
Project Name

Linux Terminal Clone

Introduction

A Linux terminal is a command-line interface that enables us to execute a wide range of Linux commands and programs. Using our own custom build command, we produced a miniature terminal that looked like a Linux terminal which we named “Turtle Shell”. Python was used to build this. We designed the terminal stage to handle crucial commands like pwd, ls, cd, date, rm, touch, mkdir, cls, and others. Users will find it easy to carry out common file and directory operations using this terminal.

Objectives

- To go over the basic concepts and Linux terminal commands.
- To understand the development of the Linux command line interface and its history.
- To illustrate navigating and interacting with files and directories using the terminal.
- To connect to the terminal from your computer and alter its behavior and appearance to your liking.
- To demonstrate how to use system calls to combine commands and process data.
- To investigate some of the fundamental tools and characteristics of the terminal, such as command history, text generation, file location, file operations, and file and directory operations.

Proposed Solution

The project launches by defining several actions to deal with various commands. These functions use the OS module to communicate with the operating system and execute system calls. To execute the commands, we used a variety of system calls, including **fork()**, **getcwd()**, **chdir()**, **read()**, **write()**, **open()**, etc.

Here is an overview of how the project operates:

1. A welcome message is displayed when the shell is launched using the "**printGreeting()**" method. To serve as the program's entry point, we built the "**main()**" function. Following the greeting, it begins a loop where it will take user commands and perform the associated command function based on the user's input. Once the user enters the "**quit**" command, the loop does not end. To deal with the problems in the commands and file name, we applied the "**try and except**" statement. The "**execvp**" function is used to carry out system commands that are not explicitly defined. Using the "**os.execvp**" system call, the system forks a child process and performs the command. The "**os.waitpid**" function is used to wait for the child process to finish. The console provides an interactive prompt where the user can enter commands. The prompt is displayed using the '**input**' function.

2. The project starts by defining a number of functions to deal with various commands. These functions use the 'os' module to communicate with the operating system and execute system calls. Some of the commands are as follows:

- **map(pwd):** Prints the current working directory.
- **inventory(ls):** Lists the files and directories in the current directory.
- **today(date):** Prints the current date and time.
- **go(cd):** It is a bidirectional command which allows to change the current directory to the specified directory(go) and to the previous directory(go ..).
- **back(cd ..):** Navigates to the previous directory which works same as (go ..).
- **make(mkdir):** Creates a new directory.
- **feel(touch):** Appends text to a file.
- **vanish(rm):** Removes a file.
- **ink(echo):** Prints the provided message.
- **sweep(cls):** Clears the terminal screen.
- **record(history):** Prints the history of previous recorded commands.

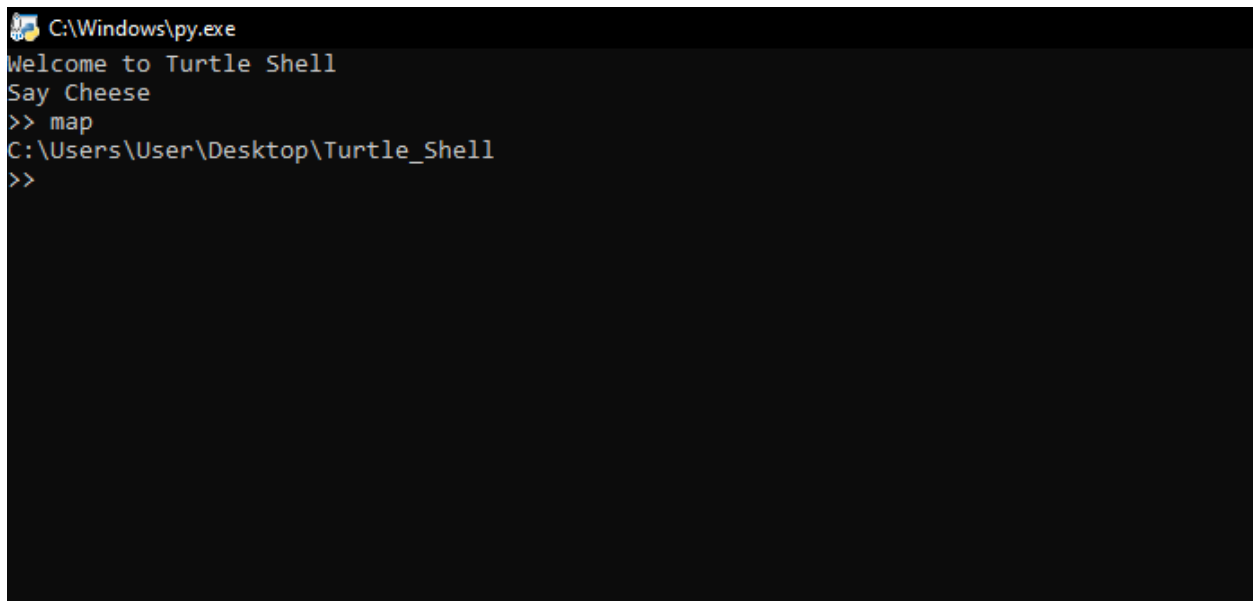
Results

1. Interface



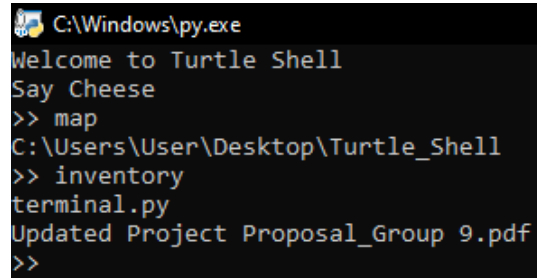
```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese
>>
```

2. map



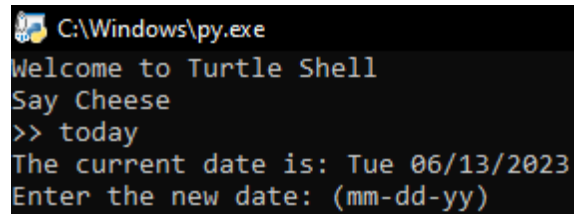
```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese
>> map
C:\Users\User\Desktop\Turtle_Shell
>>
```

3. inventory



```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese
>> map
C:\Users\User\Desktop\Turtle_Shell
>> inventory
terminal.py
Updated Project Proposal_Group 9.pdf
>>
```

4. today



```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese
>> today
The current date is: Tue 06/13/2023
Enter the new date: (mm-dd-yy)
```

5. go & go ..

```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese
>> go F:
Current directory changed to 'F:\'
>> map
F:\
>>
```

```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese
>> map
C:\Users\User\Desktop\Turtle_Shell
>> go ..
Current directory changed to 'C:\Users\User\Desktop'
>> go Turtle_Shell
Current directory changed to 'C:\Users\User\Desktop\Turtle_Shell'
>>
```

6. back (go ..)

```
C:\Windows\py.exe
>> map
C:\Users\User\Desktop\Turtle_Shell
>> back
Navigated to the previous directory: 'C:\Users\User\Desktop'
>> map
C:\Users\User\Desktop
>>
```

7.make

```
C:\Windows\py.exe
Welcome to Turtle Shell
Say Cheese

>> make Bangladesh
Directory 'Bangladesh' created successfully.

>> inventory
Bangladesh
terminal.py
Updated Project Proposal_Group 9.pdf
>>
```

8.feel

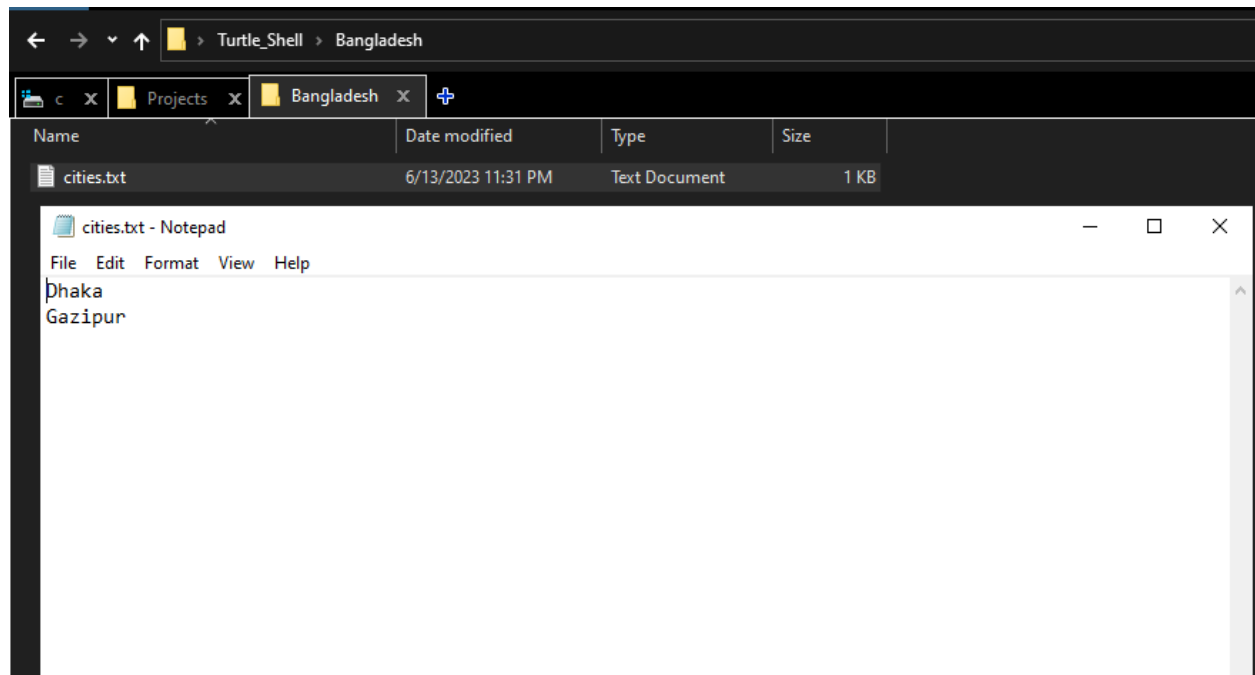
```
C:\Windows\py.exe

>> go Bangladesh
Current directory changed to 'C:\Users\User\Desktop\Turtle_Shell\Bangladesh'

>> feel cities.txt
Enter text to append to the file: Dhaka
Text appended to the file 'cities.txt' successfully.

>> feel cities.txt
Enter text to append to the file: Gazipur
Text appended to the file 'cities.txt' successfully.

>>
```



9. vanish

```
C:\Windows\py.exe

>> feel cities.txt
Enter text to append to the file: hello
Text appended to the file 'cities.txt' successfully.

>> inventory
cities.txt
Project_Proposal_Group_9.pdf
Project_Report_2013095042_2014109042.pdf
terminal.py

>> vanish cities.txt
File 'cities.txt' removed successfully.

>> inventory
Project_Proposal_Group_9.pdf
Project_Report_2013095042_2014109042.pdf
terminal.py

>>
```

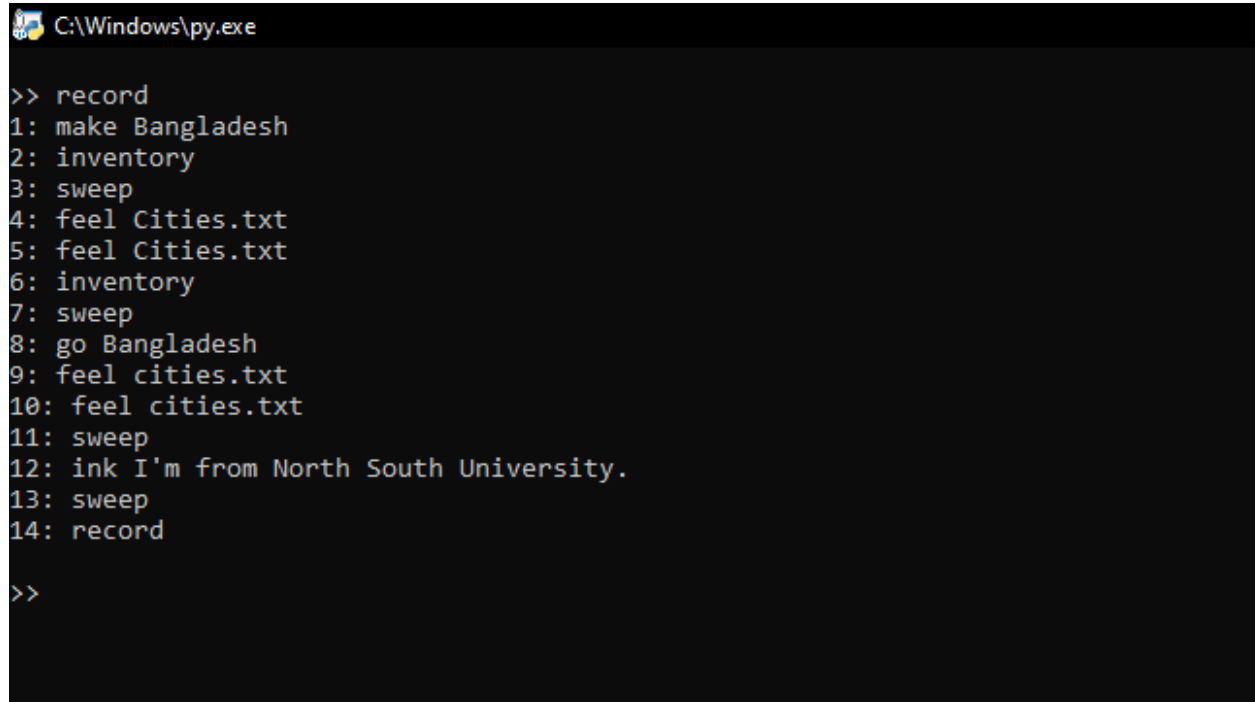
9. ink

```
C:\Windows\py.exe

>> ink I'm from North South University.
I'm from North South University.

>>
```

10. record



```
C:\Windows\py.exe

>> record
1: make Bangladesh
2: inventory
3: sweep
4: feel Cities.txt
5: feel Cities.txt
6: inventory
7: sweep
8: go Bangladesh
9: feel cities.txt
10: feel cities.txt
11: sweep
12: ink I'm from North South University.
13: sweep
14: record

>>
```

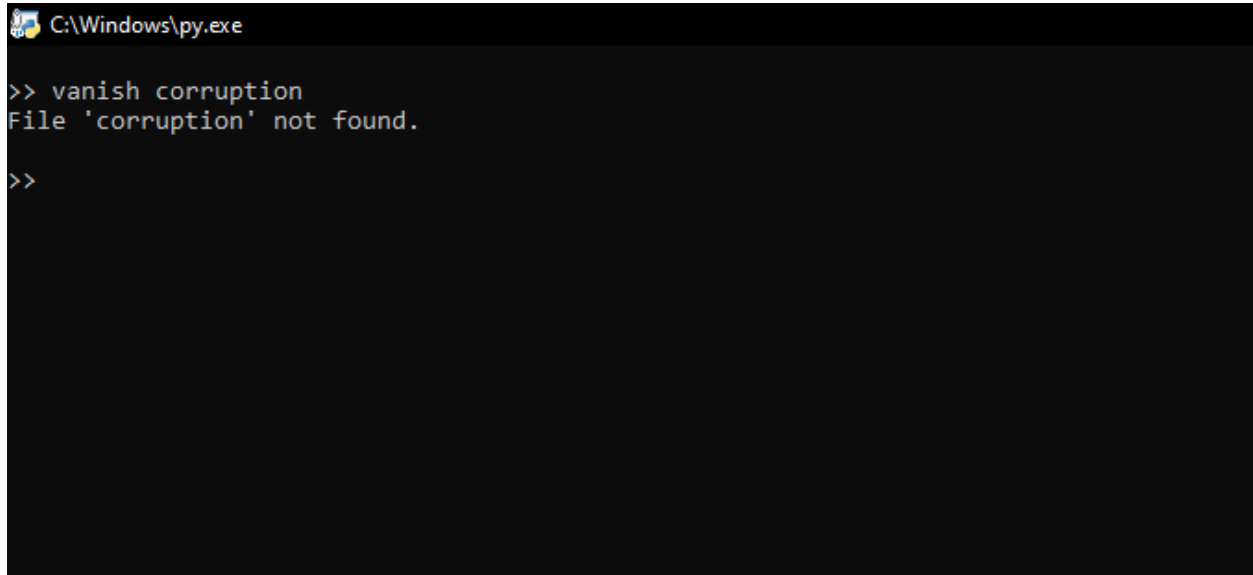
11. sweep



```
C:\Windows\py.exe

>>
```

12. Exception Handling

A screenshot of a Windows command prompt window titled "C:\Windows\py.exe". The prompt shows the command ">> vanish corruption" being entered. The response from the program is "File 'corruption' not found." followed by another prompt ">>".

```
C:\Windows\py.exe
>> vanish corruption
File 'corruption' not found.
>>
```

Summary

The Turtle Shell project uses numerous system calls and functions from the 'os' module to construct a straightforward command line shell. This shell offers a collection of fundamental commands that can be used to handle files and directories, run system commands, browse the file system, and keep track of command history. This project provides a simple command line interface for users to execute different system calls, file operations, and commands. It can be used as an introduction to creating complex shell programs or as a learning tool for learning the fundamentals of shell programming.

.....