221070055

Raj Sapale

EXPERIMENT NO. 01

**Aim:**

Create a shell for the Ubuntu operating system that mimics the behavior of the Bash shell.

**Theory:**

A shell is a command-line interface that allows users to interact with an operating system. The Bash shell is one of the most commonly used shells in Unix-like operating systems. The aim of this lab experiment is to create a simple shell using Python's tkinter library that emulates the basic functionality of the Bash shell on an Ubuntu operating system.

**Implementation:**

The shell implementation utilizes the tkinter library in Python to create a graphical user interface (GUI) that consists of a text area for both command input and output. The core functionalities include executing commands, changing directories, displaying command history, and handling basic error cases.

Command Execution:

* The subprocess module is employed to execute commands entered by the user. The output (both standard and error) is captured and displayed in the text area.

Directory Change:

* The 'cd' command allows the user to change the working directory. The 'try-except' block is used to handle cases where the specified directory is not found.

Command History:

* The 'history' command displays a numbered list of previously executed commands. The history is stored in a list, and the GUI is updated accordingly.

Clearing Output:

* The 'clear' command clears the text area, providing a clean slate for new commands and outputs.

Exit:

* The 'exit' command closes the GUI, effectively ending the shell program.

**Code:**

import subprocess

import os

import tkinter as tk

from tkinter import scrolledtext

class UltimateShellGUI:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Ultimate Shell GUI")

        self.current\_directory\_label = tk.Label(root, text=f"{os.getcwd()}", anchor='w')

        self.current\_directory\_label.pack(fill='x')

        self.entry = tk.Entry(root, width=80)

        self.entry.pack(pady=10)

        self.output\_text = scrolledtext.ScrolledText(root, wrap=tk.WORD, width=80, height=30)

        self.output\_text.pack(padx=10, pady=10)

        self.command\_history = []

        self.entry.bind('<Return>', lambda event: self.execute\_command())

    def execute\_command(self):

        command = self.entry.get()

        self.entry.delete(0, tk.END)

        if command.lower() == 'clear':

            self.clear\_output()

        elif command.lower() == 'history':

            self.print\_history()

        elif command.startswith('cd '):

            self.change\_directory(command[3:])

        elif command.lower() == 'exit':

            self.root.destroy()

        else:

            self.run\_system\_command(command)

    def run\_system\_command(self, command):

        try:

            result = subprocess.run(command, shell=True, text=True, capture\_output=True)

            if result.returncode == 0:

                self.display\_output(f">> {command}\n{result.stdout}\n")

            else:

                self.display\_output(f">> {command}\nError: {result.stderr}\n")

        except Exception as e:

            self.display\_output(f">> {command}\nException: {e}\n")

        self.display\_output('\n')

    def change\_directory(self, new\_directory):

        try:

            os.chdir(new\_directory)

            self.current\_directory\_label.config(text=f"Current Directory: {os.getcwd()}")

            self.display\_output(f"Changed directory to: {os.getcwd()}\n")

        except FileNotFoundError:

            self.display\_output(f"Directory not found: {new\_directory}\n")

    def print\_history(self):

        history\_text = "Command history:\n"

        for idx, cmd in enumerate(self.command\_history, start=1):

            history\_text += f"{idx}. {cmd}\n"

        self.display\_output(history\_text)

    def clear\_output(self):

        self.output\_text.delete(1.0, tk.END)

    def display\_output(self, output):

        self.output\_text.insert(tk.END, output)

        self.output\_text.see(tk.END)

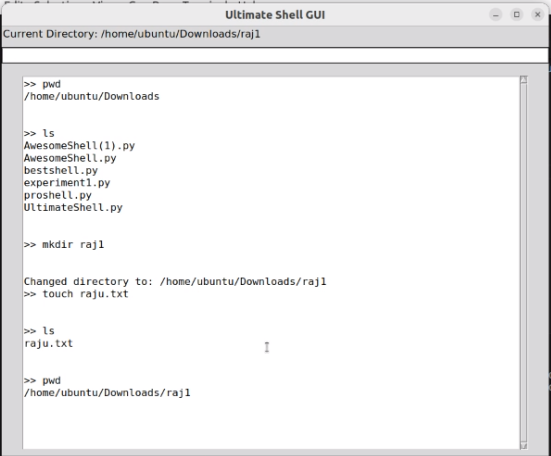
if \_\_name\_\_ == "\_\_main\_\_":

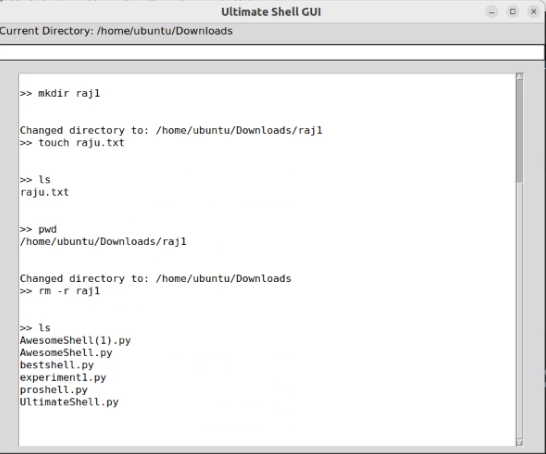
    root = tk.Tk()

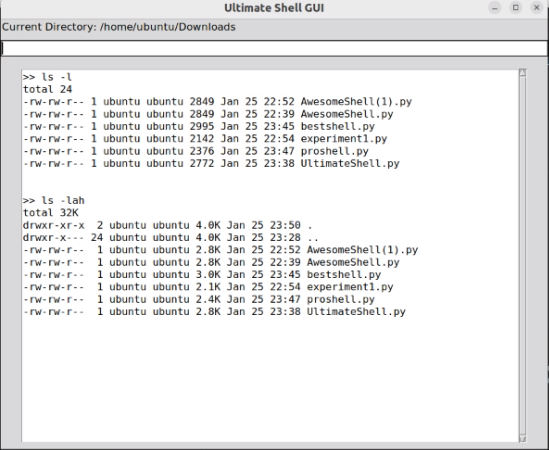
    app = UltimateShellGUI(root)

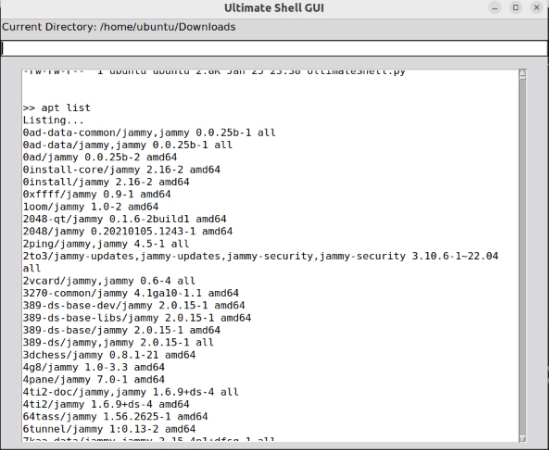
    root.mainloop()

**Output:**









**Conclusion:**

We got a hands-on experience in combining programming skills with system interaction by performing this experiment where we tried to create a tkinter based gui which emulates the behaviour of a basic Bash shell on an Ubuntu Operating System which allows user perform some basic tasks such as execute commands, change directories, view command history, and clear the output.