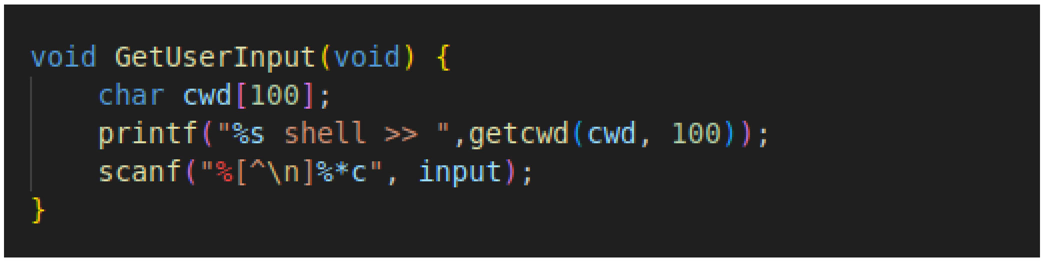
- Built-in Commands: These encompass specialized functionalities utilizing system calls such as **`getenv`**, **`setenv`**, **`export`**, **`echo`**, **`pwd`**, and **`cd`**. Each command performs specific actions like variable manipulation, directory navigation, or output display.
- Normal Commands: For executing standard commands, we fork a child process. If the fork returns a process ID of 0, we operate within the child process, utilizing **`execvp`** to execute the command. Conversely, a non-zero process ID indicates the parent process, where we manage foreground and background processes accordingly.

4. Child Process Management

To prevent zombie processes and maintain system cleanliness, we register a signal handler to monitor child processes. Upon termination, we write pertinent information, such as successful completion, to a designated log file (`log.txt`).

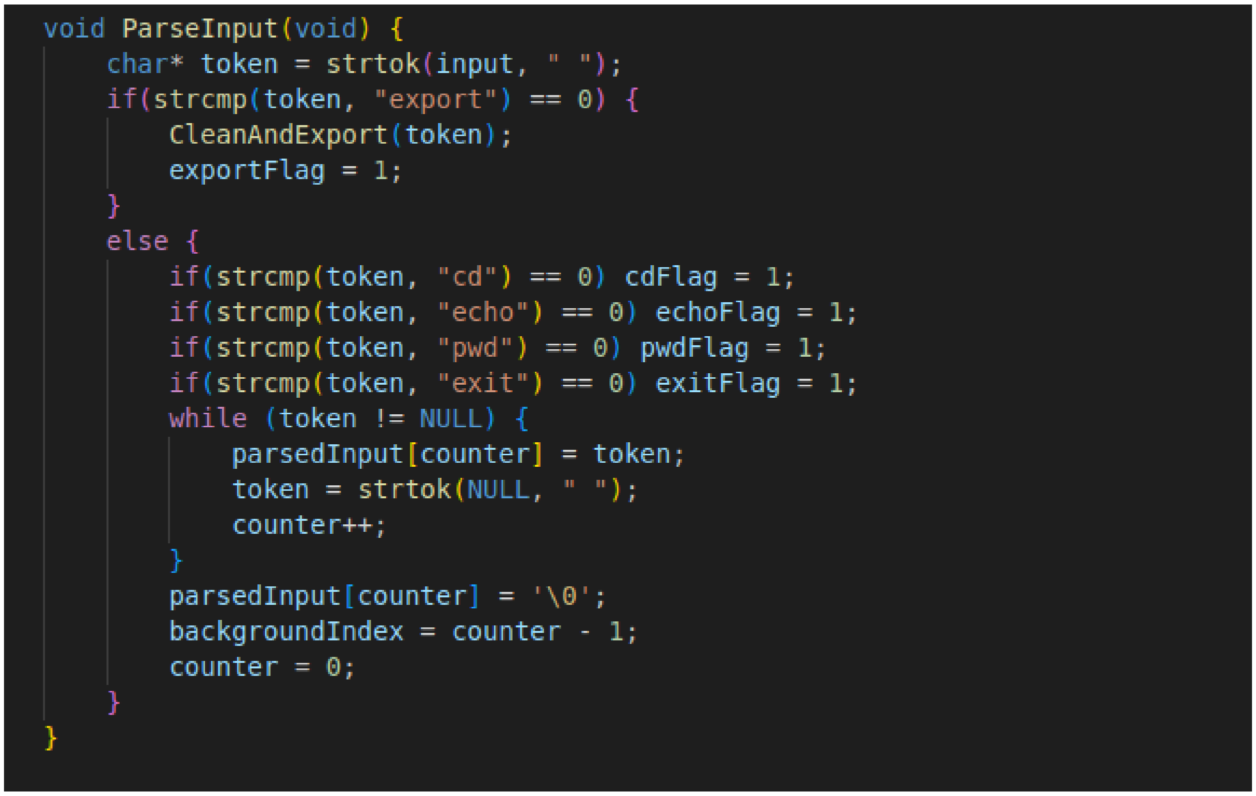
2. Major Functions

1. Get input : to get the command from the user



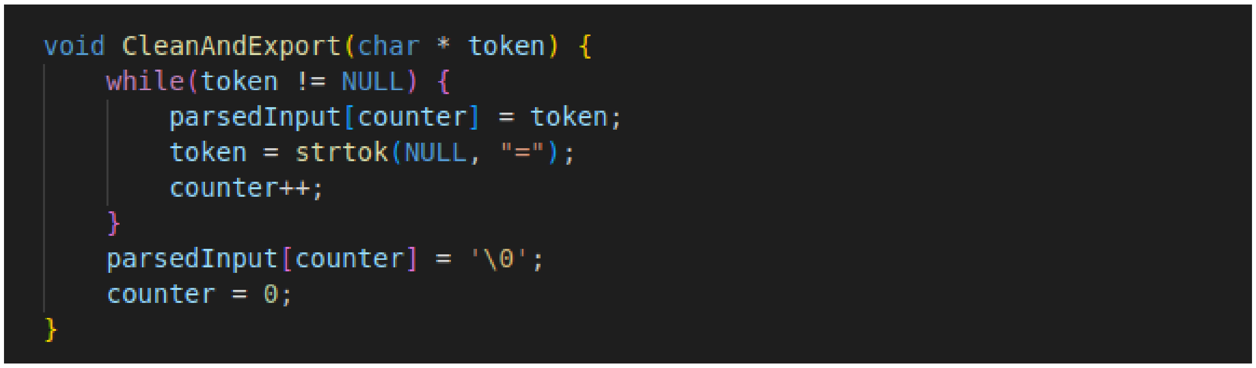
```c
void GetUserInput(void) {
    char cwd[100];
    printf("%s shell >> ",getcwd(cwd, 100));
    scanf("%[^\n]%*c", input);
}
```

2. Pares Input : to pares the input and put it in a global array of strings



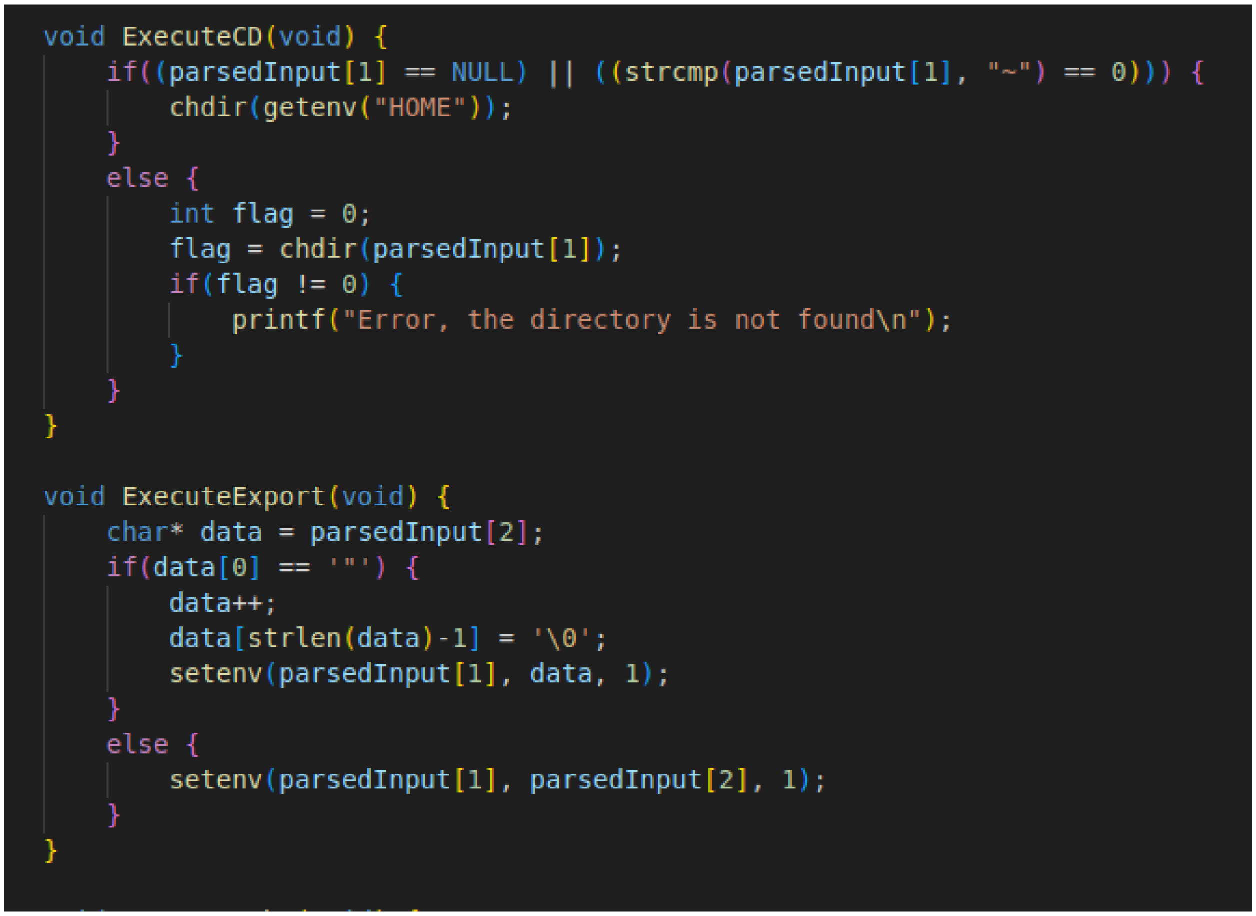
```c
void ParseInput(void) {
    char* token = strtok(input, " ");
    if(strcmp(token, "export") == 0) {
        CleanAndExport(token);
        exportFlag = 1;
    }
    else {
        if(strcmp(token, "cd") == 0) cdFlag = 1;
        if(strcmp(token, "echo") == 0) echoFlag = 1;
        if(strcmp(token, "pwd") == 0) pwdFlag = 1;
        if(strcmp(token, "exit") == 0) exitFlag = 1;
        while (token != NULL) {
            parsedInput[counter] = token;
            token = strtok(NULL, " ");
            counter++;
        }
        parsedInput[counter] = '\0';
        backgroundIndex = counter - 1;
        counter = 0;
    }
}
```

3-Clean Export : to parse the input in case command



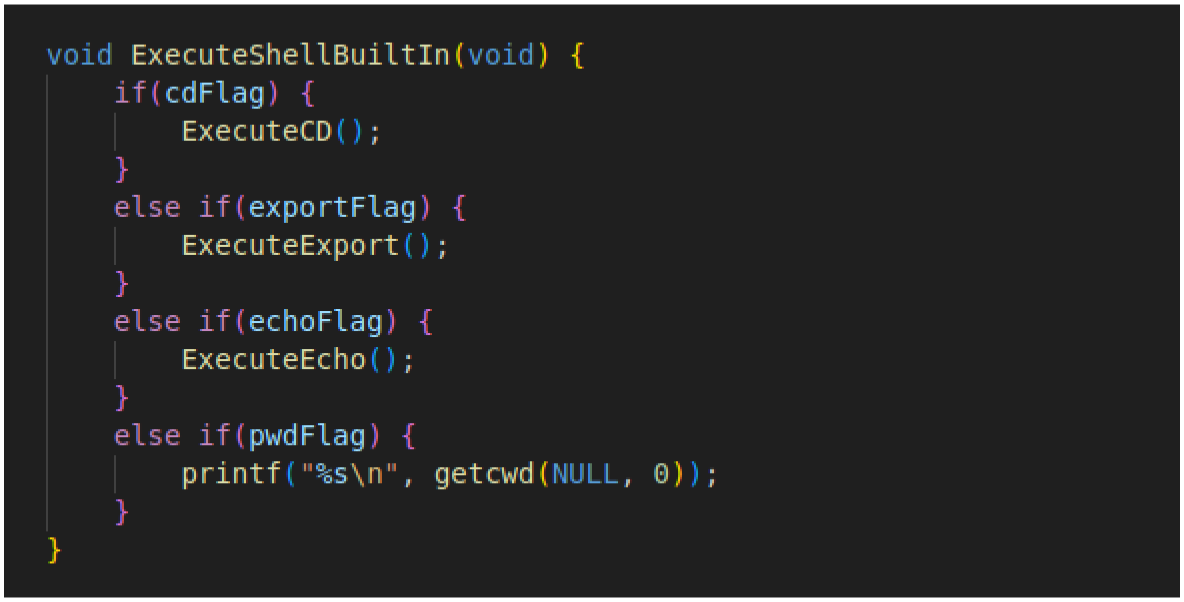
```c
void CleanAndExport(char * token) {
    while(token != NULL) {
        parsedInput[counter] = token;
        token = strtok(NULL, "=");
        counter++;
    }
    parsedInput[counter] = '\0';
    counter = 0;
}
```

4-Execute CD function & Execute Export function



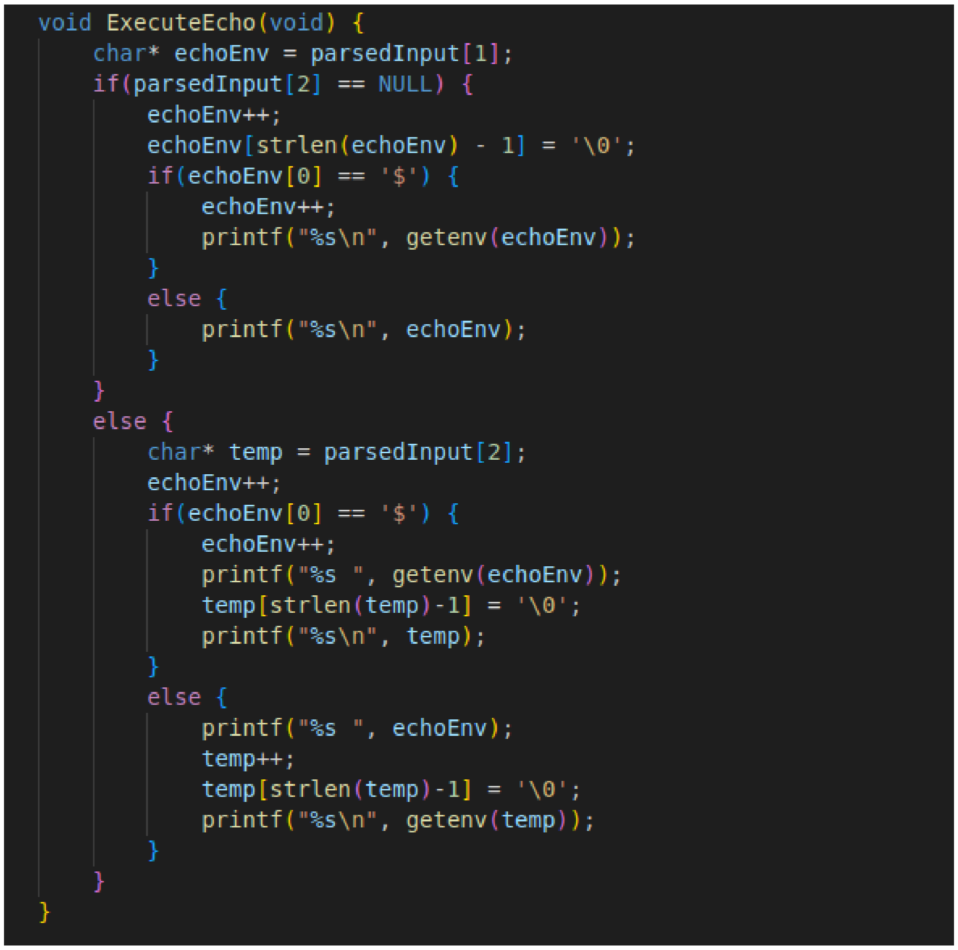
```c
void ExecuteCD(void) {
    if((parsedInput[1] == NULL) || ((strcmp(parsedInput[1], "~") == 0))) {
        chdir(getenv("HOME"));
    }
    else {
        int flag = 0;
        flag = chdir(parsedInput[1]);
        if(flag != 0) {
            printf("Error, the directory is not found\n");
        }
    }
}

void ExecuteExport(void) {
    char* data = parsedInput[2];
    if(data[0] == '"') {
        data++;
        data[strlen(data)-1] = '\0';
        setenv(parsedInput[1], data, 1);
    }
    else {
        setenv(parsedInput[1], parsedInput[2], 1);
    }
}
```

5-Execute Shell built in : (export,echo,pwd and cd)



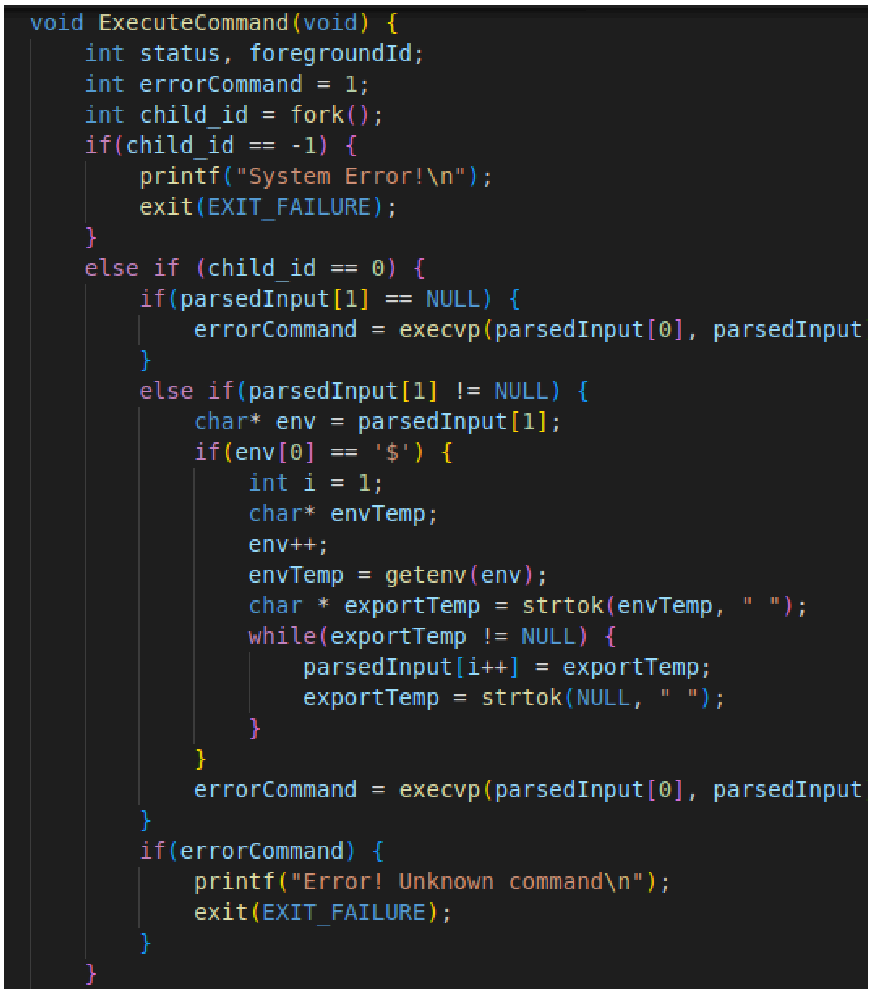
```c
void ExecuteShellBuiltIn(void) {
    if(cdFlag) {
        ExecuteCD();
    }
    else if(exportFlag) {
        ExecuteExport();
    }
    else if(echoFlag) {
        ExecuteEcho();
    }
    else if(pwdFlag) {
        printf("%s\n", getcwd(NULL, 0));
    }
}
```

6-Execute Echo function :

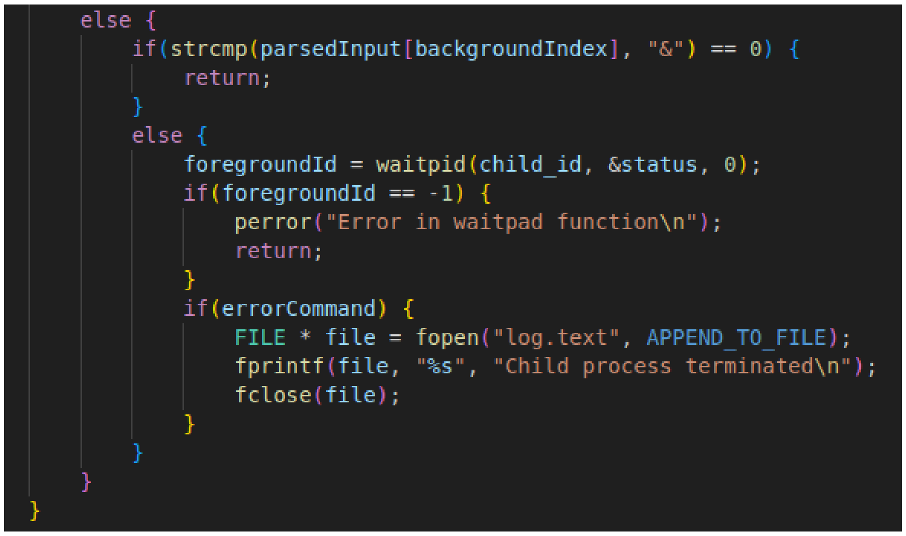


```c
void ExecuteEcho(void) {
    char* echoEnv = parsedInput[1];
    if(parsedInput[2] == NULL) {
        echoEnv++;
        echoEnv[strlen(echoEnv) - 1] = '\0';
        if(echoEnv[0] == '$') {
            echoEnv++;
            printf("%s\n", getenv(echoEnv));
        }
        else {
            printf("%s\n", echoEnv);
        }
    }
    else {
        char* temp = parsedInput[2];
        echoEnv++;
        if(echoEnv[0] == '$') {
            echoEnv++;
            printf("%s ", getenv(echoEnv));
            temp[strlen(temp)-1] = '\0';
            printf("%s\n", temp);
        }
        else {
            printf("%s ", echoEnv);
            temp++;
            temp[strlen(temp)-1] = '\0';
            printf("%s\n", getenv(temp));
        }
    }
}
```

7-Execute command :

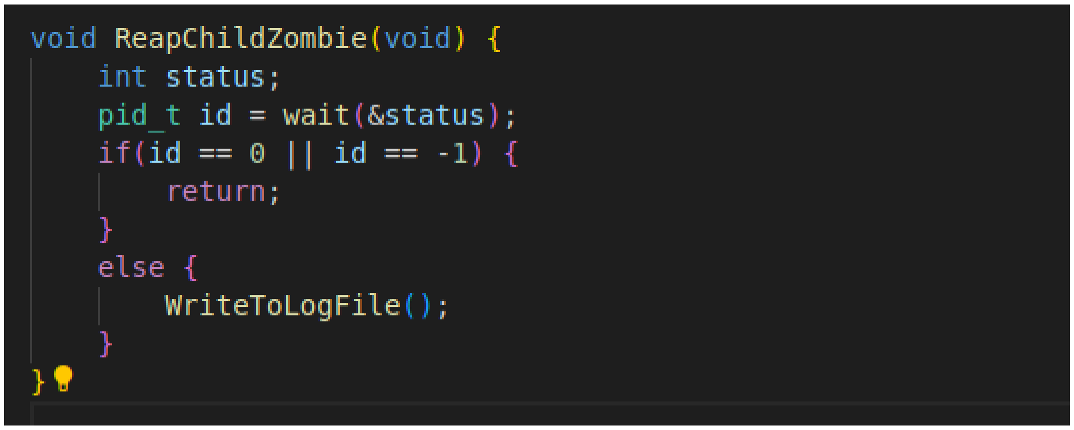


```
void ExecuteCommand(void) {
    int status, foregroundId;
    int errorCommand = 1;
    int child_id = fork();
    if(child_id == -1) {
        printf("System Error!\n");
        exit(EXIT_FAILURE);
    }
    else if (child_id == 0) {
        if(parsedInput[1] == NULL) {
            errorCommand = execvp(parsedInput[0], parsedInput
        }
        else if(parsedInput[1] != NULL) {
            char* env = parsedInput[1];
            if(env[0] == '$') {
                int i = 1;
                char* envTemp;
                env++;
                envTemp = getenv(env);
                char * exportTemp = strtok(envTemp, " ");
                while(exportTemp != NULL) {
                    parsedInput[i++] = exportTemp;
                    exportTemp = strtok(NULL, " ");
                }
            }
            errorCommand = execvp(parsedInput[0], parsedInput
        }
        if(errorCommand) {
            printf("Error! Unknown command\n");
            exit(EXIT_FAILURE);
        }
    }
```



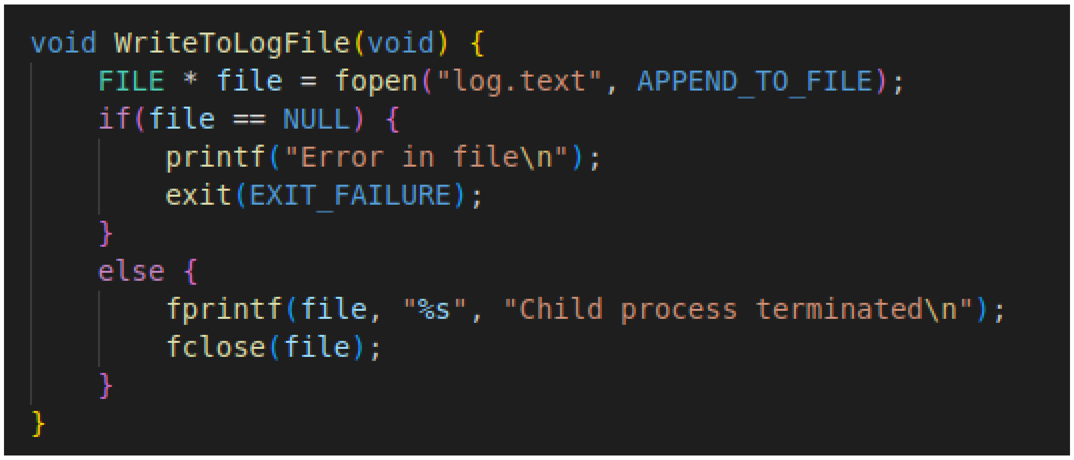
```
    else {
        if(strcmp(parsedInput[backgroundIndex], "&") == 0) {
            return;
        }
        else {
            foregroundId = waitpid(child_id, &status, 0);
            if(foregroundId == -1) {
                perror("Error in waitpad function\n");
                return;
            }
            if(errorCommand) {
                FILE * file = fopen("log.text", APPEND_TO_FILE);
                fprintf(file, "%s", "Child process terminated\n");
                fclose(file);
            }
        }
    }
}
```

8-Reap Child Zombie



```c
void ReapChildZombie(void) {
    int status;
    pid_t id = wait(&status);
    if(id == 0 || id == -1) {
        return;
    }
    else {
        WriteToLogFile();
    }
}
```

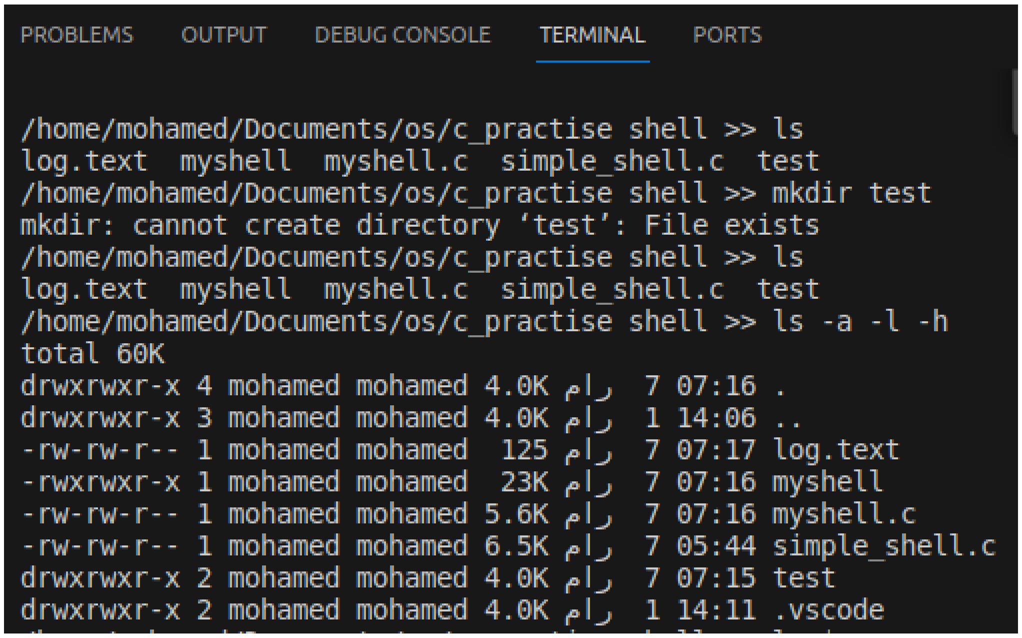
9-write in log file function:



```c
void WriteToLogFile(void) {
    FILE * file = fopen("log.text", APPEND_TO_FILE);
    if(file == NULL) {
        printf("Error in file\n");
        exit(EXIT_FAILURE);
    }
    else {
        fprintf(file, "%s", "Child process terminated\n");
        fclose(file);
    }
}
```

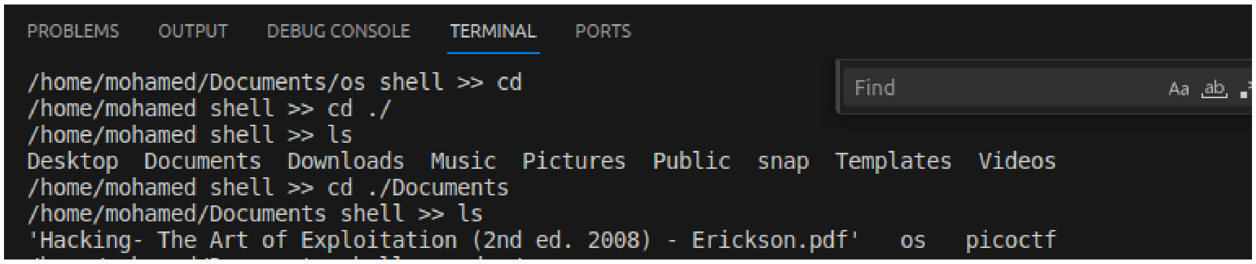
3.Sample runs

1-basic commands in shell



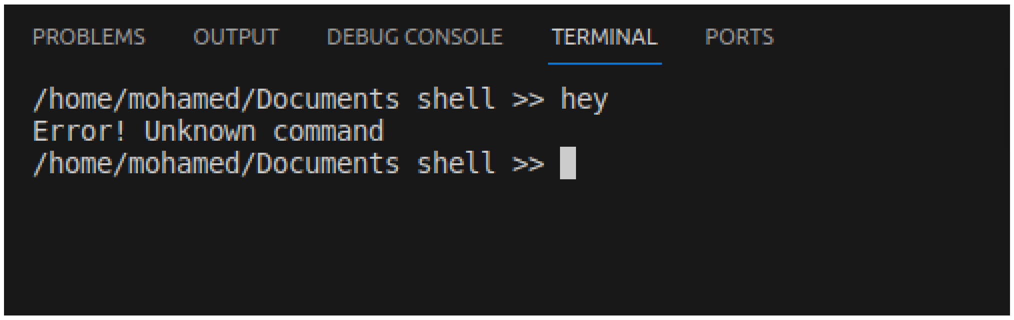
```
/home/mohamed/Documents/os/c_practise shell >> ls
log.text  myshell  myshell.c  simple_shell.c  test
/home/mohamed/Documents/os/c_practise shell >> mkdir test
mkdir: cannot create directory ‘test’: File exists
/home/mohamed/Documents/os/c_practise shell >> ls
log.text  myshell  myshell.c  simple_shell.c  test
/home/mohamed/Documents/os/c_practise shell >> ls -a -l -h
total 60K
drwxrwxr-x 4 mohamed mohamed 4.0K مار  7 07:16 .
drwxrwxr-x 3 mohamed mohamed 4.0K مار  1 14:06 ..
-rw-rw-r-- 1 mohamed mohamed  125 مار  7 07:17 log.text
-rwxrwxr-x 1 mohamed mohamed  23K مار  7 07:16 myshell
-rw-rw-r-- 1 mohamed mohamed 5.6K مار  7 07:16 myshell.c
-rw-rw-r-- 1 mohamed mohamed 6.5K مار  7 05:44 simple_shell.c
drwxrwxr-x 2 mohamed mohamed 4.0K مار  7 07:15 test
drwxrwxr-x 2 mohamed mohamed 4.0K مار  1 14:11 .vscode
```

2-built in commands (ex :cd)



```
/home/mohamed/Documents/os shell >> cd
/home/mohamed shell >> cd ./
/home/mohamed shell >> ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
/home/mohamed shell >> cd ./Documents
/home/mohamed/Documents shell >> ls
'Hacking- The Art of Exploitation (2nd ed. 2008) - Erickson.pdf'   os   picoctf
```

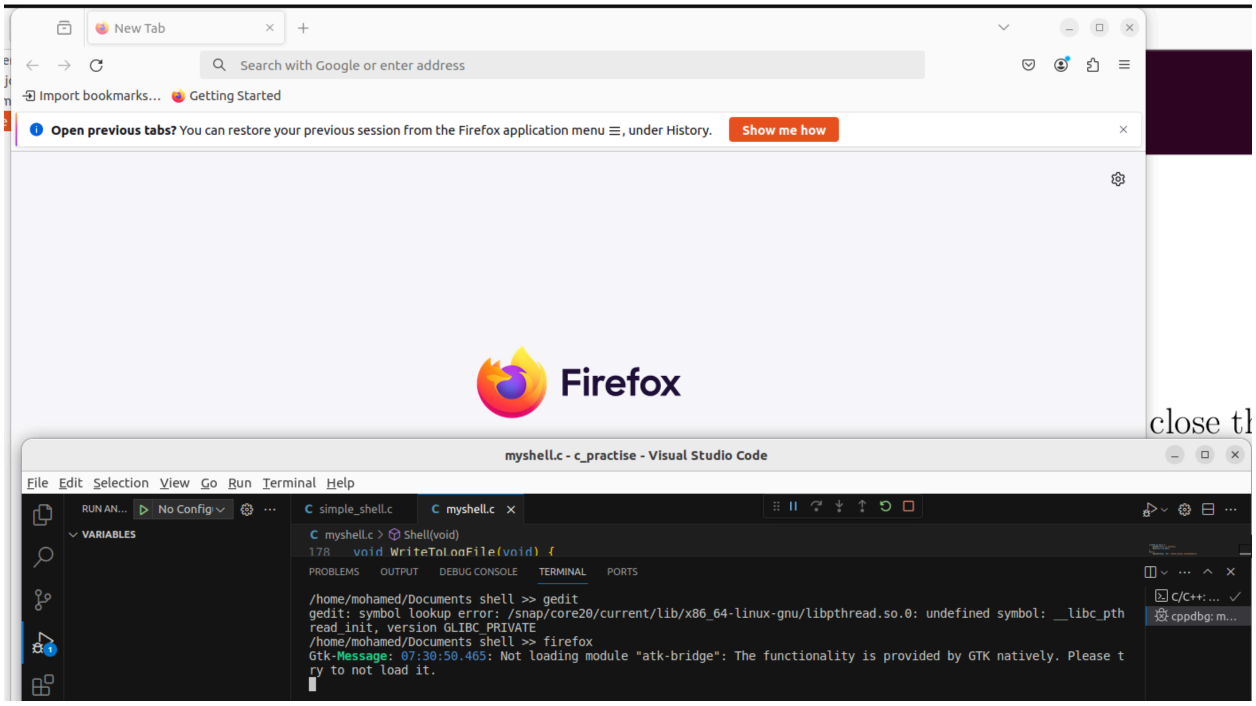
3-error



```
/home/mohamed/Documents shell >> hey
Error! Unknown command
/home/mohamed/Documents shell >> 
```

4-The process hierarchy

foreground process : the process doesn’t terminate until we close the firefox because it’s a foreground process

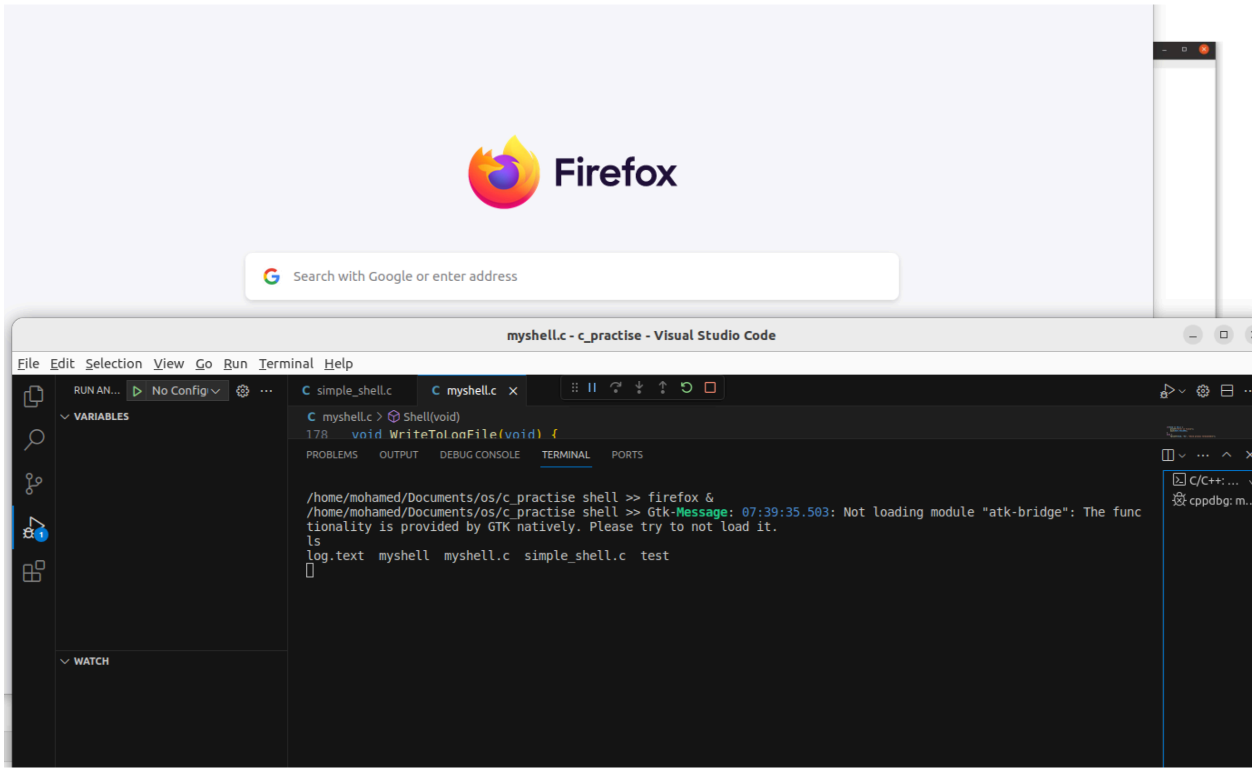


Open previous tabs? You can restore your previous session from the Firefox application menu ☰, under History.

```
void WriteToLogFile(void) {
```

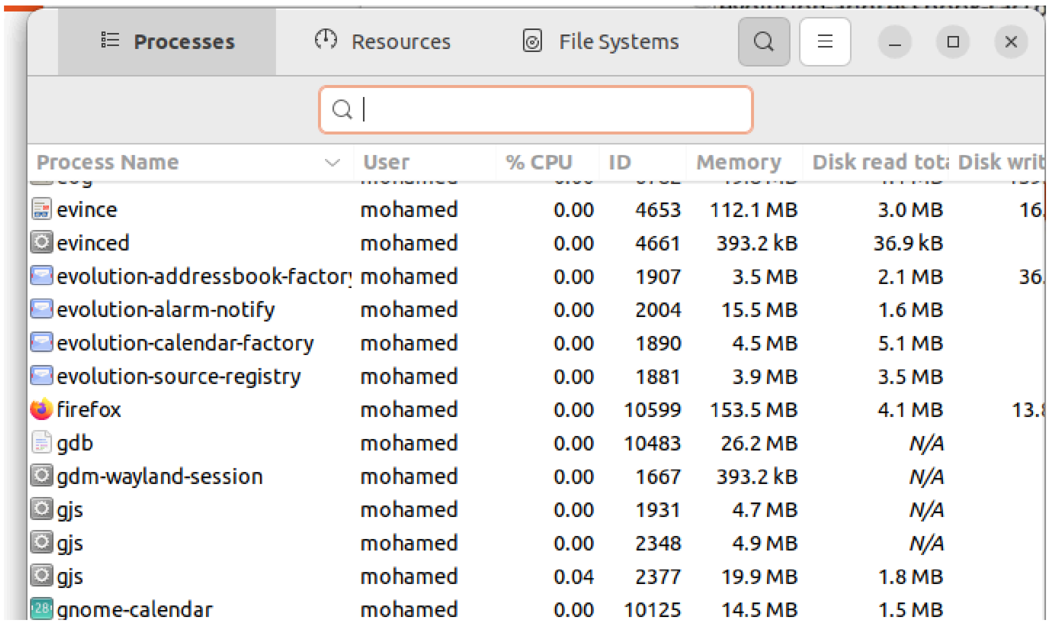
```
/home/mohamed/Documents shell >> gedit
gedit: symbol lookup error: /snap/core20/current/lib/x86_64-linux-gnu/libpthread.so.0: undefined symbol: __libc_pth
read_init, version GLIBC_PRIVATE
/home/mohamed/Documents shell >> firefox
Gtk-Message: 07:30:50.465: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please t
ry to not load it.
```

background process : the process work in the background and the user can enter another command


Firefox
Search with Google or enter address
myshell.c - c_practise - Visual Studio Code
File Edit Selection View Go Run Terminal Help
RUN AN... No Config
VARIABLES
WATCH
simple_shell.c
myshell.c
myshell.c > Shell(void)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
C/C++: ...
cppdbg: m...

```
void WriteToLogFile(void) {
```

```
/home/mohamed/Documents/os/c_practise shell >> firefox &
/home/mohamed/Documents/os/c_practise shell >> Gtk-Message: 07:39:35.503: Not loading module "atk-bridge": The func
tionality is provided by GTK natively. Please try to not load it.
ls
log.text  myshell  myshell.c  simple_shell.c  test
```



Process Name	User	% CPU	ID	Memory	Disk read tot	Disk writ
evince	mohamed	0.00	4653	112.1 MB	3.0 MB	16
evinced	mohamed	0.00	4661	393.2 kB	36.9 kB	
evolution-addressbook-factory	mohamed	0.00	1907	3.5 MB	2.1 MB	36
evolution-alarm-notify	mohamed	0.00	2004	15.5 MB	1.6 MB	
evolution-calendar-factory	mohamed	0.00	1890	4.5 MB	5.1 MB	
evolution-source-registry	mohamed	0.00	1881	3.9 MB	3.5 MB	
firefox	mohamed	0.00	10599	153.5 MB	4.1 MB	13.8
gdb	mohamed	0.00	10483	26.2 MB	N/A	
gdm-wayland-session	mohamed	0.00	1667	393.2 kB	N/A	
gjs	mohamed	0.00	1931	4.7 MB	N/A	
gjs	mohamed	0.00	2348	4.9 MB	N/A	
gjs	mohamed	0.04	2377	19.9 MB	1.8 MB	
gnome-calendar	mohamed	0.00	10125	14.5 MB	1.5 MB	