The Islamic University of Gaza
Faculty of Engineering
Department of Computer Engineering

ECOM 4001 -Operating Systems LAB

Developing A Shell using c language

Prepared by:

No.	Name	ID
1	Mahmoud Hesham Almodalal	120211126
2	Mahmoud Farag Tayeh	120210653
3	Hazem Mohammed Oukal	120212771

Submission date:22/6/2025

❖ *Objectives:*

1. Understand how the shell does work
2. To understand the process fork (child, perant) exec
3. To understand the command in Lenix.

❖ *Introduction*

What is Unix?

"Unix is a portable, multitasking, multiuser, time-sharing operating system(OS) originally developed in 1969 by a group of employees at AT&T."(1)

UNIX was first programmed in assembly language but was reprogrammed in C in 1973. Unix has been ported to more machine families than any other operating system. As a result, it has come to be identified with the concept of open systems.

Unix operating systems are widely used in PCs, servers and mobile devices. Unix had a significant impact on other operating systems, such as:

1- Hierarchical file systems.
2- Unix shell inspired many of the command line interpreters that Followed.
3- C programming language became more pervasive.

What is a Unix Shell?

"A shell is an interface that allows you to interact with the kernel of an

Operating system."(2)

Shell is a **UNIX** term, for the interactive user interface with an Operating system. In some systems, the shell is called a command interpreter. The shell is the layer of programming that understands and executes the Commands a user enters.

As the outer layer of an operating system, a shell can be contrasted with the Kernel, the operating system's core of services.

❖ PROJECT DESCRIPTION:

The *Shell* or *Command Line Interpreter* is the fundamental User interface to an operating system. Your first project is to write a simple shell–`myshell`– that has the following properties:

1. The shell must support the following internal commands:

   i. `cd <directory>`– Change the current default directory to `<directory>`. If the `<directory>` argument is not present, report the current directory. If the directory does not exist, an appropriate error should be reported. This command should also change the **`PWD`** environment variable.

   ii. `clr`– Clear the screen.

   iii. `dir <directory>`– List the contents of directory `<directory>`.

   iv. `environ`– List all the environment strings.

   v. `echo <comment>`– Display `<comment>` on the display followed by a new line (multiple spaces/tabs may be reduced to a single space).

   vi. `help`– Display the user manual using the more filter.

   vii. `pause`– Pause operation of the shell until "Enter" is pressed.

   viii. `quit`– Quit the shell.

   ix. The shell environment should contain `shell=<pathname>/myshell` where `<pathname>/myshell` is the full path for the shell executable (not a hardwired path back to your directory, but the one from which it was executed).

2. All other command line input is interpreted as program invocation, which should be done by the shell `fork`ing and `exec`ing the programs as its own child processes. The programs should be executed with an environment that contains the entry: `parent=<pathname>/myshell` where `<pathname>/myshell` is as described in **1.ix** above.

3. The shell must be able to take its command line input from a file. That is, if the shell is invoked with a command line argument:

   ```
   myshell batchfile
   ```

   then `batchfile` is assumed to contain a set of command lines for the shell to process. When the end-of-file is reached, the shell should exit. Obviously, if the shell is invoked without a command line argument, it solicits input from the user via a prompt on the display.

4. The shell must support I/O redirection on either or both *stdin* and/or *stdout*. That is, the command line

   ```
   programname arg1 arg2 < inputfile > outputfile
   ```

   will execute the program `programname` with arguments `arg1` and `arg2`, the *stdin FILE stream* replaced by `inputfile` and the *stdout FILE stream* replaced by `outputfile`.

stdout redirection should also be possible for the internal commands `dir`, `environ`, `echo`, and `help`.

With output redirection, if the redirection character is > then the `outputfile` is created if it does not exist, and truncated if it does. If the redirection token is >> then `outputfile` is created if it does not exist, and appended to if it does.

5. The shell must support background execution of programs. An ampersand (**&**) at the end of the command line indicates that the shell should return to the command line prompt immediately after launching that program.

6. The command line prompt must contain the pathname of the current directory.

Note: You can assume all command line arguments (including the redirection symbols, <, > & >> and the background execution symbol, **&**) will be delimited from other command line arguments by white space—one or more spaces and/or tabs (see the command line in 4. above).

❖ *PROJECT REQUIREMENTS*

1. Design a simple command line shell that satisfies the above criteria and implement it on the specified UNIX platform.
2. Write a simple manual describing how to use the shell. The manual should contain enough detail for a beginner to UNIX to use it. For example, you should explain the concepts of I/O redirection, the program environment, and background program execution. The manual **MUST** be named `readme` and must be a simple text document capable of being read by a standard Text Editor. For an example of the sort of depth and type of description required, you should have a look at the online manuals for `csh` and `tcsh` (`man csh, man tcsh`). These shells obviously have much more functionality than yours and thus, your manuals don't have to be quite so large. You should **NOT** include building instructions, included file lists, or source code—we can find that out from the other files you submit. This should be an Operator's manual not a Developer's manual.
3. The source code MUST be extensively commented and appropriately structured to allow your peers to understand and easily maintain the code. Properly commented and laid out code is much easier to interpret, and it is in your interests to ensure the person marking your project is able to understand your coding without having to perform mental gymnastics!
4. Details of submission procedures will be supplied well before the deadline.
5. The submission should contain only source code file(s), include file(s), a `makefile` (all lowercase please), and the `readme` file (all lowercase, please). No executable program should be included. The person marking your project will be automatically rebuilding your shell program from the source code provided. If the submitted code does not compile, it cannot be marked!
6. The makefile (all lowercase, please) **MUST** generate the binary file myshell (all lowercase please). A sample makefile would be

```
# Joe Citizen, s1234567 - Operating Systems Project 1 # CompLab1/01
tutor: Fred Bloggs
myshell: myshell.c utility.c myshell.h
	gcc -Wall myshell.c utility.c -o myshell
```

The program myshell is then generated by just typing make at the command line prompt. *Note:* The fourth line in the above makefile **MUST** begin with a tab.

7. In the instance shown above, the files in the submitted directory would be:

```
Makefile
myshell.c
utility.c
myshell.h
readme
```

❖ *SUBMISSION*

A makefile is required. All files in your submission will be copied to the same directory, therefore, do not include any paths in your makefile. The makefile should include all dependencies that build your program. If a library is included, your makefile should also build the library. Do not hand in any binary or object code files. All that is required is your source code, a makefile, and a readme file. Test your project by copying the source code only into an empty directory then compile it by entering the command make. We shall be using a shell script that copies your files to a test directory, deletes any preexisting myshell, *.a, and/or *.o files, performs a make, copies a set of test files to the test directory, and then exercises your shell with a standard set of test scripts through stdin and command line arguments. If this sequence fails due to wrong names, wrong case for names, wrong version of source code that fails to compile, nonexistence of files, and so on, then the marking sequence will also stop. In this instance, the only marks that can be awarded will be for the tests completed at that point, and the source code and manual.

❖ *PROJECT C SOURCE CODE*

myshell.h: *header file for defined and include library's and functions*

```c
#ifndef MYSHELL_H
#define MYSHELL_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>

#define MAX_COMMAND_LENGTH 1024
#define MAX_PATH 1024

// Color macros
#define COLOR_RESET   "\033[0m"
#define COLOR_RED     "\033[1;31m"
#define COLOR_GREEN   "\033[1;32m"
#define COLOR_BLUE    "\033[1;34m"
#define COLOR_CYAN    "\033[1;36m"

// Prototypes
char *read_command();
char **parse_command(char *command_line);
void execute_command(char **args);
void redirect_io(char **args);
void restore_io();
void internal_cd(char **args);
void internal_echo(char **args);
void internal_environ();
void internal_help();
void internal_pause();
void internal_quit();
void internal_dir(char **args);
void internal_clr();
void show_pwd();
void free_args(char **args);
```

```
#endif
```

myshell.c: *contain the main function to Initializes the shell, sets the shell path environment variable, checks for batch mode, loops over command input, and dispatches commands to execution.*

```
#include "myshell.h"

int main(int argc, char *argv[]) {
    char *command = NULL;
    char **args = NULL;
    FILE *batch_file = NULL;
    size_t len = 0;

    if (argc == 2) {
        // Open batch file
        batch_file = fopen(argv[1], "r");
        if (!batch_file) {
            perror("myshell: cannot open batch file");
            return 1;
        }

        while (getline(&command, &len, batch_file) != -1) {
            command[strcspn(command, "\n")] = 0;  // Strip newline
            printf(COLOR_GREEN "Batch> %s\n" COLOR_RESET, command);
            args = parse_command(command);
            execute_command(args);
            free_args(args);
        }

        fclose(batch_file);
        free(command);
    } else {
        // Interactive shell
        while ((command = read_command()) != NULL) {
            args = parse_command(command);
            execute_command(args);
            free_args(args);
            free(command);
        }
    }

    return 0;
}
```

Utility.c: contain function for execute command external and internal file

```
#include "myshell.h"

extern char **environ;

int original_stdin, original_stdout;
FILE *input_file = NULL, *output_file = NULL;

void redirect_io(char **args) {
    original_stdin = dup(STDIN_FILENO);
    original_stdout = dup(STDOUT_FILENO);
    input_file = output_file = NULL;

    for (int i = 0; args[i]; ++i) {
        if (strcmp(args[i], "<") == 0) {
            input_file = fopen(args[i + 1], "r");
            if (!input_file) {
                fprintf(stderr, COLOR_RED "myshell: input error\n" COLOR_RESET);
                return;
            }
            dup2(fileno(input_file), STDIN_FILENO);
            for (int j = i; args[j]; ++j)
                args[j] = args[j + 2];
            i--;
        } else if (strcmp(args[i], ">") == 0 || strcmp(args[i], ">>") == 0) {
            const char *mode = (strcmp(args[i], ">>") == 0) ? "a" : "w";
            output_file = fopen(args[i + 1], mode);
            if (!output_file) {
                fprintf(stderr, COLOR_RED "myshell: output error\n" COLOR_RESET);
                return;
            }
            dup2(fileno(output_file), STDOUT_FILENO);
            for (int j = i; args[j]; ++j)
                args[j] = args[j + 2];
            i--;
        }
    }
}

void restore_io() {
    dup2(original_stdin, STDIN_FILENO);
    dup2(original_stdout, STDOUT_FILENO);
    close(original_stdin);
    close(original_stdout);
    if (input_file) fclose(input_file);
    if (output_file) fclose(output_file);
}

char *read_command() {
    char *line = NULL;
    size_t len = 0;
    printf(COLOR_BLUE "%s>" COLOR_RESET, getenv("PWD") );
    if (getline(&line, &len, stdin) == -1) {
        printf("\n");
        return NULL;
    }
    line[strcspn(line, "\n")] = 0;
    return line;
}

char **parse_command(char *line) {
    int bufsize = 64, pos = 0;
    char **tokens = malloc(bufsize * sizeof(char *));
    char *token = strtok(line, " \t\r\n");

    while (token) {
        tokens[pos++] = token;
        if (pos >= bufsize) {
            bufsize += 64;
            tokens = realloc(tokens, bufsize * sizeof(char *));
        }
        token = strtok(NULL, " \t\r\n");
    }
    tokens[pos] = NULL;
    return tokens;
}
void show_pwd () {
    fprintf(stdout, "PWD=%s\n", getenv("PWD"));
    return;
}

void execute_command(char **args) {
    if (!args[0]) return;

    int background = 0;
    for (int i = 0; args[i]; ++i) {
        if (strcmp(args[i], "&") == 0) {
            background = 1;
            args[i] = NULL;
            break;
        }
    }

    redirect_io(args);

    if (strcmp(args[0], "cd") == 0) {
        internal_cd(args);
    } else if (strcmp(args[0], "echo") == 0) {
        internal_echo(args);
    } else if (strcmp(args[0], "environ") == 0) {
        internal_environ();
    } else if (strcmp(args[0], "help") == 0) {
        internal_help();
    } else if (strcmp(args[0], "pause") == 0) {
        internal_pause();
    } else if (strcmp(args[0], "pwd") == 0) {
    show_pwd();
    }else if (strcmp(args[0], "quit") == 0) {
        internal_quit();
    } else if (strcmp(args[0], "clr") == 0) {
        internal_clr();
    } else if (strcmp(args[0], "dir") == 0) {
        internal_dir(args);
    } else {
        pid_t pid = fork();
        if (pid == 0) {
            setenv("PATH", "/usr/bin:/bin:/usr/local/bin", 1);
            if (execvp(args[0], args) == -1) {
                fprintf(stderr, COLOR_RED "myshell: command failed\n" COLOR_RESET);
                exit(1);
            }
        } else if (pid > 0) {
            if (!background)
                waitpid(pid, NULL, 0);
        } else {
            perror("myshell");
        }
    }

    restore_io();
}

void internal_cd(char **args) {
    char path[MAX_COMMAND_LENGTH];

    if (input_file == NULL) {
```

```c
        // Interactive mode
        if (args[1] == NULL) {
            printf("%s\n", getenv("PWD"));
        } else {
            if (chdir(args[1]) != 0)
                perror("cd");
        }
    } else {
        // Redirected input
        if (fgets(path, sizeof(path), stdin)) {
            path[strcspn(path, "\n")] = 0;
            if (strlen(path) == 0) {
                // Empty line in input file
                printf("%s\n", getenv("PWD"));
            } else {
                if (chdir(path) != 0)
                    perror("cd");
            }
        } else {
            // Nothing read (EOF) — empty file
            printf("%s\n", getenv("PWD"));
        }
    }

    // Always update PWD environment variable
    char cwd[MAX_PATH];
    if (getcwd(cwd, sizeof(cwd)))
        setenv("PWD", cwd, 1);
}

void internal_echo(char **args) {
    if (input_file == NULL) {
        for (int i = 1; args[i]; i++) {
            printf("%s", args[i]);
            if (args[i + 1]) printf(" ");
        }
        printf("\n");
    } else {
        char buffer[1024];
        while (fgets(buffer, sizeof(buffer), stdin)) {
            fputs(buffer, stdout);
        }
    }
}

void internal_environ() {
    for (char **env = environ; *env; ++env) {
        printf(COLOR_GREEN "%s\n" COLOR_RESET, *env);
    }
}

void internal_help() {
    FILE *file = fopen("README.md", "r");
    if (!file) {
        perror("help");
        return;
    }
    pid_t pid = fork();
    if (pid == 0) {
        dup2(fileno(file), STDIN_FILENO);
        execlp("more", "more", NULL);
        exit(0);
    } else {
        waitpid(pid, NULL, 0);
    }
    fclose(file);
}

void internal_pause() {
    printf(COLOR_CYAN "Press Enter to continue..." COLOR_RESET);
    while (getchar() != '\n');
}

void internal_quit() {
    printf(COLOR_GREEN "Goodbye!\n" COLOR_RESET);
    exit(0);
}

void internal_dir(char **args) {
    DIR *dir;
    const char *path = (args[1]) ? args[1] : ".";
    struct dirent *entry;

    if (!(dir = opendir(path))) {
        perror("dir");
        return;
    }

    while ((entry = readdir(dir)) != NULL) {
        printf(COLOR_CYAN "%s\n" COLOR_RESET, entry->d_name);
    }

    closedir(dir);
}
```

```
void internal_clr() {
    system("clear");
}

void free_args(char **args) {
    free(args);
}
```

❖ MyShell

Overview

MyShell is a basic command-line shell implemented in C. It supports execution of internal and external commands, input/output redirection, background processing, and batch mode via a script file.

Features

*- **Command prompt** showing current working directory*
*- **Internal commands** like `cd`, `clr`, `dir`, `environ`, `echo`, `help`, `pwd`,`pause`, `quit` **executable external** file can include these commands as well.*
*- **Redirection support: `>`, `>>`, `<`***
Myshell supports i/o-redirection on either or both stdin(standard input) and/or stdout(standard output). i.e. the command line:
e.g.   programname arg1 arg2 < inputfile > outputfile this will execute the program programname with arguments arg1 and arg2, the stdin FILE stream replaced by inputfile and the stdout FILE stream replaced by outputfile. The file can be given by the full path as: /home/username/test.txt

Input redirection → i.e. stdin redirection → is possible for the internal commands: cd, dir, echo.
Output redirection → i.e. stdout redirection → is possible for the internal commands: pwd, dir, environ, echo, help.

=> when the redirection token is < then the inputfile is opened if it exists, or a "Path Error" wil be reported in the screen.
e.g.   cd < test.txt
=> when the redirection token is > then the outputfile is created if it does not exist and truncated if it does.
e.g.   ls > test.txt
=> when the redirection token is >> then outputfile is created if it does not exist and appended to if it does.
e.g.   environ >> test.txt
Sometimes you can use more than one token to open several files as input or/and output.

e.g.   echo < m.txt > n.txt
dir > m.txt > n.txt
If the file cannot opened due to read permission or write permission, an "Open Error" wil be reported in the screen.

*- **Background execution using `&`***
Normally, when a command is execution, you have to wait for the completion and type another command. Especially, when the comand is executed from a batchfile, much time is wasted in waiting. Background execution allows you not to wait for the execution. An ampersand (&) at the end of the command line indicates that the shell should execute the command in background.
Myshell supports background execution of programs. An ampersand (&) at the end of the command line indicates that the shell should return to the command line prompt immediately after launching that program.

*- **Batch file support***
Myshell is able to take its command line input from a batchfile. if the shell is invoked with a command line argument:
e.g.   myshell test.bat
then the batchfile [test.bat] is assumed to contain a set of command lines for the shell to process.
When the end of file is reached, the shell should exit. Obviously, if the shell is invoked without a command line argument, It solicits input from the user via a prompt on the display.

Internal Command Descriptions

`cd [dir]`

Format: cd [directory]
=> change the current default directory to [directory].
e.g.   cd /home
This command also change the PWD environment variable (use the command "pwd" to see it).
If the [directory] argument is not present, report the current directory.
If the directory does not exist, a "Path Error" wil be reported in the screen.
Moreover, you can use a directory path writen in a file as:
e.g.   cd < test.txt
The file can be given by the full path as:
cd < /home/username/test.txt

`clr`

Format: clr
or   clear
-> clear the screen, no arguments is needed.

`dir [dir]`

Format: dir [directory]
-> list the contents of directory [directory]
e.g.   dir /home
This command is different from "cd", it change neither the current default directory nor the PWD environment variable(use the command "pwd" to see it).
As to the directory path, type "help path" to get more information.

If the [directory] argument is not present, list the contents of the current directory. If the directory does not exist, a "Path Error" wil be reported in the screen.

Moreover, you can use a directory path written in a file as:
e.g.   dir < a.txt
And you can list the contents into a file or more than one file as:
e.g.   dir > b.txt   or   dir > b.txt > c.txt
The token ">" can be replaced by ">>", type "help" to see the difference of ">" and ">>".
You can use both input redirection and output redirection as:
e.g.   dir < a.txt > b.txt > c.txt
The file can be given by the full path as:
dir < /home/username/a.txt

`environ`

Format: environ
-> list all the environment strings in screen or into one file or more than one file as:
e.g.   envieron   or   environ > b.txt   or   environ > b.txt > c.txt
The token ">" can be replaced by ">>", type "help" to see the difference of ">" and ">>".
The file can be given by the full path as:
/home/a.txt

`echo [args...]`

Format: echo [comment]
-> display [comment] on the display or output files followed by a new line. And multiple spaces/tabs will be reduced to a single space.
e.g.   echo hello   world
The words "hello world" will display in the screen.
[comment] can be multiple words either typed from keyboard or read from one or more input files.
e.g.   echo < a.txt   or   echo < a.txt < b.txt
And you can display or output the contents into one file or more than one file:
e.g.   echo hello world > c.txt or echo hello world > c.txt > d.txt
The token ">" can be replaced by ">>", type "help" to see the difference of ">" and ">>".
You can use both input redirection and output redirection:
e.g.   echo < a.txt < b.txt > c.txt > d.txt
The file can be given by the full path as:
echo < /home/a.txt > /home/b.txt

`help`

Format: help
-> display the user manual. Type "help [command]" to get the detail usage of a command as:
e.g.   help dir   or   ? dir
Type "help command" to see the internal commands.
And you can display or output the help information into one file or more than one file as:
e.g.   help dir > c.txt   or   help dir > c.txt > d.txt
The token ">" can be replaced by ">>", type "help" to see the difference of ">" and ">>".
The file can be given by the full path as:
help dir > /home/a.txt

`pwd`
Format: pwd
-> show the PWD environment variable. If you want to list all the environment strings, use command "environ", type "help environ" to see.
You can display or output the execution results into one file or more than one file as:
e.g.   pwd test.bat > c.txt   or   pwd test.bat > c.txt > d.txt
The token ">" can be replaced by ">>", type "help" to see the difference of ">" and ">>".
The file can be given by the full path as:
e.g.   pwd > /home/a.txt

`pause`
Format: pause
-> display "Press Enter to continue..." and pause operation of the shell until the 'Enter' key is pressed (ignore any intervening non-'Enter' input).

`quit`
Format: quit   or   exit
-> Exit myshell, no argument is needed.

## File: main()

Initializes the shell, sets the shell path environment variable, checks for batch mode, loops over command input, and dispatches commands to execution.

### *File: main()*
Initializes the shell, sets the shell path environment variable, checks for batch mode, loops over command input, and dispatches commands to execution.

Function: execute_command()
Handles the parsing and execution of both internal and external commands. Supports:
- Redirection (`>`, `>>`, `<`)
- Background execution (`&`)
- Setting the `parent` environment variable

Function: read_command()
Reads a full line of user input from `stdin`.

Function: parse_command()
Splits a raw command line into an array of tokens (i.e., arguments for exec).

Function: redirect_io()
Scans the `args[]` array for `<`, `>`, or `>>`, and performs I/O redirection.

Function: restore_io()
Restores original standard input/output after redirection.

Functoin: free_args()
Frees memory allocated by `parse_command()`.

Function: internal_cd()

Handles changing the directory using `chdir()` and updates `PWD` environment variable.

Function: internal_clr()

Clears the screen using escape codes or system("clear").

Function: internal_dir()

Opens the specified directory (or current one), and prints its contents using `readdir()`.

Function: internal_environ()

Iterates over `environ` to print all environment variables.

Function: internal_echo()

Prints all arguments passed to it starting from the second one.

Function: internal_help()

Uses `more` to display the `readme` file content. Forks a process to run `more` and redirects the help file as standard input.

Function: internal_pause()

Prompts the user to press Enter and waits for it using `getchar()`.

Function: internal_quit()

Simply calls `exit(EXIT_SUCCESS)` to terminate the shell.

Function: show_pwd()

Prints the `PWD` environment variable to stdout.

Dependencies

- ***`stdio.h`, `stdlib.h`, `string.h`, `unistd.h`***
- ***`fcntl.h`, `sys/wait.h`, `sys/stat.h`, `dirent.h`***

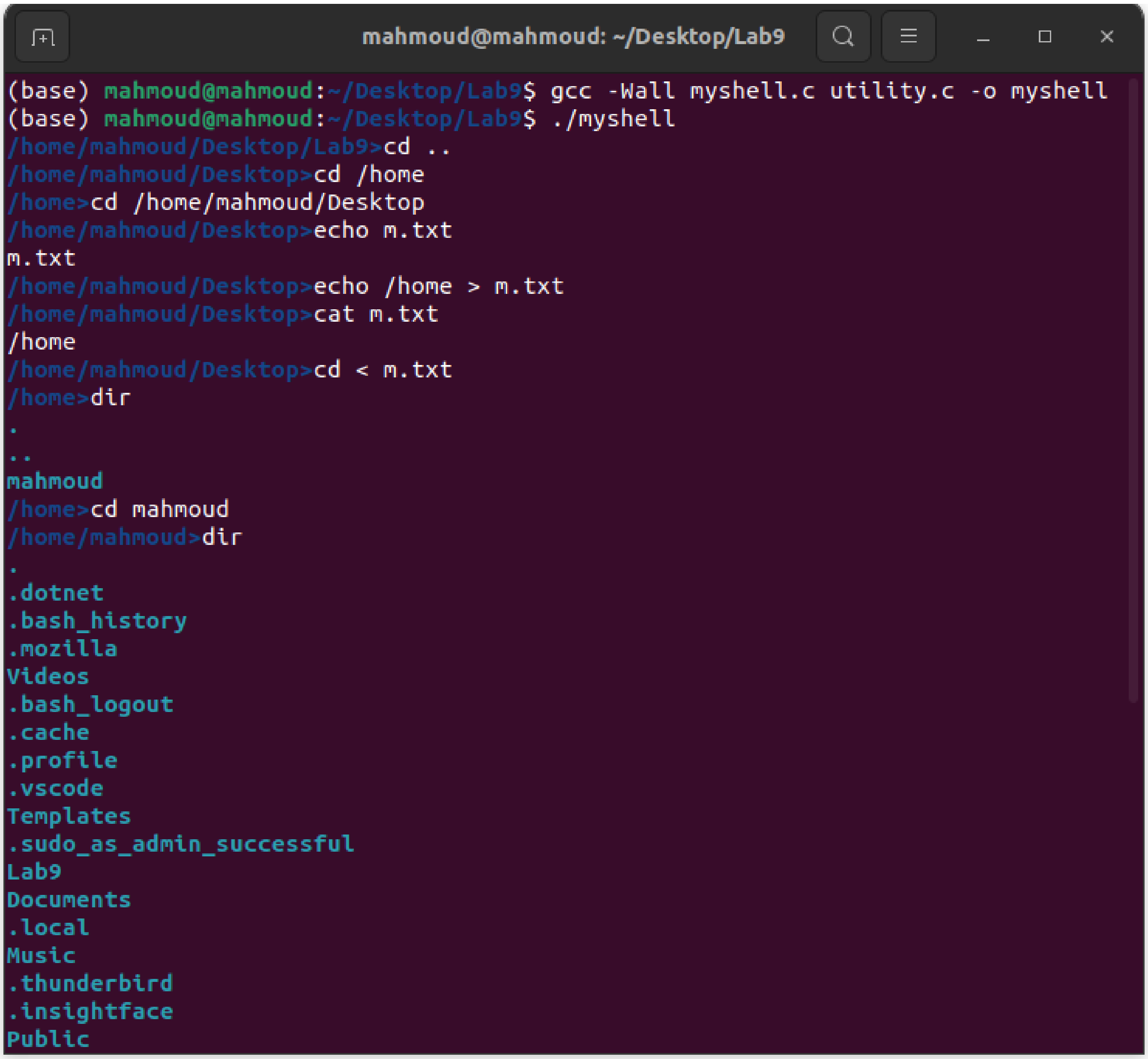
Compilation

```
Home/desctop> gcc -o myshell myshell.c
```

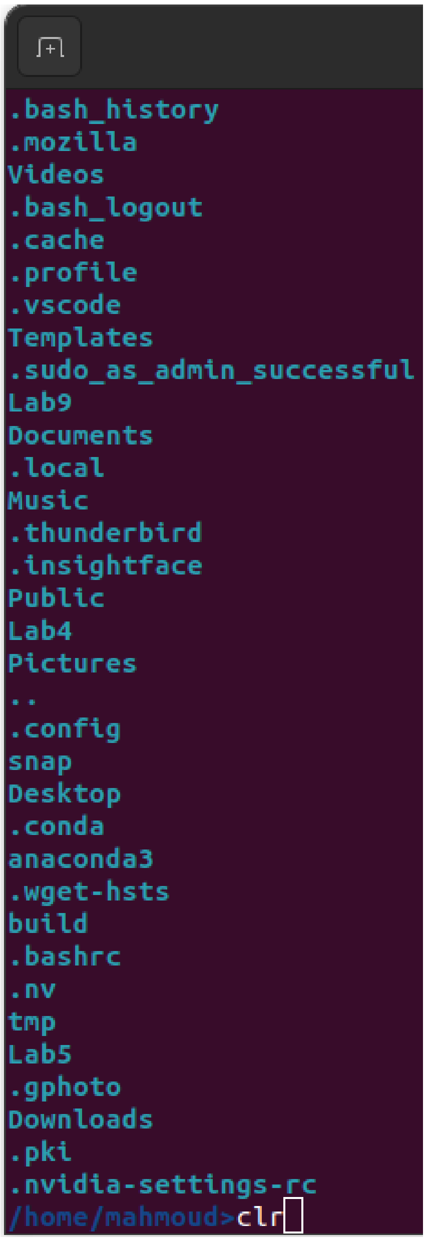
Usage

```
Home/desctop>./myshell # Interactive mode

Home/desctop>./myshell script.txt # Batch mode using script file
```

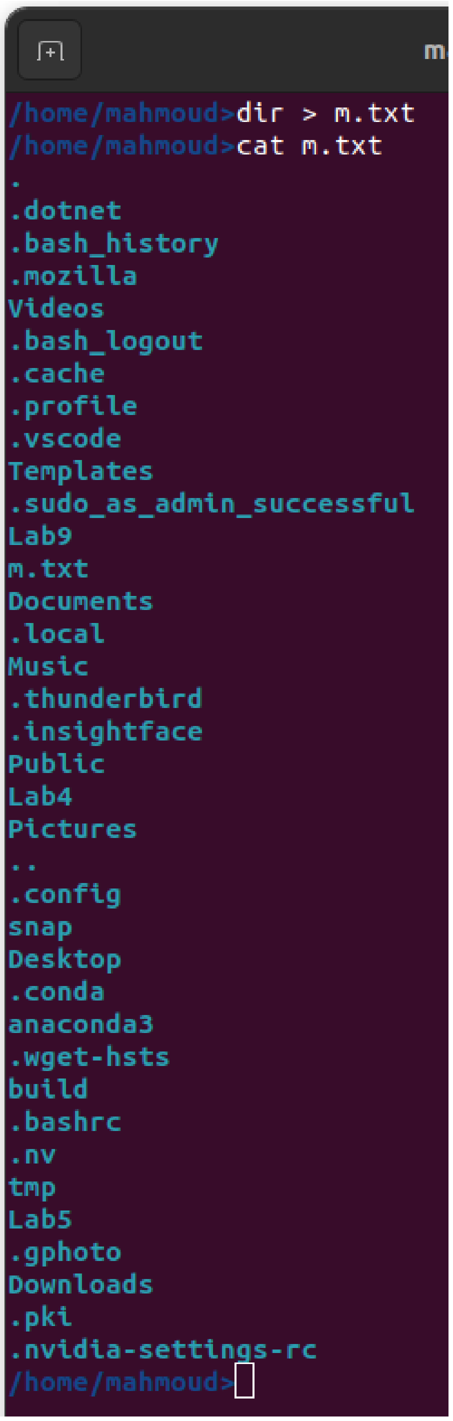
❖ *PROJECT OUTPUT*
Test myshell



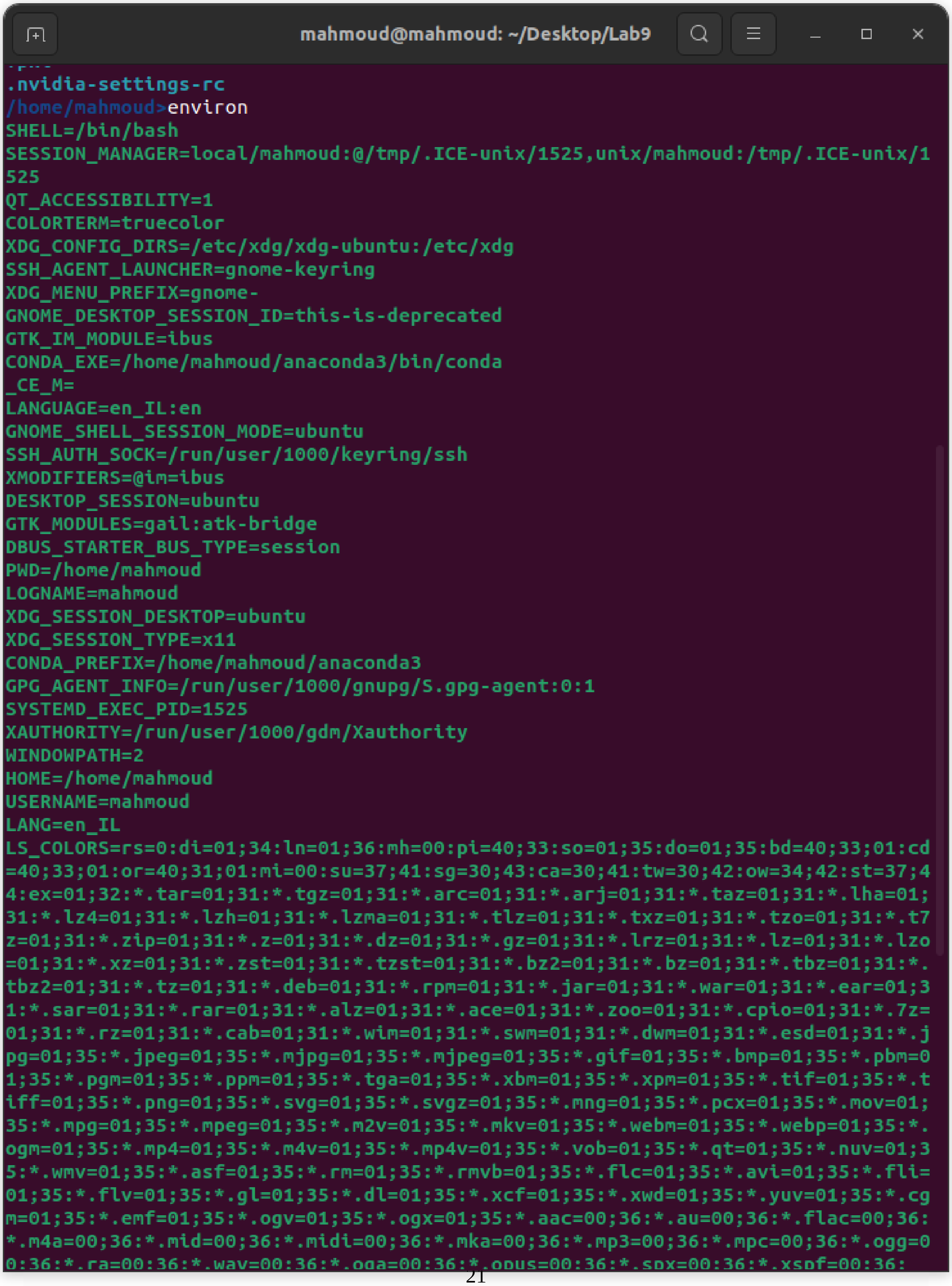
```
(base) mahmoud@mahmoud:~/Desktop/Lab9$ gcc -Wall myshell.c utility.c -o myshell
(base) mahmoud@mahmoud:~/Desktop/Lab9$ ./myshell
/home/mahmoud/Desktop/Lab9>cd ..
/home/mahmoud/Desktop>cd /home
/home>cd /home/mahmoud/Desktop
/home/mahmoud/Desktop>echo m.txt
m.txt
/home/mahmoud/Desktop>echo /home > m.txt
/home/mahmoud/Desktop>cat m.txt
/home
/home/mahmoud/Desktop>cd < m.txt
/home>dir
.
..
mahmoud
/home>cd mahmoud
/home/mahmoud>dir
.
.dotnet
.bash_history
.mozilla
Videos
.bash_logout
.cache
.profile
.vscode
Templates
.sudo_as_admin_successful
Lab9
Documents
.local
Music
.thunderbird
.insightface
Public
```



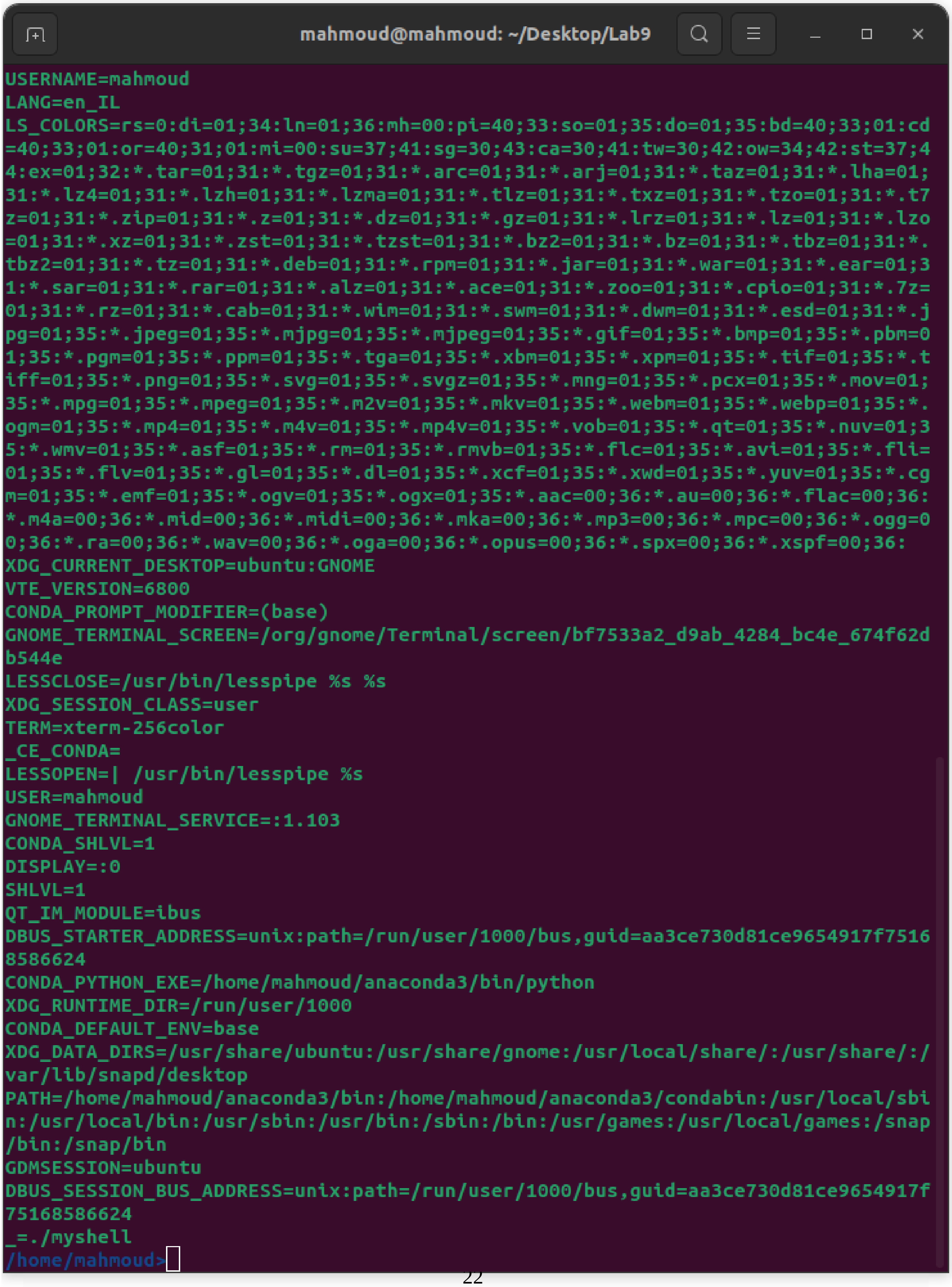
```
.bash_history
.mozilla
Videos
.bash_logout
.cache
.profile
.vscode
Templates
.sudo_as_admin_successful
Lab9
Documents
.local
Music
.thunderbird
.insightface
Public
Lab4
Pictures
..
.config
snap
Desktop
.conda
anaconda3
.wget-hsts
build
.bashrc
.nv
tmp
Lab5
.gphoto
Downloads
.pki
.nvidia-settings-rc
/home/mahmoud>clr
```



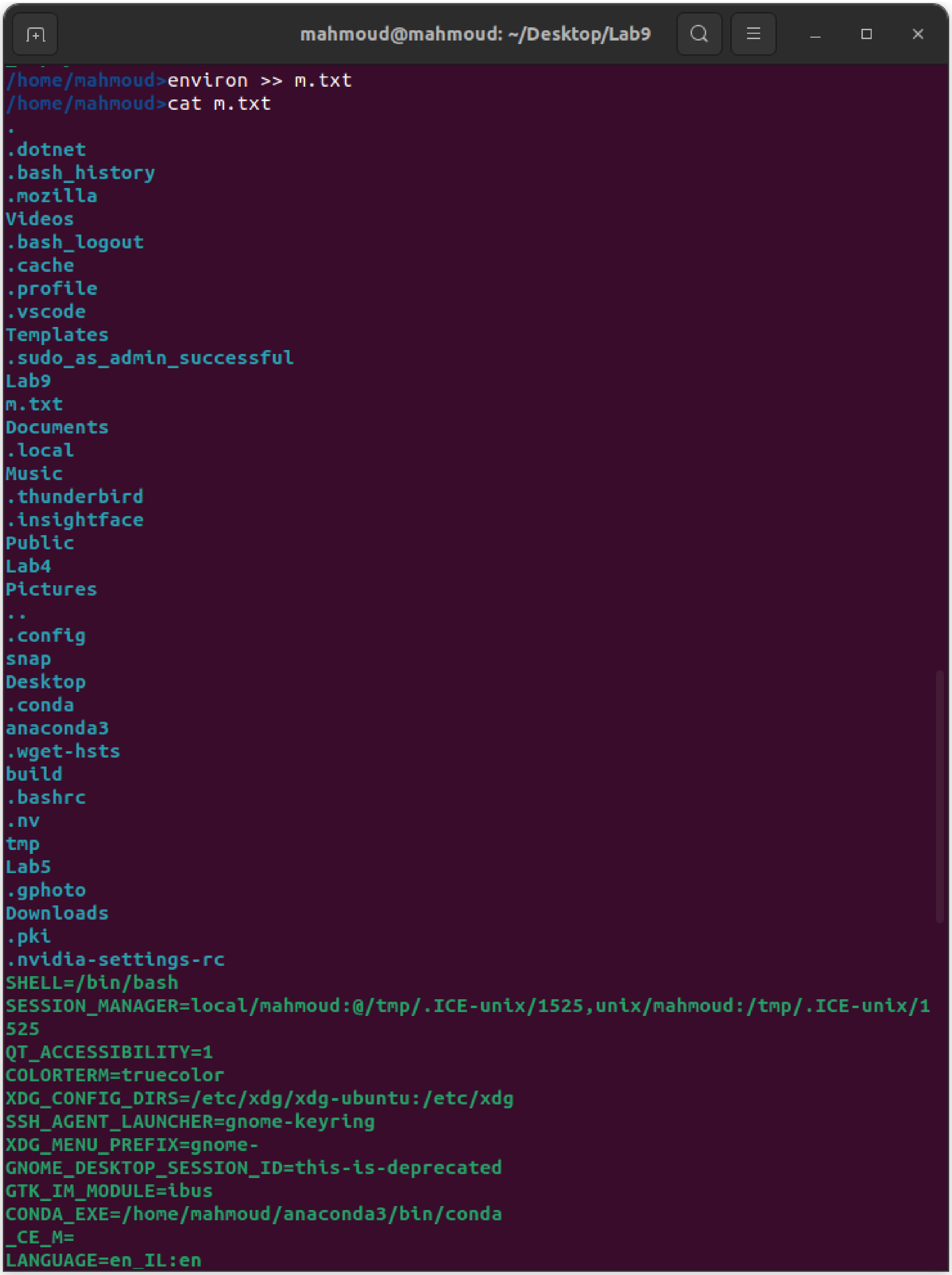
```
/home/mahmoud>dir > m.txt
/home/mahmoud>cat m.txt
.
.dotnet
.bash_history
.mozilla
Videos
.bash_logout
.cache
.profile
.vscode
Templates
.sudo_as_admin_successful
Lab9
m.txt
Documents
.local
Music
.thunderbird
.insightface
Public
Lab4
Pictures
..
.config
snap
Desktop
.conda
anaconda3
.wget-hsts
build
.bashrc
.nv
tmp
Lab5
.gphoto
Downloads
.pki
.nvidia-settings-rc
/home/mahmoud>
```


mahmoud@mahmoud: ~/Desktop/Lab9

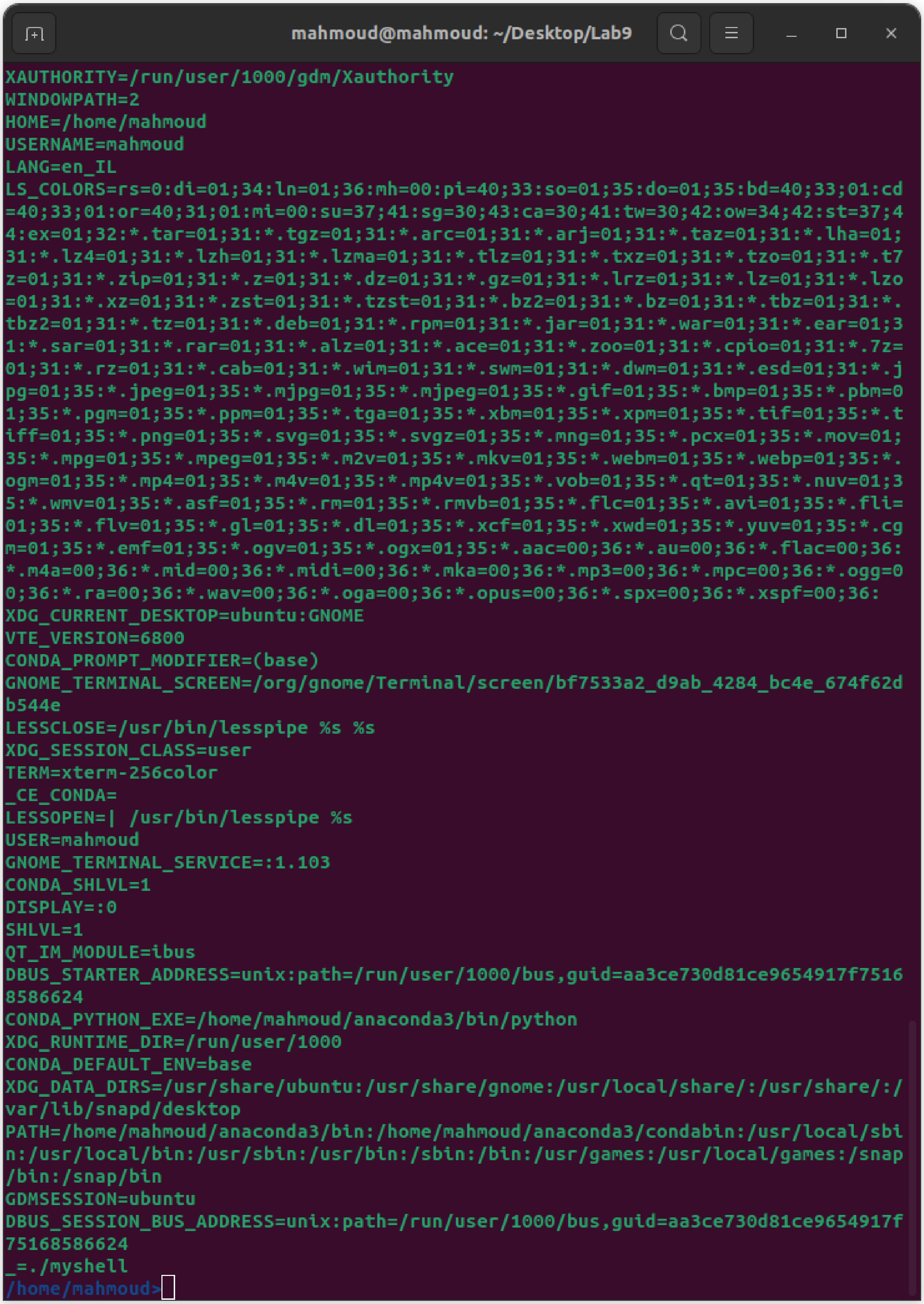
```
.nvidia-settings-rc
/home/mahmoud>environ
SHELL=/bin/bash
SESSION_MANAGER=local/mahmoud:@/tmp/.ICE-unix/1525,unix/mahmoud:/tmp/.ICE-unix/1
525
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
CONDA_EXE=/home/mahmoud/anaconda3/bin/conda
_CE_M=
LANGUAGE=en_IL:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/mahmoud
LOGNAME=mahmoud
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
CONDA_PREFIX=/home/mahmoud/anaconda3
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
SYSTEMD_EXEC_PID=1525
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/mahmoud
USERNAME=mahmoud
LANG=en_IL
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.
ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;3
5:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
m=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=0
0;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
```


mahmoud@mahmoud: ~/Desktop/Lab9

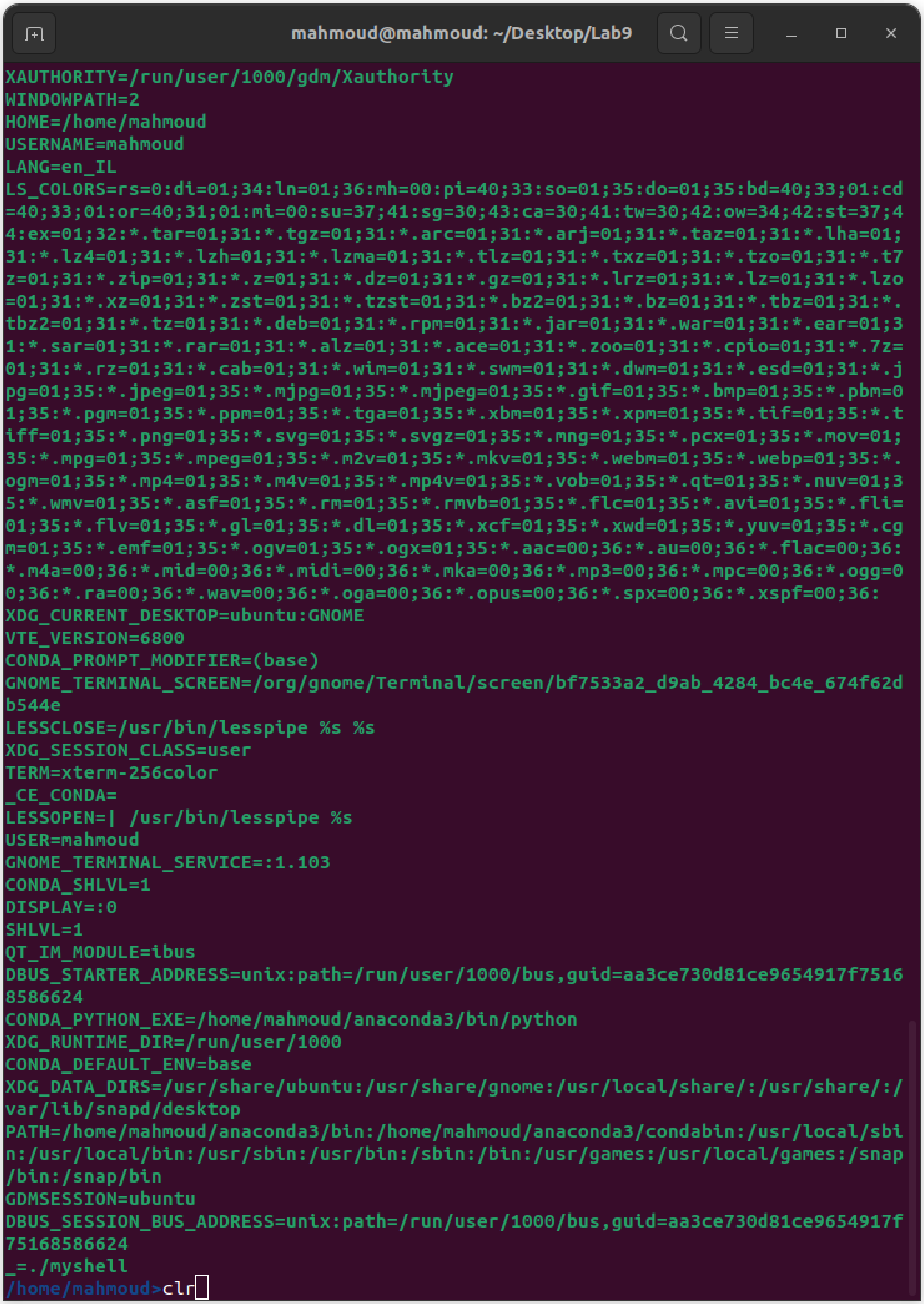
```
USERNAME=mahmoud
LANG=en_IL
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.
ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;3
5:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
m=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=0
0;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6800
CONDA_PROMPT_MODIFIER=(base)
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/bf7533a2_d9ab_4284_bc4e_674f62d
b544e
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
_CE_CONDA=
LESSOPEN=| /usr/bin/lesspipe %s
USER=mahmoud
GNOME_TERMINAL_SERVICE=:1.103
CONDA_SHLVL=1
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f7516
8586624
CONDA_PYTHON_EXE=/home/mahmoud/anaconda3/bin/python
XDG_RUNTIME_DIR=/run/user/1000
CONDA_DEFAULT_ENV=base
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/
var/lib/snapd/desktop
PATH=/home/mahmoud/anaconda3/bin:/home/mahmoud/anaconda3/condabin:/usr/local/sbi
n:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap
/bin:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f
75168586624
_=./myshell
/home/mahmoud>
```


mahmoud@mahmoud: ~/Desktop/Lab9

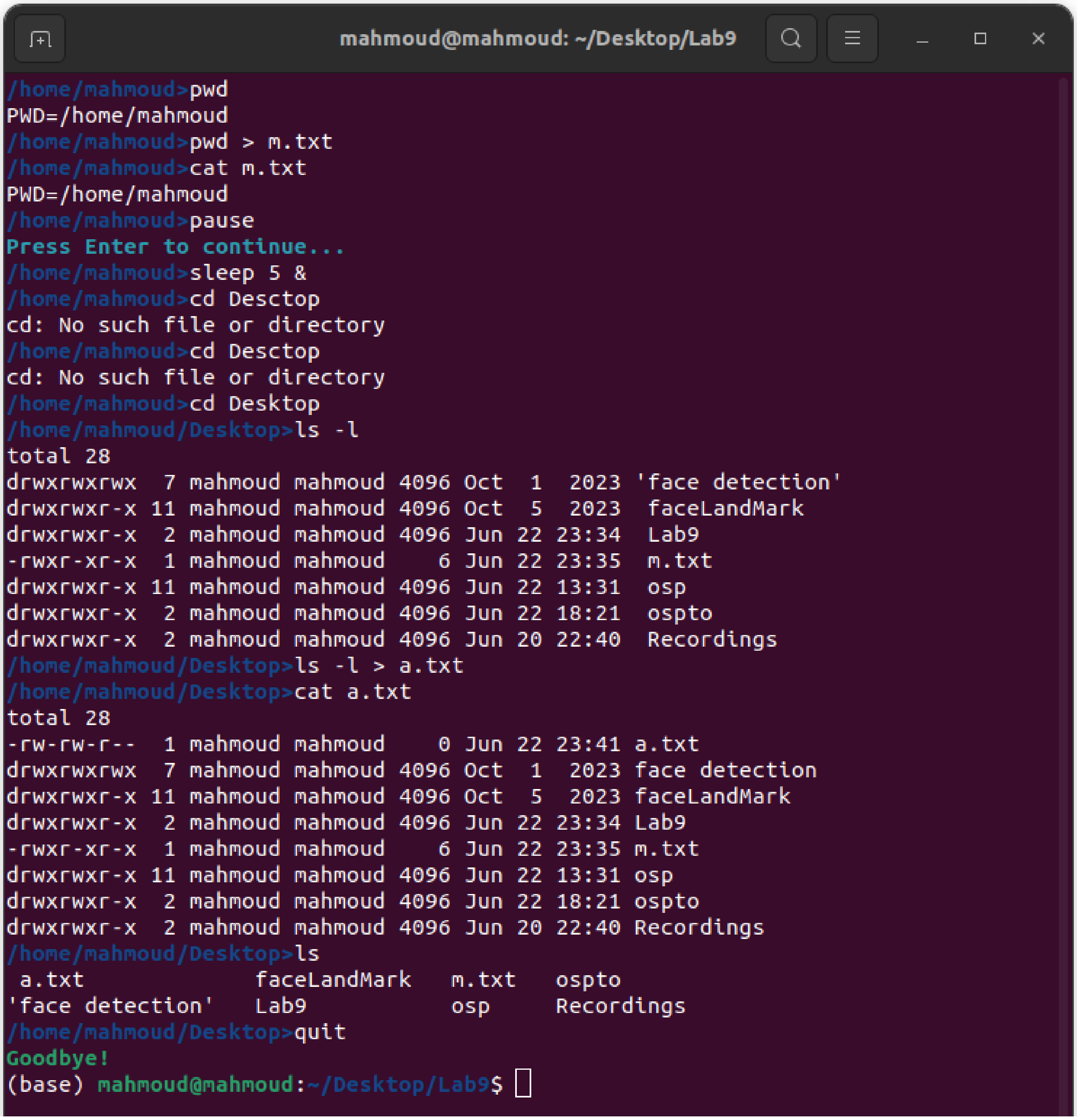
```
/home/mahmoud>environ >> m.txt
/home/mahmoud>cat m.txt
.
.dotnet
.bash_history
.mozilla
Videos
.bash_logout
.cache
.profile
.vscode
Templates
.sudo_as_admin_successful
Lab9
m.txt
Documents
.local
Music
.thunderbird
.insightface
Public
Lab4
Pictures
..
.config
snap
Desktop
.conda
anaconda3
.wget-hsts
build
.bashrc
.nv
tmp
Lab5
.gphoto
Downloads
.pki
.nvidia-settings-rc
SHELL=/bin/bash
SESSION_MANAGER=local/mahmoud:@/tmp/.ICE-unix/1525,unix/mahmoud:/tmp/.ICE-unix/1
525
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
CONDA_EXE=/home/mahmoud/anaconda3/bin/conda
_CE_M=
LANGUAGE=en_IL:en
```


mahmoud@mahmoud: ~/Desktop/Lab9

```
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/mahmoud
USERNAME=mahmoud
LANG=en_IL
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.
ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;3
5:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
m=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=0
0;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6800
CONDA_PROMPT_MODIFIER=(base)
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/bf7533a2_d9ab_4284_bc4e_674f62d
b544e
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
_CE_CONDA=
LESSOPEN=| /usr/bin/lesspipe %s
USER=mahmoud
GNOME_TERMINAL_SERVICE=:1.103
CONDA_SHLVL=1
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f7516
8586624
CONDA_PYTHON_EXE=/home/mahmoud/anaconda3/bin/python
XDG_RUNTIME_DIR=/run/user/1000
CONDA_DEFAULT_ENV=base
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/
var/lib/snapd/desktop
PATH=/home/mahmoud/anaconda3/bin:/home/mahmoud/anaconda3/condabin:/usr/local/sbi
n:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap
/bin:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f
75168586624
_=./myshell
/home/mahmoud>
```

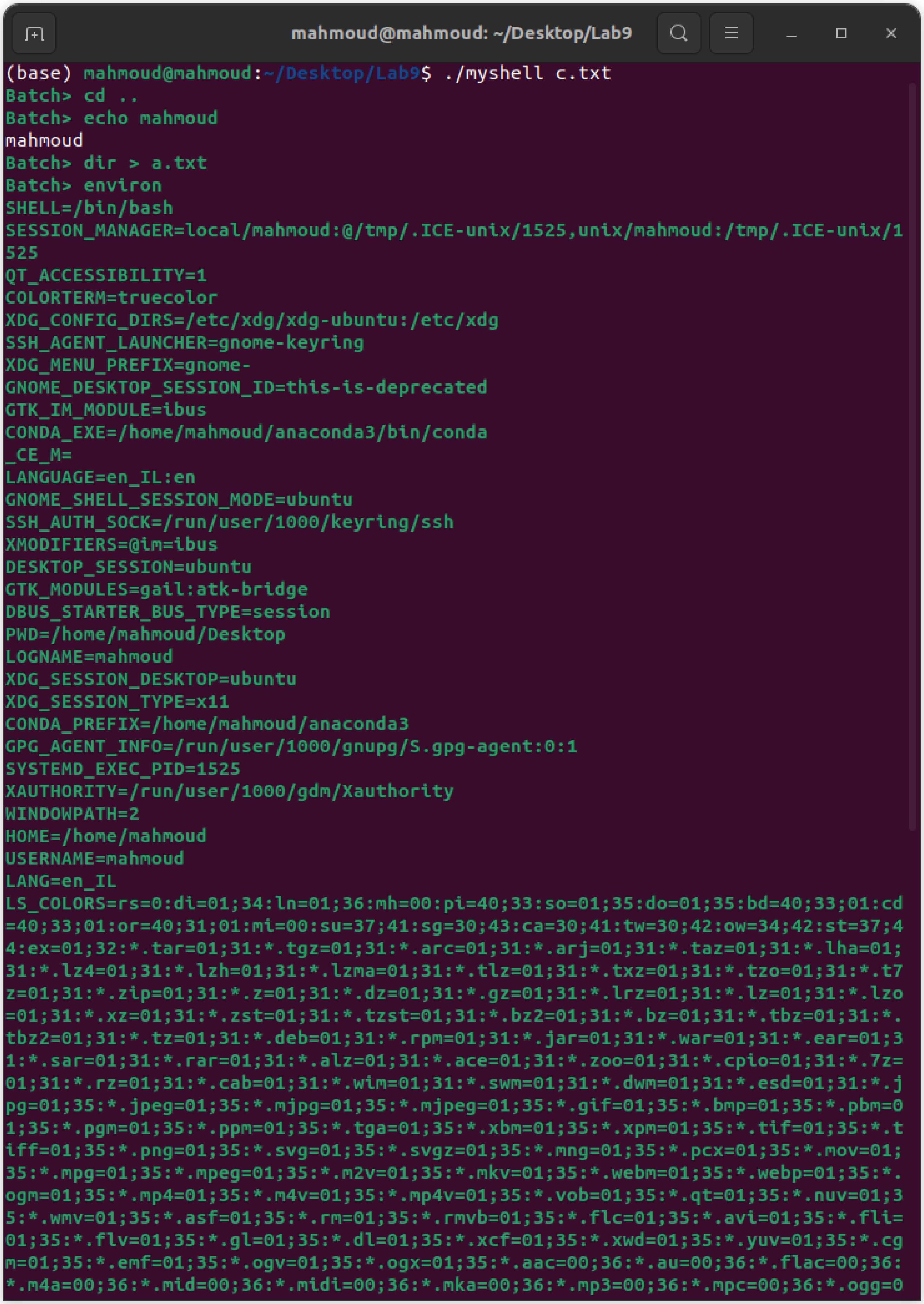


```
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/mahmoud
USERNAME=mahmoud
LANG=en_IL
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.
ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;3
5:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
m=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=0
0;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6800
CONDA_PROMPT_MODIFIER=(base)
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/bf7533a2_d9ab_4284_bc4e_674f62d
b544e
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
_CE_CONDA=
LESSOPEN=| /usr/bin/lesspipe %s
USER=mahmoud
GNOME_TERMINAL_SERVICE=:1.103
CONDA_SHLVL=1
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f7516
8586624
CONDA_PYTHON_EXE=/home/mahmoud/anaconda3/bin/python
XDG_RUNTIME_DIR=/run/user/1000
CONDA_DEFAULT_ENV=base
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/
var/lib/snapd/desktop
PATH=/home/mahmoud/anaconda3/bin:/home/mahmoud/anaconda3/condabin:/usr/local/sbi
n:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap
/bin:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f
75168586624
_=./myshell
/home/mahmoud>clr
```



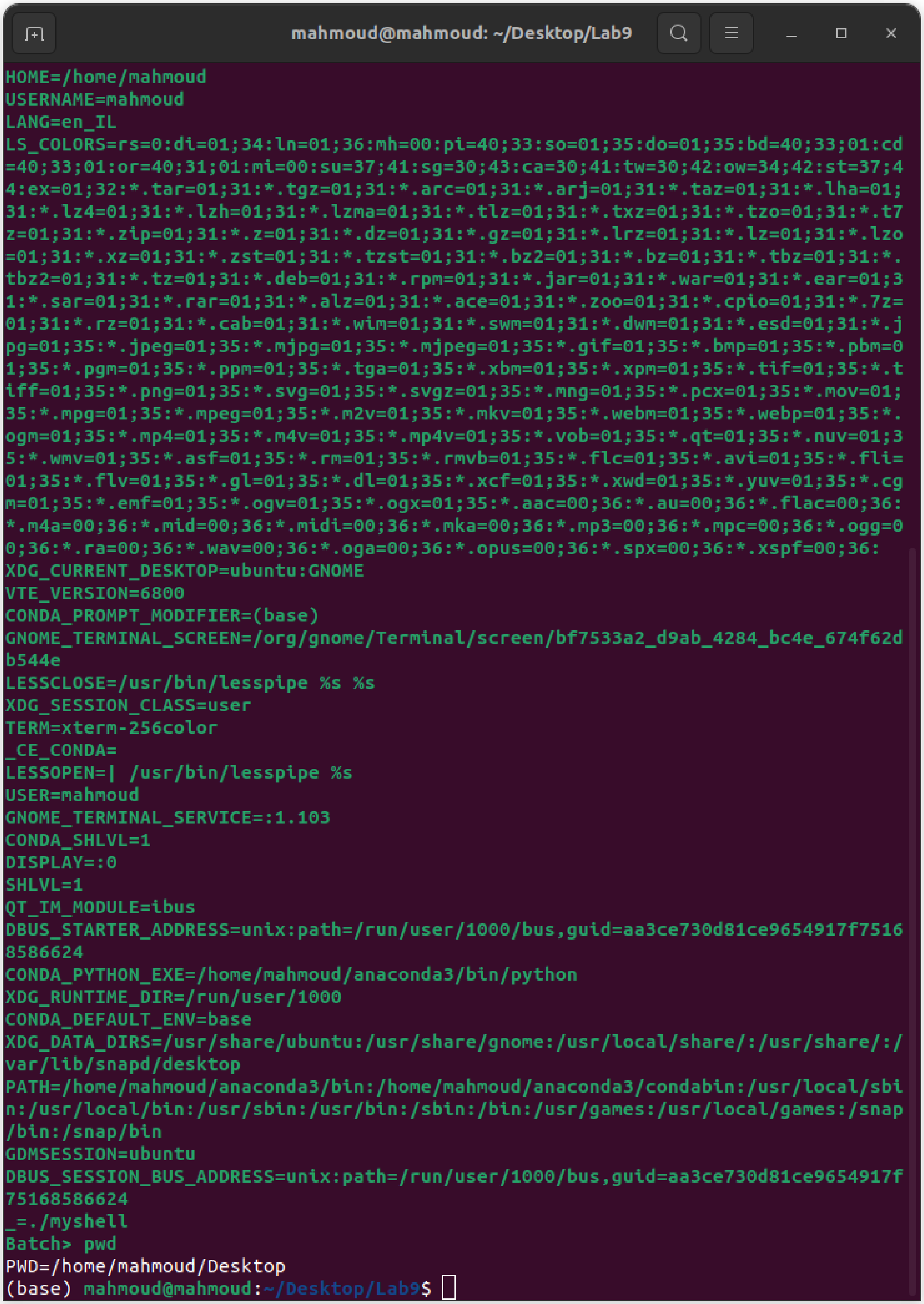
mahmoud@mahmoud: ~/Desktop/Lab9

```
/home/mahmoud>pwd
PWD=/home/mahmoud
/home/mahmoud>pwd > m.txt
/home/mahmoud>cat m.txt
PWD=/home/mahmoud
/home/mahmoud>pause
Press Enter to continue...
/home/mahmoud>sleep 5 &
/home/mahmoud>cd Desctop
cd: No such file or directory
/home/mahmoud>cd Desctop
cd: No such file or directory
/home/mahmoud>cd Desktop
/home/mahmoud/Desktop>ls -l
total 28
drwxrwxrwx  7 mahmoud mahmoud 4096 Oct  1  2023 'face detection'
drwxrwxr-x 11 mahmoud mahmoud 4096 Oct  5  2023  faceLandMark
drwxrwxr-x  2 mahmoud mahmoud 4096 Jun 22 23:34  Lab9
-rwxr-xr-x  1 mahmoud mahmoud    6 Jun 22 23:35  m.txt
drwxrwxr-x 11 mahmoud mahmoud 4096 Jun 22 13:31  osp
drwxrwxr-x  2 mahmoud mahmoud 4096 Jun 22 18:21  ospto
drwxrwxr-x  2 mahmoud mahmoud 4096 Jun 20 22:40  Recordings
/home/mahmoud/Desktop>ls -l > a.txt
/home/mahmoud/Desktop>cat a.txt
total 28
-rw-rw-r--  1 mahmoud mahmoud    0 Jun 22 23:41 a.txt
drwxrwxrwx  7 mahmoud mahmoud 4096 Oct  1  2023 face detection
drwxrwxr-x 11 mahmoud mahmoud 4096 Oct  5  2023 faceLandMark
drwxrwxr-x  2 mahmoud mahmoud 4096 Jun 22 23:34 Lab9
-rwxr-xr-x  1 mahmoud mahmoud    6 Jun 22 23:35 m.txt
drwxrwxr-x 11 mahmoud mahmoud 4096 Jun 22 13:31 osp
drwxrwxr-x  2 mahmoud mahmoud 4096 Jun 22 18:21 ospto
drwxrwxr-x  2 mahmoud mahmoud 4096 Jun 20 22:40 Recordings
/home/mahmoud/Desktop>ls
 a.txt             faceLandMark   m.txt   ospto
'face detection'   Lab9           osp     Recordings
/home/mahmoud/Desktop>quit
Goodbye!
(base) mahmoud@mahmoud:~/Desktop/Lab9$ 
```

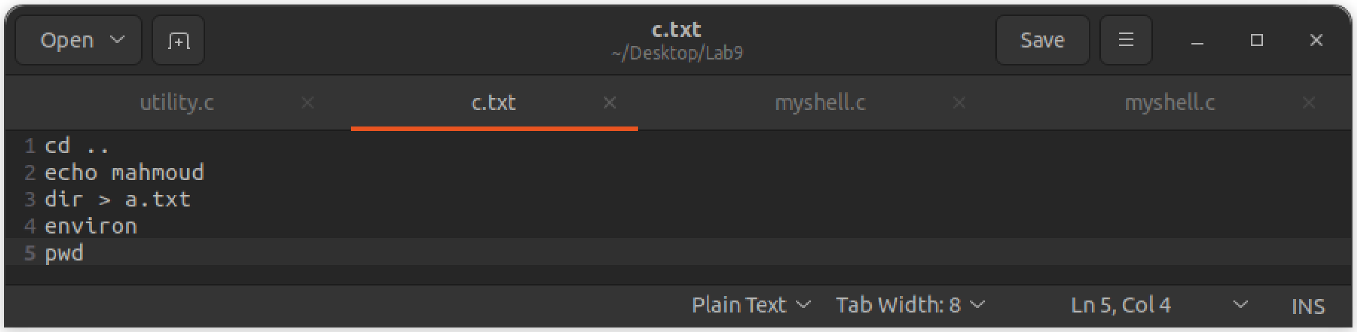


mahmoud@mahmoud: ~/Desktop/Lab9

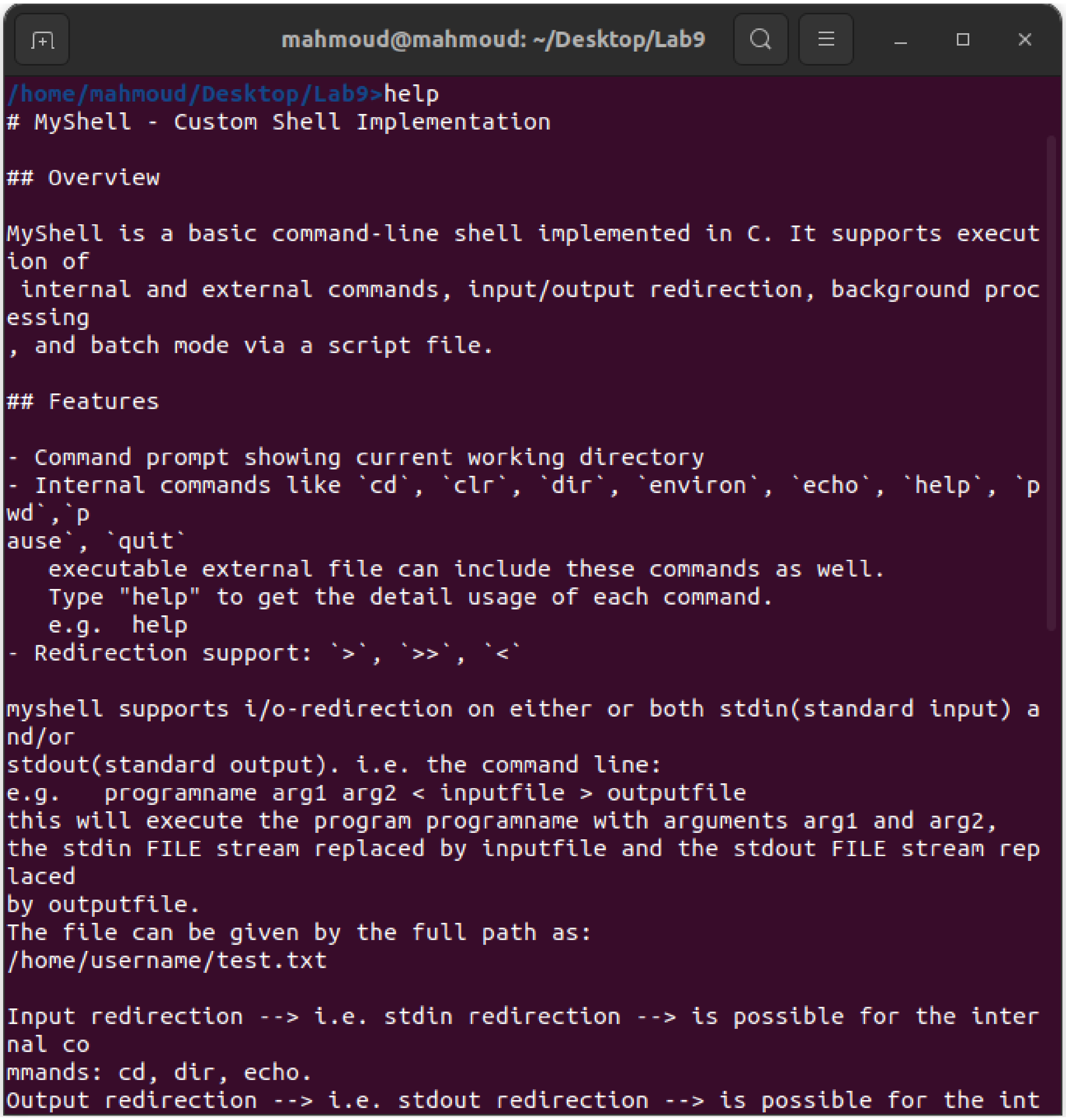
```
(base) mahmoud@mahmoud:~/Desktop/Lab9$ ./myshell c.txt
Batch> cd ..
Batch> echo mahmoud
mahmoud
Batch> dir > a.txt
Batch> environ
SHELL=/bin/bash
SESSION_MANAGER=local/mahmoud:@/tmp/.ICE-unix/1525,unix/mahmoud:/tmp/.ICE-unix/1
525
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
CONDA_EXE=/home/mahmoud/anaconda3/bin/conda
_CE_M=
LANGUAGE=en_IL:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/mahmoud/Desktop
LOGNAME=mahmoud
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
CONDA_PREFIX=/home/mahmoud/anaconda3
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
SYSTEMD_EXEC_PID=1525
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/mahmoud
USERNAME=mahmoud
LANG=en_IL
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.
ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;3
5:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
m=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=0
```



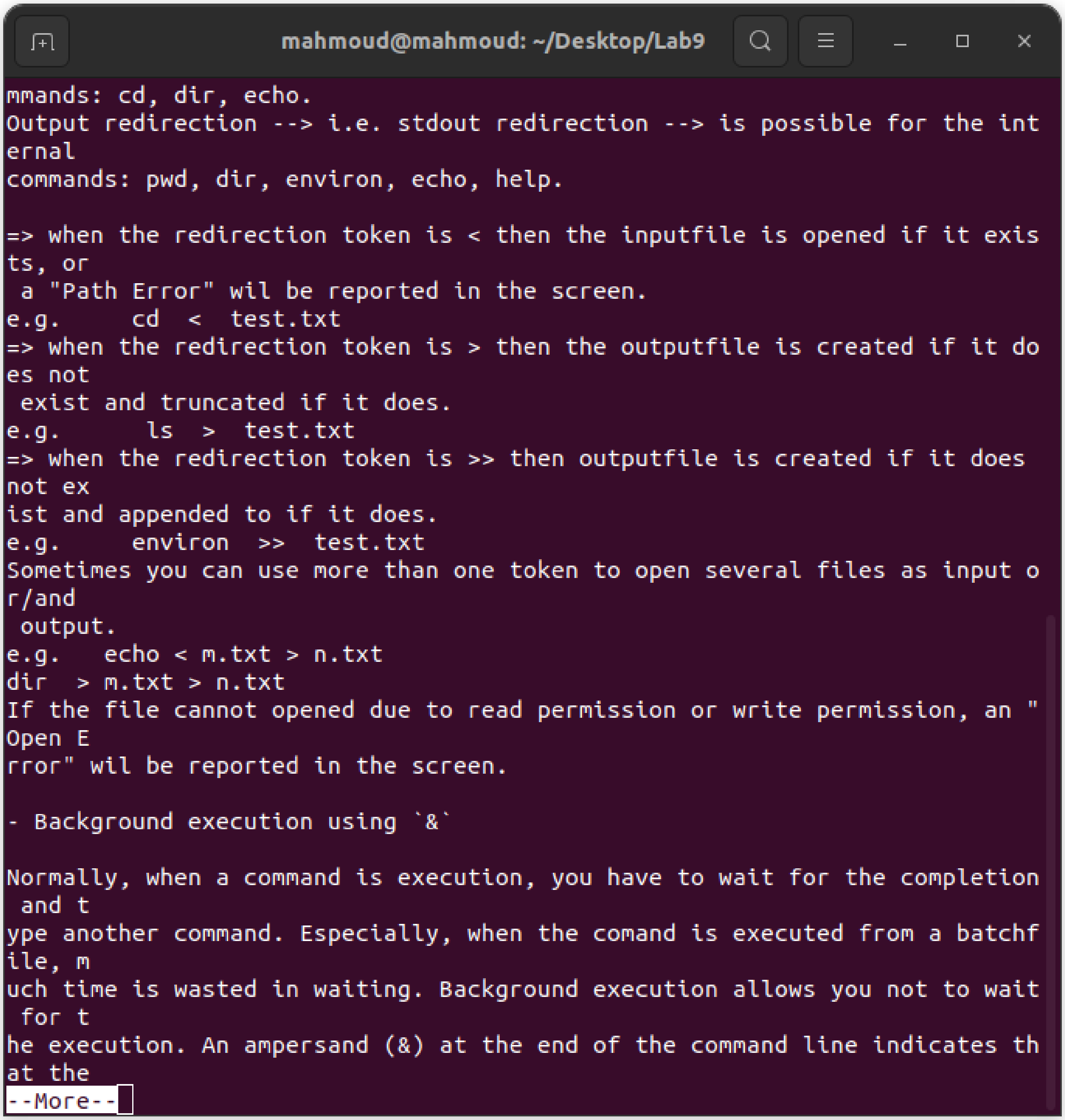
```
HOME=/home/mahmoud
USERNAME=mahmoud
LANG=en_IL
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo
=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;3
1:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=
01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
pg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=0
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.
ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;3
5:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=
01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cg
m=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=0
0;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6800
CONDA_PROMPT_MODIFIER=(base)
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/bf7533a2_d9ab_4284_bc4e_674f62d
b544e
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
_CE_CONDA=
LESSOPEN=| /usr/bin/lesspipe %s
USER=mahmoud
GNOME_TERMINAL_SERVICE=:1.103
CONDA_SHLVL=1
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f7516
8586624
CONDA_PYTHON_EXE=/home/mahmoud/anaconda3/bin/python
XDG_RUNTIME_DIR=/run/user/1000
CONDA_DEFAULT_ENV=base
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/
var/lib/snapd/desktop
PATH=/home/mahmoud/anaconda3/bin:/home/mahmoud/anaconda3/condabin:/usr/local/sbi
n:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap
/bin:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce9654917f
75168586624
_=./myshell
Batch> pwd
PWD=/home/mahmoud/Desktop
(base) mahmoud@mahmoud:~/Desktop/Lab9$ 
```



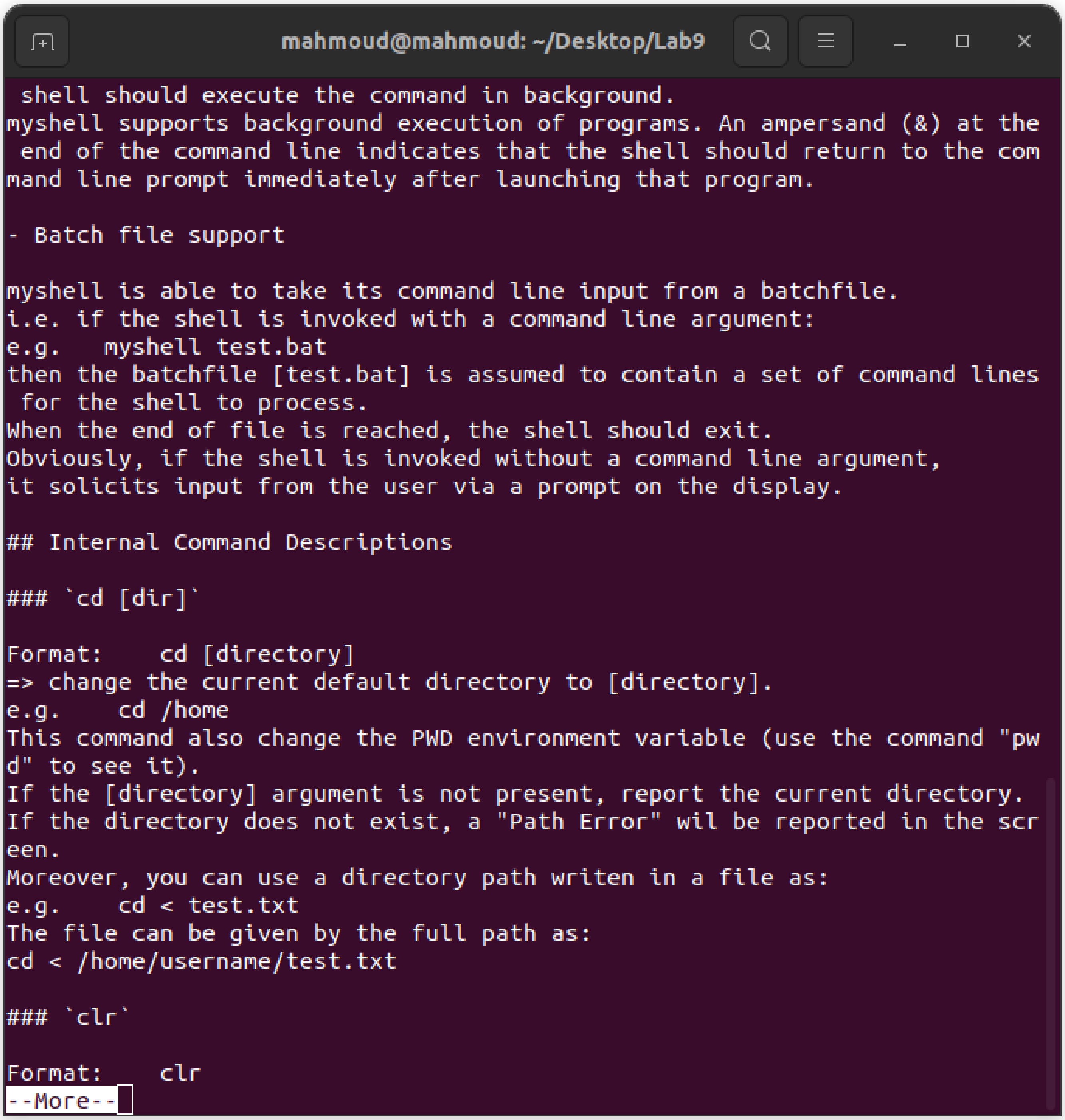
```
cd ..
echo mahmoud
dir > a.txt
environ
pwd
```



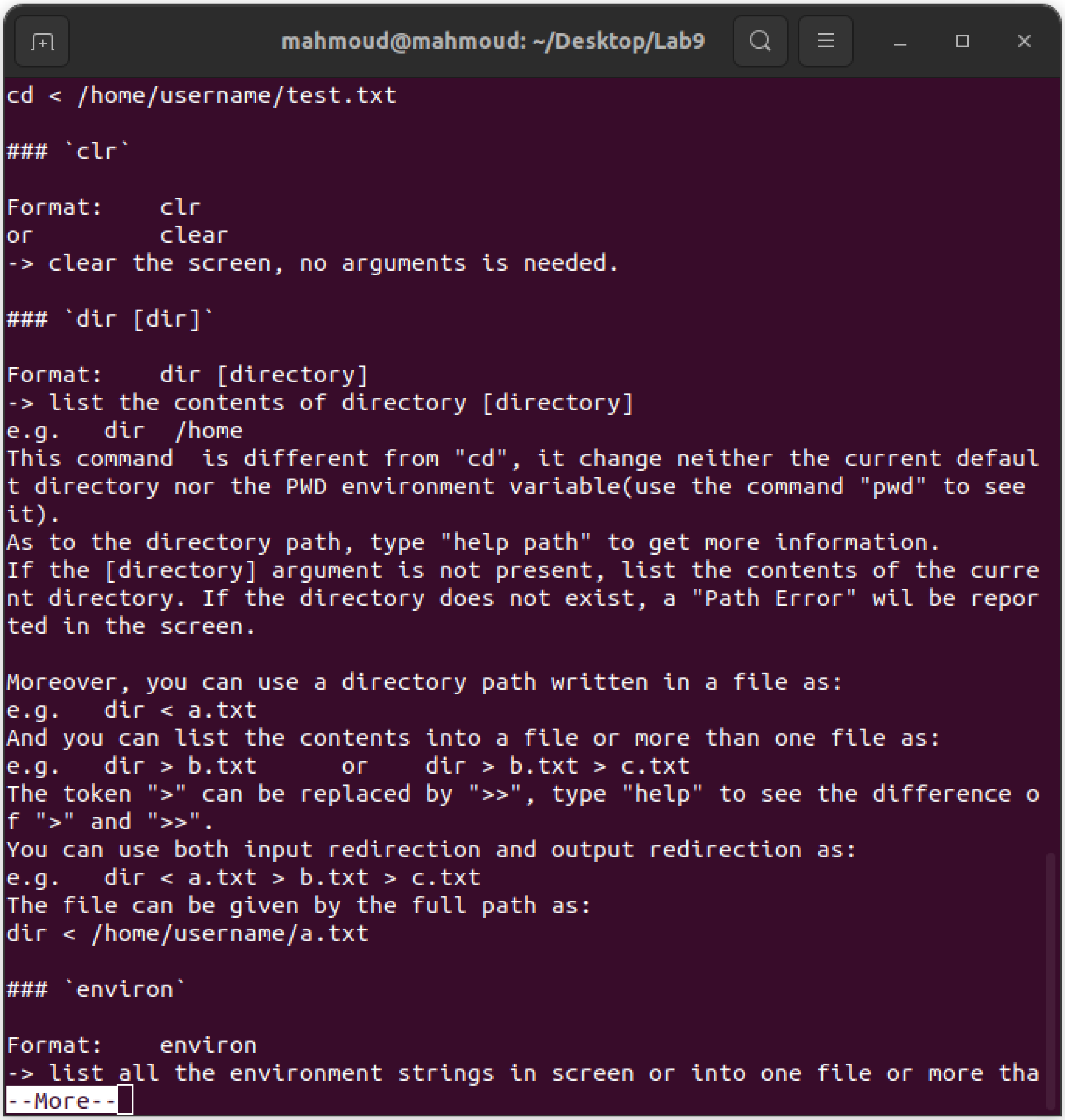
```
/home/mahmoud/Desktop/Lab9>help
# MyShell - Custom Shell Implementation

## Overview

MyShell is a basic command-line shell implemented in C. It supports execut
ion of
 internal and external commands, input/output redirection, background proc
essing
, and batch mode via a script file.

## Features

- Command prompt showing current working directory
- Internal commands like `cd`, `clr`, `dir`, `environ`, `echo`, `help`, `p
wd`,`p
ause`, `quit`
   executable external file can include these commands as well.
   Type "help" to get the detail usage of each command.
   e.g.  help
- Redirection support: `>`, `>>`, `<`

myshell supports i/o-redirection on either or both stdin(standard input) a
nd/or
stdout(standard output). i.e. the command line:
e.g.   programname arg1 arg2 < inputfile > outputfile
this will execute the program programname with arguments arg1 and arg2,
the stdin FILE stream replaced by inputfile and the stdout FILE stream rep
laced
by outputfile.
The file can be given by the full path as:
/home/username/test.txt

Input redirection --> i.e. stdin redirection --> is possible for the inter
nal co
mmands: cd, dir, echo.
Output redirection --> i.e. stdout redirection --> is possible for the int
```



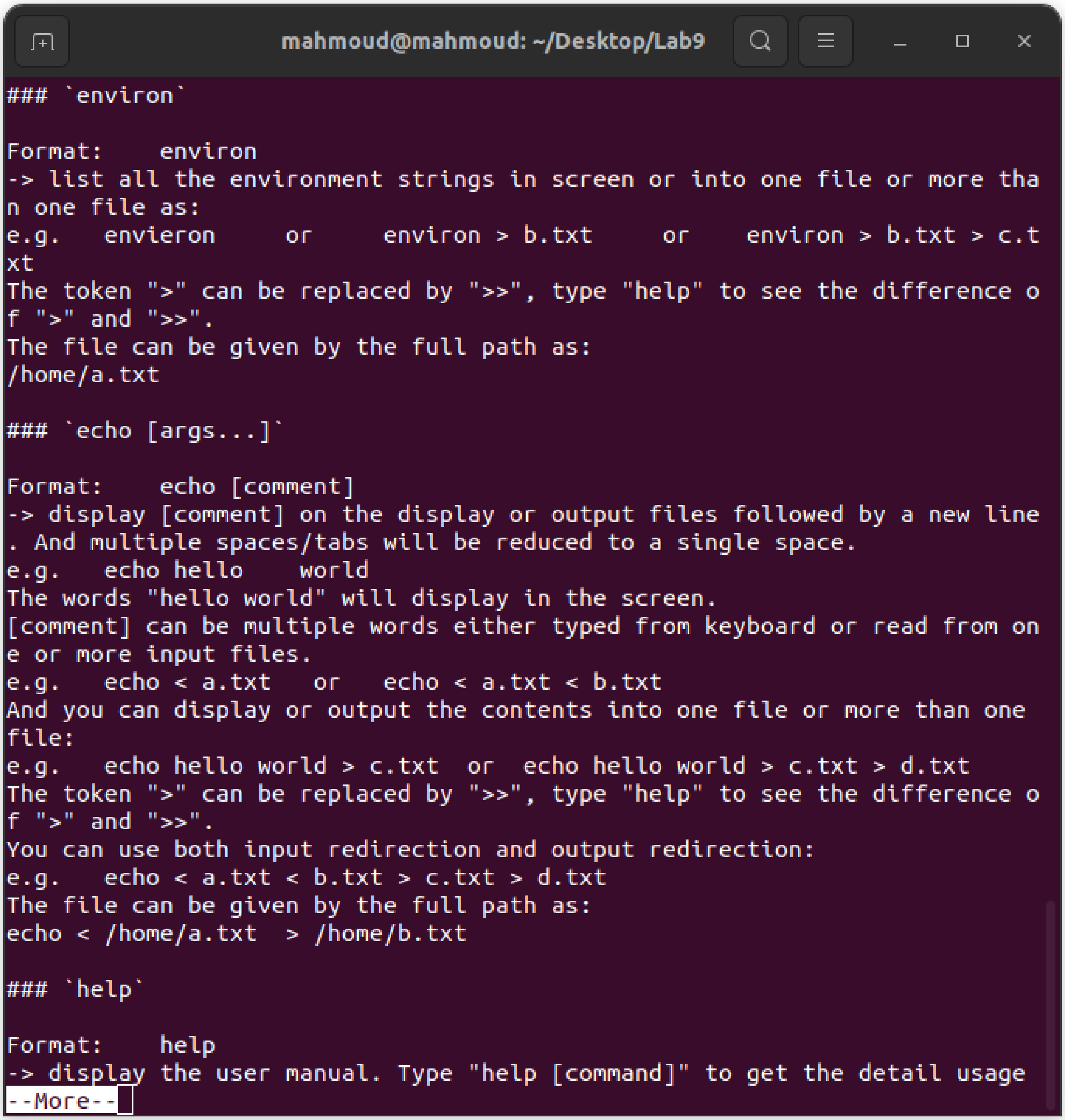
```
mmands: cd, dir, echo.
Output redirection --> i.e. stdout redirection --> is possible for the int
ernal
commands: pwd, dir, environ, echo, help.

=> when the redirection token is < then the inputfile is opened if it exis
ts, or
 a "Path Error" wil be reported in the screen.
e.g.     cd  <  test.txt
=> when the redirection token is > then the outputfile is created if it do
es not
 exist and truncated if it does.
e.g.      ls  >  test.txt
=> when the redirection token is >> then outputfile is created if it does
not ex
ist and appended to if it does.
e.g.     environ  >>  test.txt
Sometimes you can use more than one token to open several files as input o
r/and
 output.
e.g.   echo < m.txt > n.txt
dir  > m.txt > n.txt
If the file cannot opened due to read permission or write permission, an "
Open E
rror" wil be reported in the screen.

- Background execution using `&`

Normally, when a command is execution, you have to wait for the completion
 and t
ype another command. Especially, when the comand is executed from a batchf
ile, m
uch time is wasted in waiting. Background execution allows you not to wait
 for t
he execution. An ampersand (&) at the end of the command line indicates th
at the
--More--
```



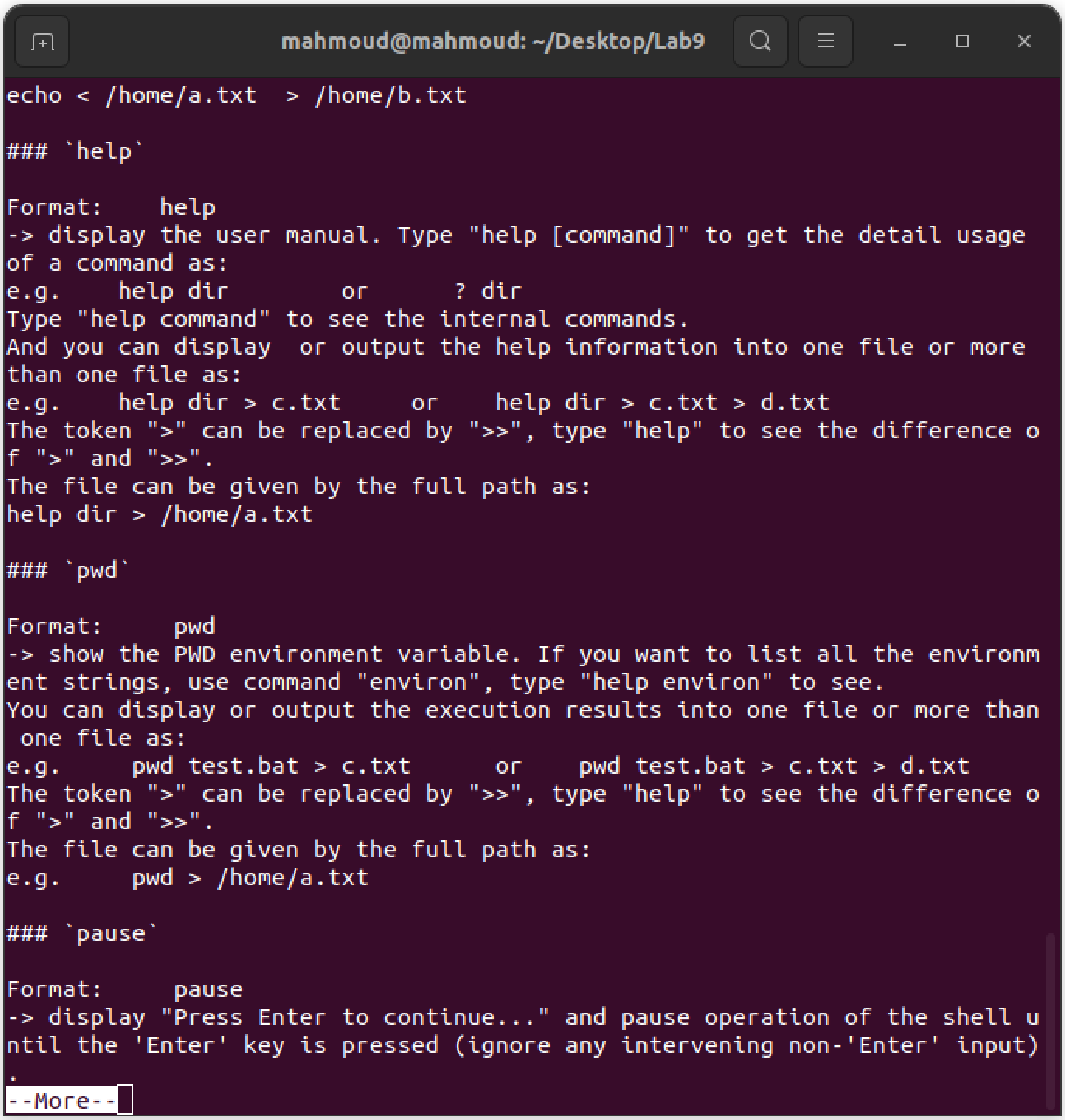
```
 shell should execute the command in background.
myshell supports background execution of programs. An ampersand (&) at the
 end of the command line indicates that the shell should return to the com
mand line prompt immediately after launching that program.

- Batch file support

myshell is able to take its command line input from a batchfile.
i.e. if the shell is invoked with a command line argument:
e.g.   myshell test.bat
then the batchfile [test.bat] is assumed to contain a set of command lines
 for the shell to process.
When the end of file is reached, the shell should exit.
Obviously, if the shell is invoked without a command line argument,
it solicits input from the user via a prompt on the display.

## Internal Command Descriptions

### `cd [dir]`

Format:    cd [directory]
=> change the current default directory to [directory].
e.g.    cd /home
This command also change the PWD environment variable (use the command "pw
d" to see it).
If the [directory] argument is not present, report the current directory.
If the directory does not exist, a "Path Error" wil be reported in the scr
een.
Moreover, you can use a directory path writen in a file as:
e.g.    cd < test.txt
The file can be given by the full path as:
cd < /home/username/test.txt

### `clr`

Format:    clr
--More--
```



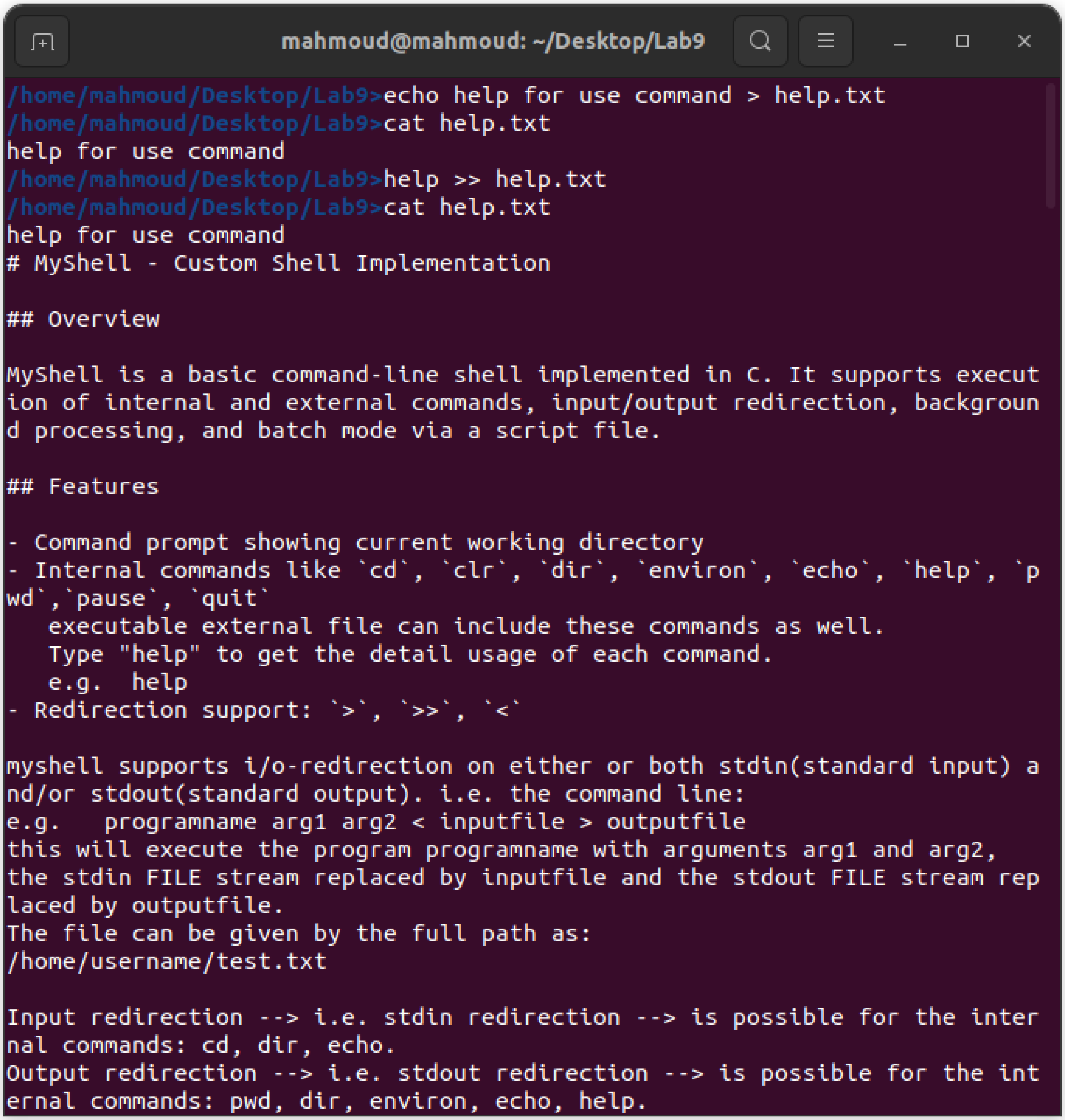
```
cd < /home/username/test.txt

### `clr`

Format:    clr
or         clear
-> clear the screen, no arguments is needed.

### `dir [dir]`

Format:    dir [directory]
-> list the contents of directory [directory]
e.g.   dir  /home
This command  is different from "cd", it change neither the current defaul
t directory nor the PWD environment variable(use the command "pwd" to see
it).
As to the directory path, type "help path" to get more information.
If the [directory] argument is not present, list the contents of the curre
nt directory. If the directory does not exist, a "Path Error" wil be repor
ted in the screen.

Moreover, you can use a directory path written in a file as:
e.g.   dir < a.txt
And you can list the contents into a file or more than one file as:
e.g.   dir > b.txt      or    dir > b.txt > c.txt
The token ">" can be replaced by ">>", type "help" to see the difference o
f ">" and ">>".
You can use both input redirection and output redirection as:
e.g.   dir < a.txt > b.txt > c.txt
The file can be given by the full path as:
dir < /home/username/a.txt

### `environ`

Format:    environ
-> list all the environment strings in screen or into one file or more tha
--More--
```



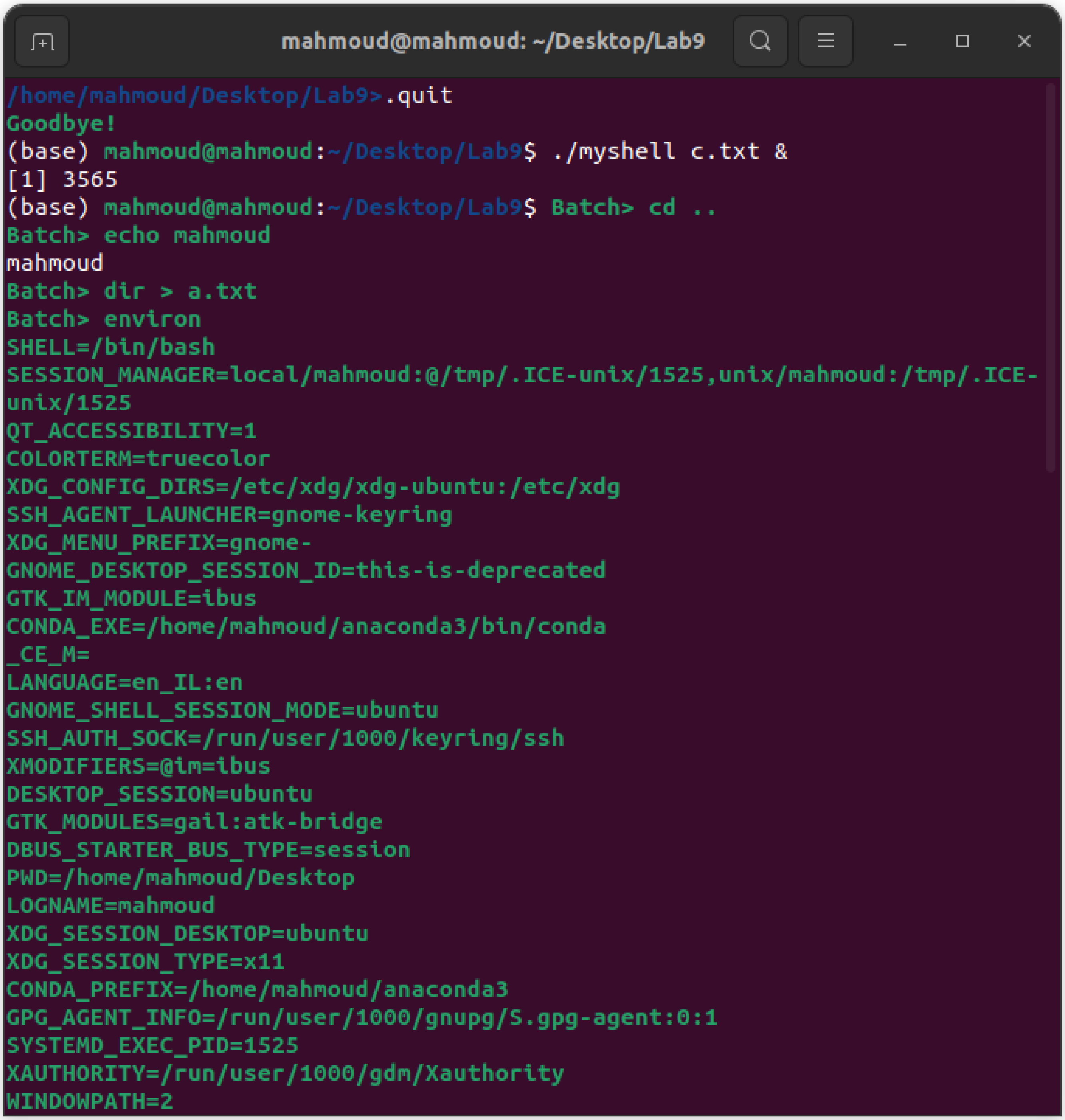
```
### `environ`

Format:    environ
-> list all the environment strings in screen or into one file or more tha
n one file as:
e.g.   envieron     or     environ > b.txt     or    environ > b.txt > c.t
xt
The token ">" can be replaced by ">>", type "help" to see the difference o
f ">" and ">>".
The file can be given by the full path as:
/home/a.txt

### `echo [args...]`

Format:    echo [comment]
-> display [comment] on the display or output files followed by a new line
. And multiple spaces/tabs will be reduced to a single space.
e.g.   echo hello    world
The words "hello world" will display in the screen.
[comment] can be multiple words either typed from keyboard or read from on
e or more input files.
e.g.   echo < a.txt   or   echo < a.txt < b.txt
And you can display or output the contents into one file or more than one
file:
e.g.   echo hello world > c.txt  or  echo hello world > c.txt > d.txt
The token ">" can be replaced by ">>", type "help" to see the difference o
f ">" and ">>".
You can use both input redirection and output redirection:
e.g.   echo < a.txt < b.txt > c.txt > d.txt
The file can be given by the full path as:
echo < /home/a.txt  > /home/b.txt

### `help`

Format:    help
-> display the user manual. Type "help [command]" to get the detail usage
--More--
```


mahmoud@mahmoud: ~/Desktop/Lab9

```
echo < /home/a.txt  > /home/b.txt

### `help`

Format:    help
-> display the user manual. Type "help [command]" to get the detail usage
of a command as:
e.g.    help dir        or      ? dir
Type "help command" to see the internal commands.
And you can display  or output the help information into one file or more
than one file as:
e.g.    help dir > c.txt     or    help dir > c.txt > d.txt
The token ">" can be replaced by ">>", type "help" to see the difference o
f ">" and ">>".
The file can be given by the full path as:
help dir > /home/a.txt

### `pwd`

Format:     pwd
-> show the PWD environment variable. If you want to list all the environm
ent strings, use command "environ", type "help environ" to see.
You can display or output the execution results into one file or more than
 one file as:
e.g.     pwd test.bat > c.txt      or    pwd test.bat > c.txt > d.txt
The token ">" can be replaced by ">>", type "help" to see the difference o
f ">" and ">>".
The file can be given by the full path as:
e.g.     pwd > /home/a.txt

### `pause`

Format:     pause
-> display "Press Enter to continue..." and pause operation of the shell u
ntil the 'Enter' key is pressed (ignore any intervening non-'Enter' input)
.
--More--
```

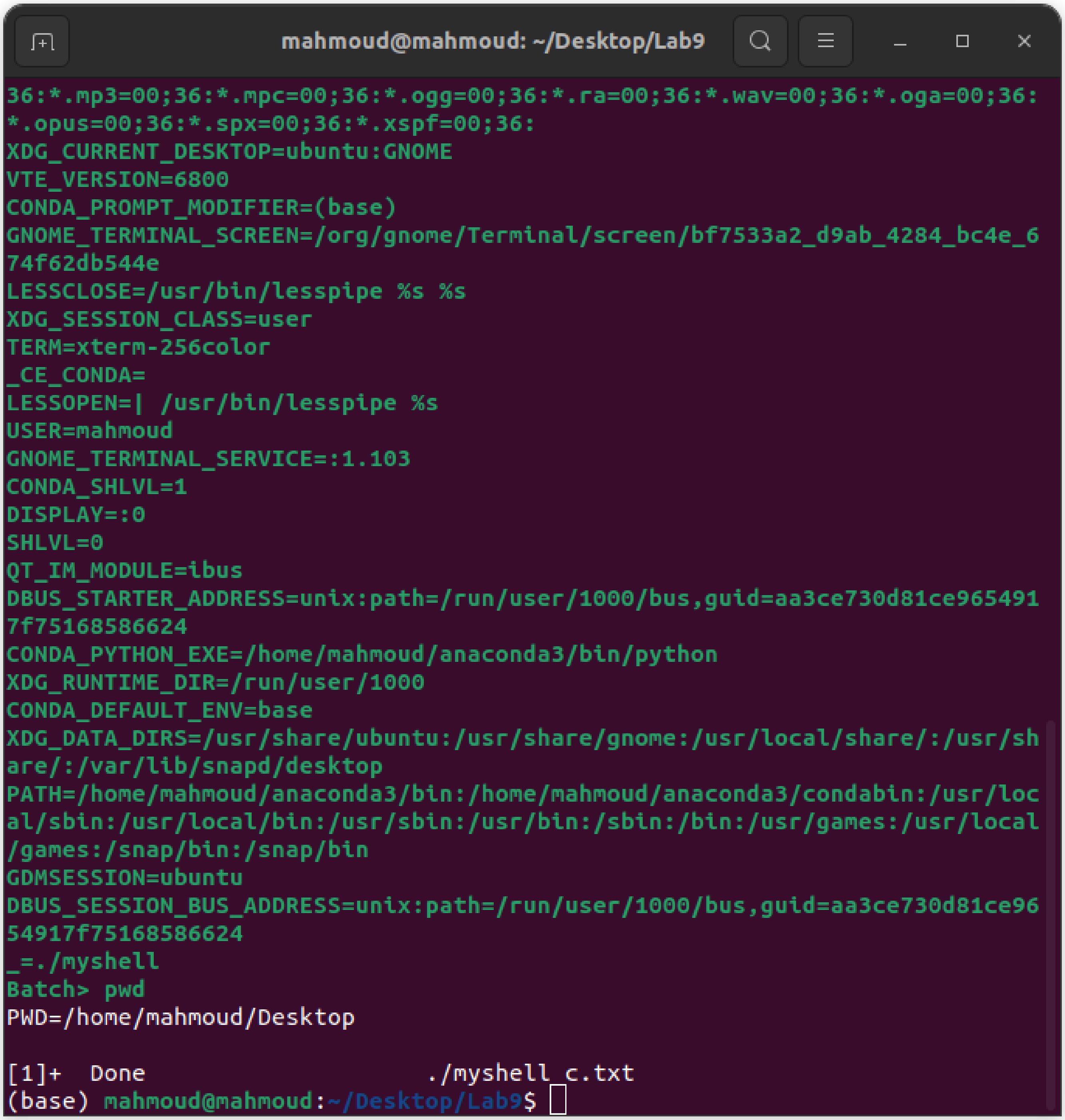


```
/home/mahmoud/Desktop/Lab9>echo help for use command > help.txt
/home/mahmoud/Desktop/Lab9>cat help.txt
help for use command
/home/mahmoud/Desktop/Lab9>help >> help.txt
/home/mahmoud/Desktop/Lab9>cat help.txt
help for use command
# MyShell - Custom Shell Implementation

## Overview

MyShell is a basic command-line shell implemented in C. It supports execut
ion of internal and external commands, input/output redirection, backgroun
d processing, and batch mode via a script file.

## Features

- Command prompt showing current working directory
- Internal commands like `cd`, `clr`, `dir`, `environ`, `echo`, `help`, `p
wd`,`pause`, `quit`
   executable external file can include these commands as well.
   Type "help" to get the detail usage of each command.
   e.g.  help
- Redirection support: `>`, `>>`, `<`

myshell supports i/o-redirection on either or both stdin(standard input) a
nd/or stdout(standard output). i.e. the command line:
e.g.   programname arg1 arg2 < inputfile > outputfile
this will execute the program programname with arguments arg1 and arg2,
the stdin FILE stream replaced by inputfile and the stdout FILE stream rep
laced by outputfile.
The file can be given by the full path as:
/home/username/test.txt

Input redirection --> i.e. stdin redirection --> is possible for the inter
nal commands: cd, dir, echo.
Output redirection --> i.e. stdout redirection --> is possible for the int
ernal commands: pwd, dir, environ, echo, help.
```



mahmoud@mahmoud: ~/Desktop/Lab9

```
/home/mahmoud/Desktop/Lab9>.quit
Goodbye!
(base) mahmoud@mahmoud:~/Desktop/Lab9$ ./myshell c.txt &
[1] 3565
(base) mahmoud@mahmoud:~/Desktop/Lab9$ Batch> cd ..
Batch> echo mahmoud
mahmoud
Batch> dir > a.txt
Batch> environ
SHELL=/bin/bash
SESSION_MANAGER=local/mahmoud:@/tmp/.ICE-unix/1525,unix/mahmoud:/tmp/.ICE-
unix/1525
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GTK_IM_MODULE=ibus
CONDA_EXE=/home/mahmoud/anaconda3/bin/conda
_CE_M=
LANGUAGE=en_IL:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/mahmoud/Desktop
LOGNAME=mahmoud
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
CONDA_PREFIX=/home/mahmoud/anaconda3
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
SYSTEMD_EXEC_PID=1525
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
```



mahmoud@mahmoud: ~/Desktop/Lab9

```
36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:
*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6800
CONDA_PROMPT_MODIFIER=(base)
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/bf7533a2_d9ab_4284_bc4e_6
74f62db544e
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
_CE_CONDA=
LESSOPEN=| /usr/bin/lesspipe %s
USER=mahmoud
GNOME_TERMINAL_SERVICE=:1.103
CONDA_SHLVL=1
DISPLAY=:0
SHLVL=0
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce965491
7f75168586624
CONDA_PYTHON_EXE=/home/mahmoud/anaconda3/bin/python
XDG_RUNTIME_DIR=/run/user/1000
CONDA_DEFAULT_ENV=base
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/sh
are/:/var/lib/snapd/desktop
PATH=/home/mahmoud/anaconda3/bin:/home/mahmoud/anaconda3/condabin:/usr/loc
al/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local
/games:/snap/bin:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=aa3ce730d81ce96
54917f75168586624
_=./myshell
Batch> pwd
PWD=/home/mahmoud/Desktop

[1]+  Done                    ./myshell c.txt
(base) mahmoud@mahmoud:~/Desktop/Lab9$ 
```

❖ *CONCLUSION*

This was a simple project, So it can be expanding to include more complex shell with various of advanced commands.

Overall, this project really helped us gain hands-on skills and experience necessary to implement the basic instructions for my developed shell. In addition these skills as well as the project can be an aiding for us in having an advantage over other entry level engineers searching.

❖ *REFERENCES*

1. **(1):** https://www.techopedia.com/
2. **(2):** https://indradhanush.github.io/blog/writing-a-unix-shell-part-1/
3. **General References:** Operating Systems Labs