



Operating Systems

Assignment 1

Name: Mohamed Nasser Mohamed Ibrahim

ID: 18011632

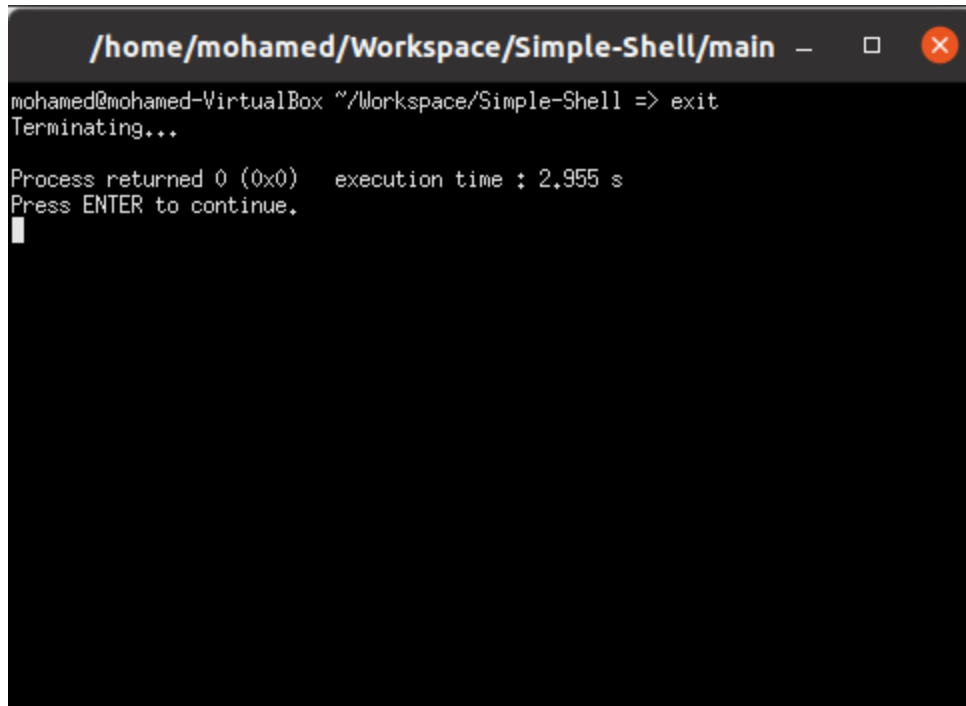
Name: Mohammed Adel Omar Bayoumy

ID: 18011514

Name: Mohammed Badr Saad Zaghloul

ID: 18015047

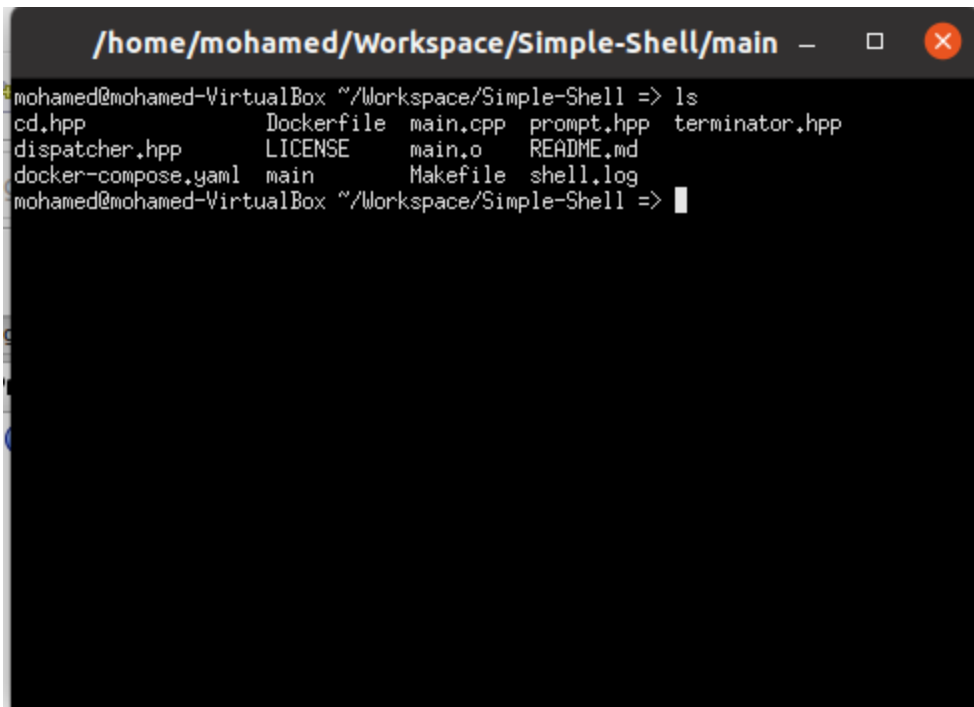
1- Exit command:



```
/home/mohamed/Workspace/Simple-Shell/main - □ ×
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => exit
Terminating...

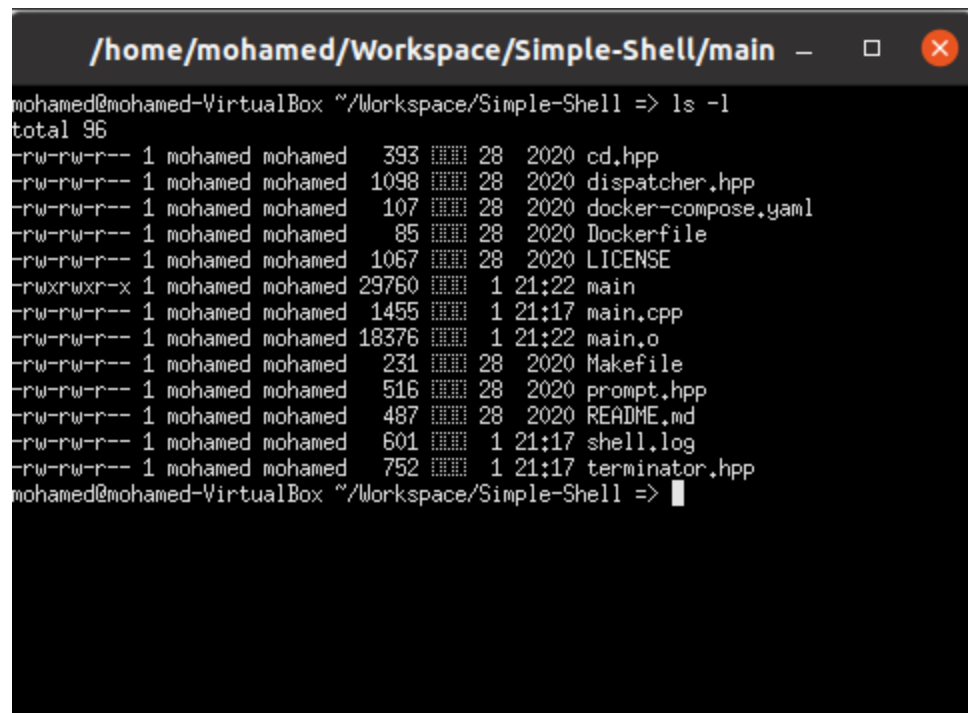
Process returned 0 (0x0)   execution time : 2.955 s
Press ENTER to continue.
█
```

2- A command with no arguments:



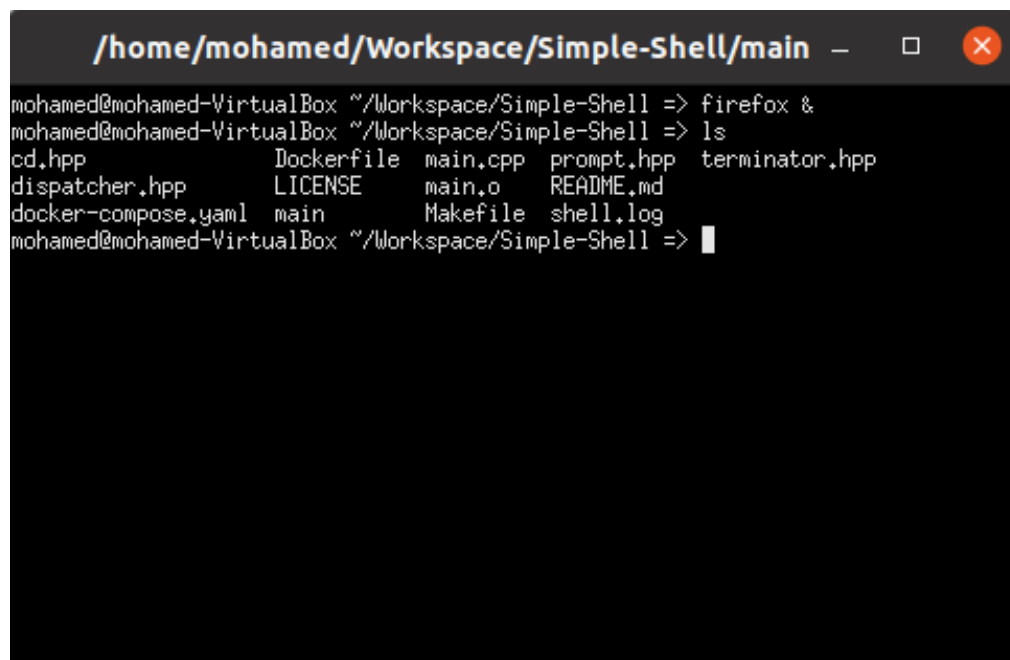
```
/home/mohamed/Workspace/Simple-Shell/main - □ ×
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => ls
cd.hpp          Dockerfile      main.cpp        prompt.hpp      terminator.hpp
dispatcher.hpp   LICENSE         main.o          README.md
docker-compose.yaml  main           Makefile        shell.log
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => █
```

3- A command with arguments:



```
/home/mohamed/Workspace/Simple-Shell/main - □ ×
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => ls -l
total 96
-rw-rw-r-- 1 mohamed mohamed 393 28 2020 cd.hpp
-rw-rw-r-- 1 mohamed mohamed 1098 28 2020 dispatcher.hpp
-rw-rw-r-- 1 mohamed mohamed 107 28 2020 docker-compose.yaml
-rw-rw-r-- 1 mohamed mohamed 85 28 2020 Dockerfile
-rw-rw-r-- 1 mohamed mohamed 1067 28 2020 LICENSE
-rwxrwxr-x 1 mohamed mohamed 29760 1 21:22 main
-rw-rw-r-- 1 mohamed mohamed 1455 1 21:17 main.cpp
-rw-rw-r-- 1 mohamed mohamed 18376 1 21:22 main.o
-rw-rw-r-- 1 mohamed mohamed 231 28 2020 Makefile
-rw-rw-r-- 1 mohamed mohamed 516 28 2020 prompt.hpp
-rw-rw-r-- 1 mohamed mohamed 487 28 2020 README.md
-rw-rw-r-- 1 mohamed mohamed 601 1 21:17 shell.log
-rw-rw-r-- 1 mohamed mohamed 752 1 21:17 terminator.hpp
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => █
```

4- A command in the background using &:



```
/home/mohamed/Workspace/Simple-Shell/main - □ ×
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => firefox &
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => ls
cd.hpp          Dockerfile  main.cpp   prompt.hpp  terminator.hpp
dispatcher.hpp  LICENSE    main.o     README.md
docker-compose.yaml  main      Makefile   shell.log
mohamed@mohamed-VirtualBox ~/Workspace/Simple-Shell => █
```

5- System Monitor:

Processes							
Process Name	User	% CPU	ID	Memory	Disk read tot	Disk v	
▼ gnome-shell	mohamed	4	1690	259.8 MiB	267.3 MiB		2
▼ codeblocks	mohamed	1	3635	87.9 MiB	120.0 KiB		
▼ xterm	mohamed	0	4524	4.0 MiB	N/A		
▼ cb_console_runne	mohamed	0	4525	180.0 KiB	N/A		
▼ sh	mohamed	0	4526	68.0 KiB	N/A		
▼ main	mohamed	0	4527	208.0 KiB	N/A		
▶ firefox	mohamed	1	4529	112.6 MiB	4.0 KiB		
gedit	mohamed	0	4788	18.4 MiB	N/A		
gnome-cal	mohamed	0	4778	12.2 MiB	N/A		
gnome-system-monitor	mohamed	5	4505	15.8 MiB	N/A		
▼ ibus-daemon	mohamed	0	1723	1.6 MiB	24.0 KiB		
ibus-dconf	mohamed	0	1727	752.0 KiB	12.0 KiB		
ibus-engine-simple	mohamed	0	2009	752.0 KiB	N/A		
ibus-extension-gtk3	mohamed	0	1728	13.4 MiB	744.0 KiB		
gnome-shell-calendar-serv	mohamed	0	1753	3.3 MiB	4.6 MiB		

End Process

6- Shell log:

Open		shell.log	
		~/Workspace/Simple-Shell	
1	Fri Apr 1 21:58:41 2022		
2	Child process was terminated		
3	Fri Apr 1 21:58:45 2022		
4	Child process was terminated		
5	Fri Apr 1 21:58:50 2022		
6	Child process was terminated		

Source Code:

Main.cpp:

```
#include <iostream>
#include <sstream>
#include "dispatcher.hpp"
#include "prompt.hpp"
#define bufferLength 99
using namespace std;

int main(int argc, char *argv[]) {
    if (argc > 1) {
        for (int i = 0; i < argc; i++) {
            argv[i] = argv[i + 1];
        }
        dispatcher(argc, argv);
        wait(NULL);
    }
    char username[bufferLength];
    char hostname[bufferLength];
    char cwd[bufferLength];
    cuserid(username);
    gethostname(hostname, bufferLength);
    getcwd(cwd, bufferLength);
    string userInput;
    string token;
    string argsVector[bufferLength];
    char *args[bufferLength];
    int arraySize;
```

```
while (1) {
    // clearing
    userInput.clear();
    token.clear();
    arraySize = 0;
    for (int i = 0; i < bufferLength; i++) {
        argsVector[i].clear();
        args[i] = NULL;
    }
    // prompting
    do {
        userInput = prompt(username, hostname, cwd);
    } while (userInput.empty() || isspace(userInput.at(0)));
    // string splitting
    stringstream stream(userInput);
    while (stream >> token) {
        argsVector[arraySize] = token;
        args[arraySize] = (char *)argsVector[arraySize].c_str();
        arraySize++;
    }
    // actual processing
    if (string(args[0]) == "exit" || string(args[0]) == "quit") {
        terminator(SIGQUIT);
    } else {
        dispatcher(arraySize + 1, args, cwd);
        wait(NULL);
    }
}
```

Prompt.hpp:

```
#ifndef PROMPT
#define PROMPT
#include <iostream>
using namespace std;

string prompt(char *username, char *hostname, char *cwd) {
    string userInput;
    string cwdString = string(cwd);
    string replaceString = "/home/" + string(username);

    if (cwdString.find(replaceString) < cwdString.length())
        cwdString.replace(0, replaceString.length(), "~");

    cout << username << "@" << hostname << " " << cwdString << " => ";
    getline(cin, userInput);
    if (cin.eof())
        return "logout";
    return userInput;
}
#endif
```

CD.hpp:

```
#ifndef CD
#define CD
#include <string.h>
#include <unistd.h>

#include <iostream>
#ifndef bufferLength
#define bufferLength 99
#endif

using namespace std;

void cd(int argc, char *args[], char *cwd) {
    if (argc == 2) {
        chdir(getenv("HOME"));
    } else {
        if (string(args[1]) == "~")
            chdir(getenv("HOME"));
        else
            chdir(args[1]);
    }
    getcwd(cwd, bufferLength);
}

#endif
```


Dispatcher.hpp:

```
#ifndef DISPATCHER
#define DISPATCHER
#include <string.h>
#include <unistd.h>
#include <iostream>

#include "cd.hpp"
#include "terminator.hpp"

typedef pid_t pid;
using namespace std;

void dispatcher(int argc, char* argv[], char* cwd = (char*)"") {
    char* args[argc];
    for (int i = 0; i <= argc; i++) {
        if (i == argc)
            args[i] = NULL;
        else
            args[i] = (char*)argv[i];
    }
    if (argc > 1) {
        pid processID = fork();

        if (processID < 0) {
            perror("Fork failed!\n");
            exit(1);
        }
    }
}
```

```
} else if (processID == 0) {
    if (string(args[argc - 2]) == "&" || string(args[0]) == "cd") {
        exit(0);
    } else {
        if (string(args[0]) == "log") {
            const char* log[] = {"vi", "shell.log", NULL};
            execvp(log[0], (char**)log);
        } else
            execvp(args[0], args);
    }
} else {
    if (string(args[0]) == "cd") {
        cd(argc, args, cwd);
    } else if (string(args[argc - 2]) == "&") {
        args[argc - 2] = NULL;
        dispatcher(argc - 1, args);
        signal(SIGCHLD, terminator);
    }
}
}
}
#endif
```

Terminator.hpp:

```
#ifndef TERMINATOR
#define TERMINATOR


#include <sys/wait.h>
#include <time.h>
#include <unistd.h>

#include <iostream>

using namespace std;
typedef pid_t pid;

void terminator(pid sigPid) {
    int status;
    pid processPid;
    time_t localTime = time(NULL);
    FILE *logFile;

    switch (sigPid) {
        case SIGCHLD:
            logFile = fopen("./shell.log", "a+");
            while ((processPid = waitpid(-1, &status, WNOHANG)) > 0) {
                fprintf(logFile, "%s\tChild process was terminated\n", asctime(localtime(&localTime)));
            }
            fclose(logFile);
            break;
    }
}
```



```
case SIGQUIT:
    cout << "Terminating..." << endl;
    exit(0);
    break;

default:
    cout << sigPid << endl;
    break;
}
}
#endif
```