BASIC SHELL IMPLEMENTATION

Step-by-Step Guide

OBJECTIVE: THE GOAL IS TO IMPLEMENT A COMMAND-LINE INTERFACE (CLI) THAT ALLOWS USERS TO EXECUTE BUILT-IN AND EXTERNAL COMMANDS, HANDLE BACKGROUND PROCESSES, AND PROVIDE OUTPUT REDIRECTION. THIS SHELL WILL MIMIC THE BASIC FUNCTIONALITY OF A UNIX SHELL.

Key Concepts:

- Process Creation: fork(), exec(), and wait()

- Input/Output Redirection: <, >

- Signal Handling: SIGINT, SIGCHLD

- Piping Commands: |

DETAILED STEPS:

Features:

1. Parse user input.
2. Implement built-in commands like cd, exit.
3. Use fork() and exec() to execute other programs.
4. Add support for background processes (using &).
5. Implement I/O redirection for commands.
6. Implement piping between commands (command1 | command2).

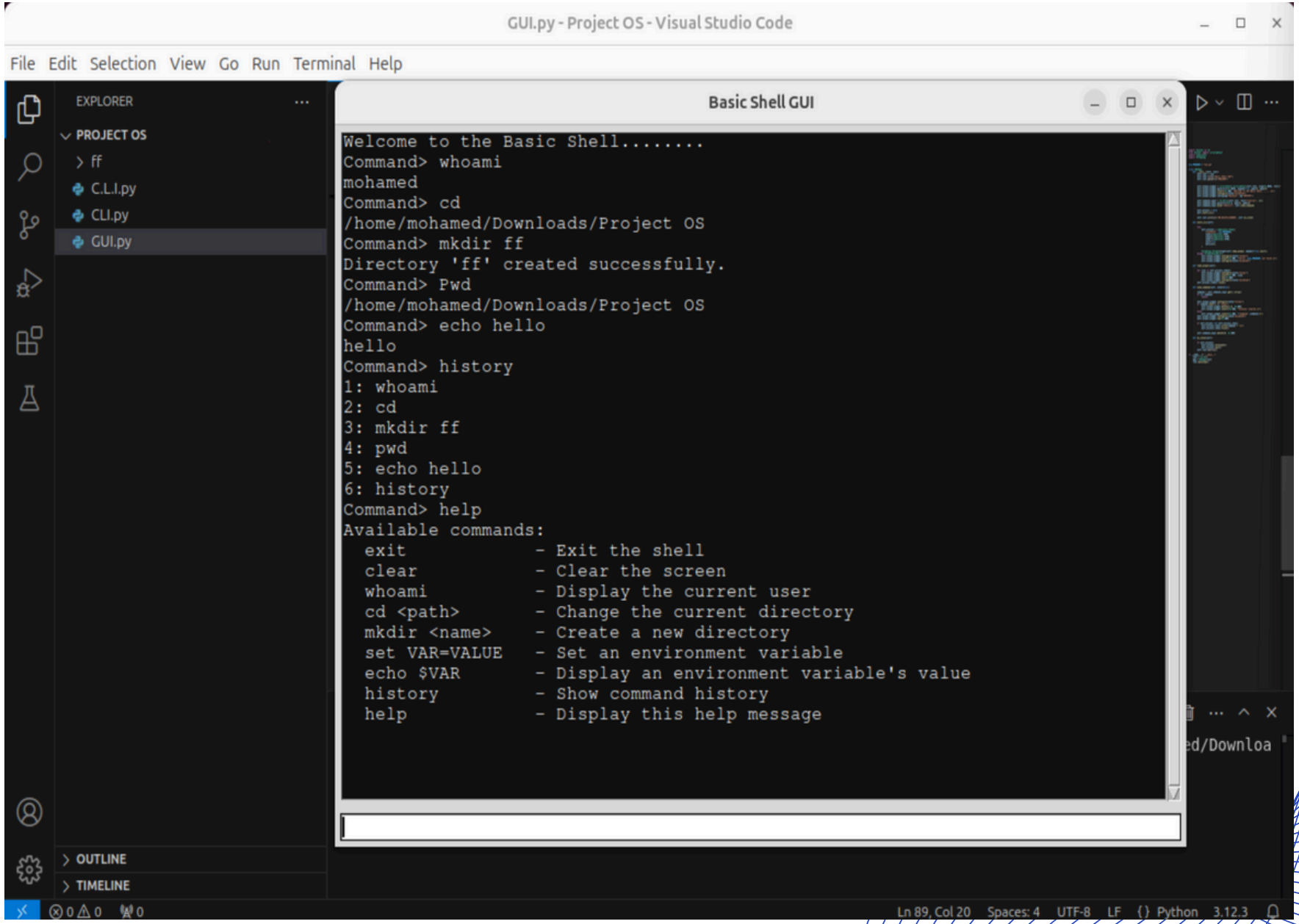
1. IMPLEMENT BUILT-IN COMMANDS (CD, EXIT,...)

The shell should support built-in commands like cd (change directory) and exit. Here's an example

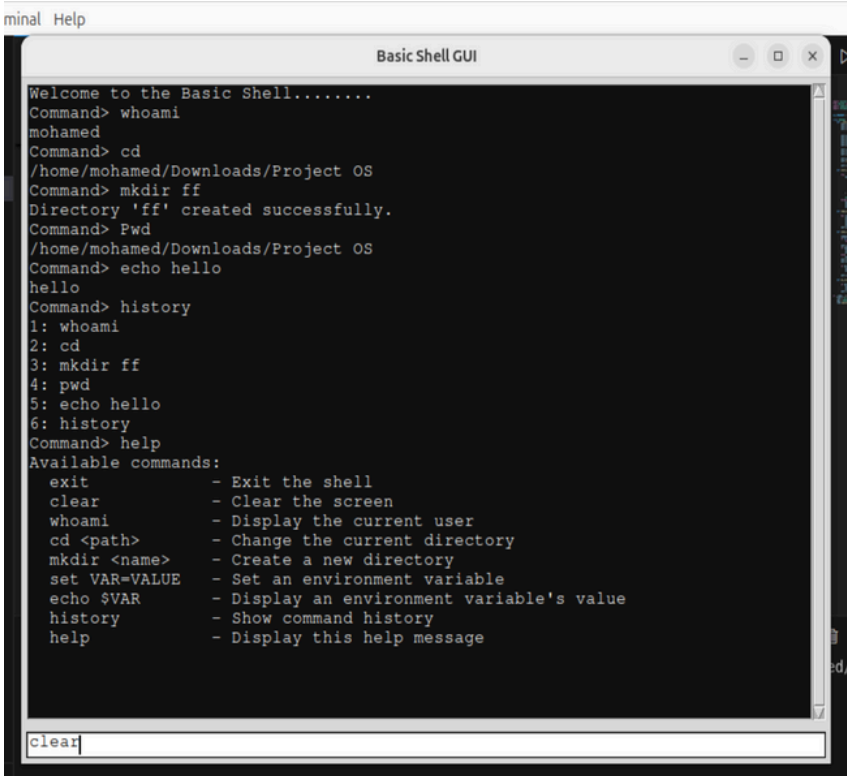
for each:

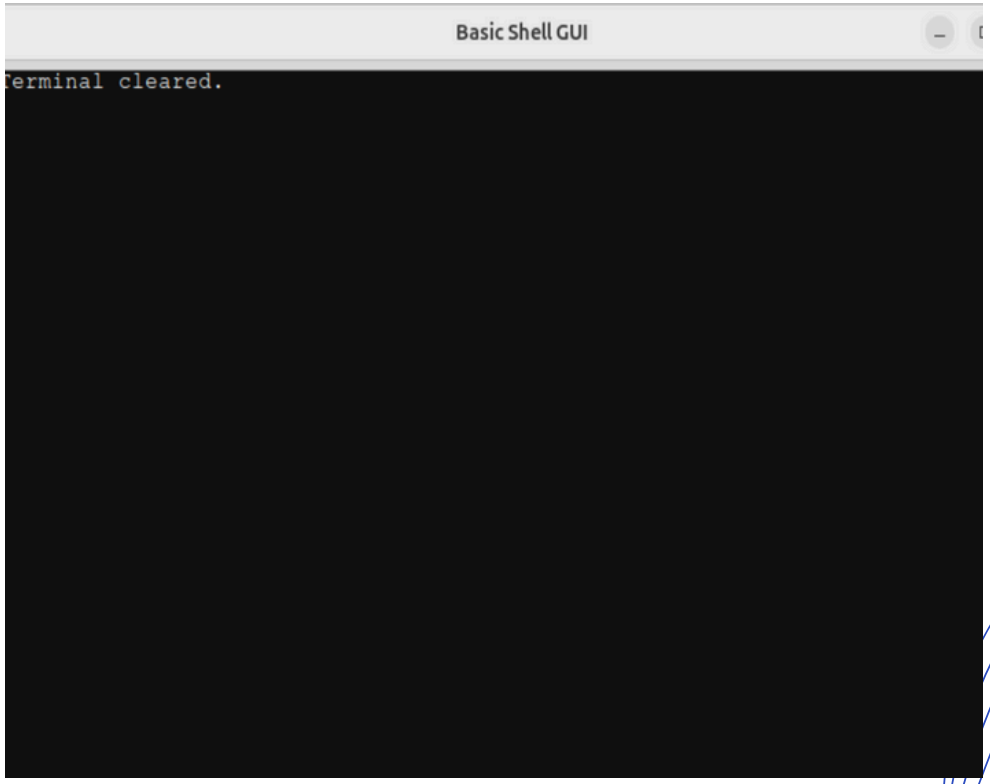
- **exit: Exits the shell.**
- **clear: Clears the screen.**
- **whoami: Displays the current user.**
- **cd: Changes the current directory.**
- **mkdir: Creates a new directory.**
- **Pwd: Prints the absolute path of the current working directory.**
- **echo: Displays the value of an environment variable or an argument.**
- **history: Displays a list of past commands.**
- **help: Displays a list of available commands.**

1. IMPLEMENT BUILT-IN COMMANDS (CD, EXIT,...)



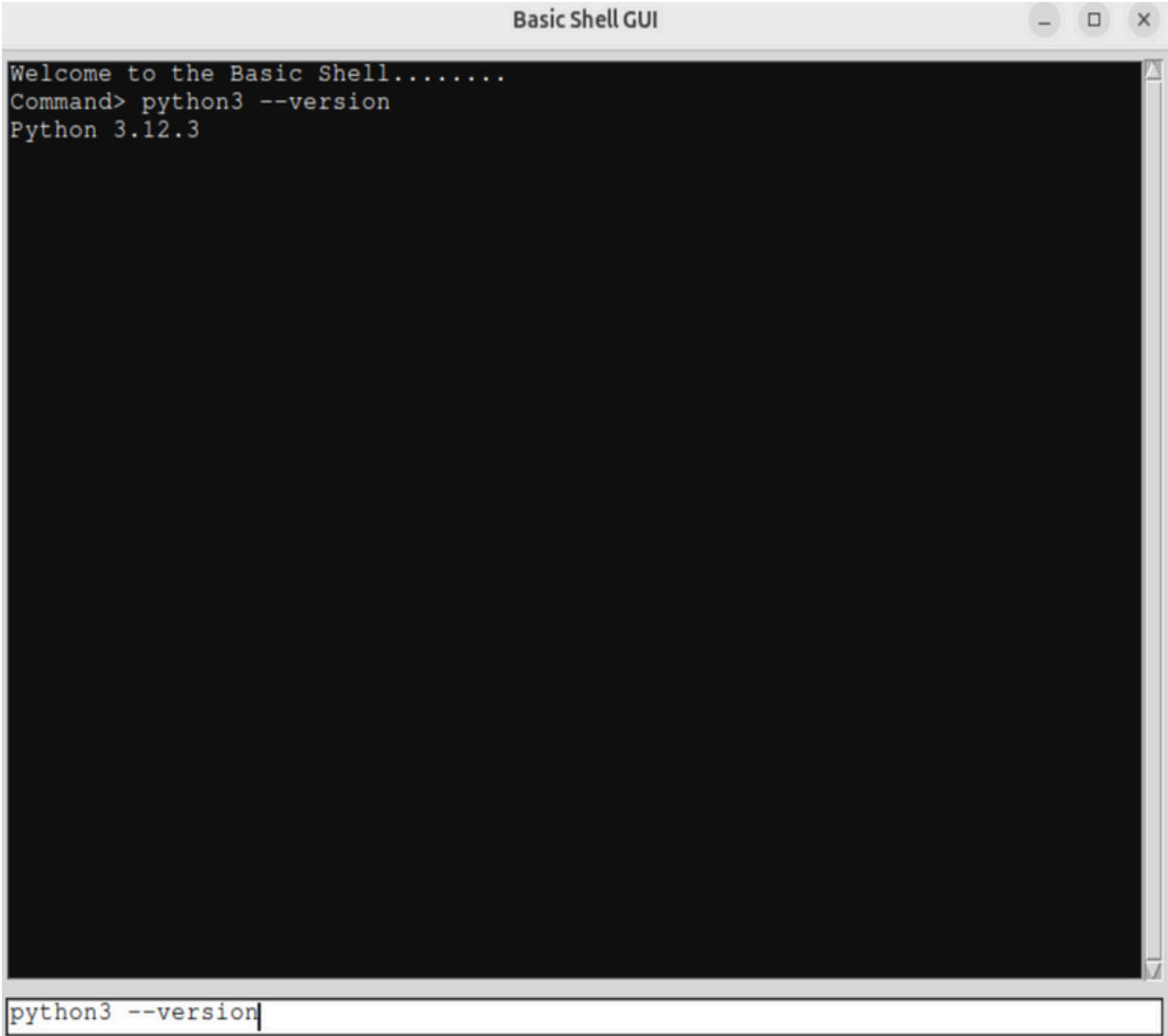
1. IMPLEMENT BUILT-IN COMMANDS (CD, EXIT,...)





2. PROCESS CREATION USING FORK() AND EXEC()

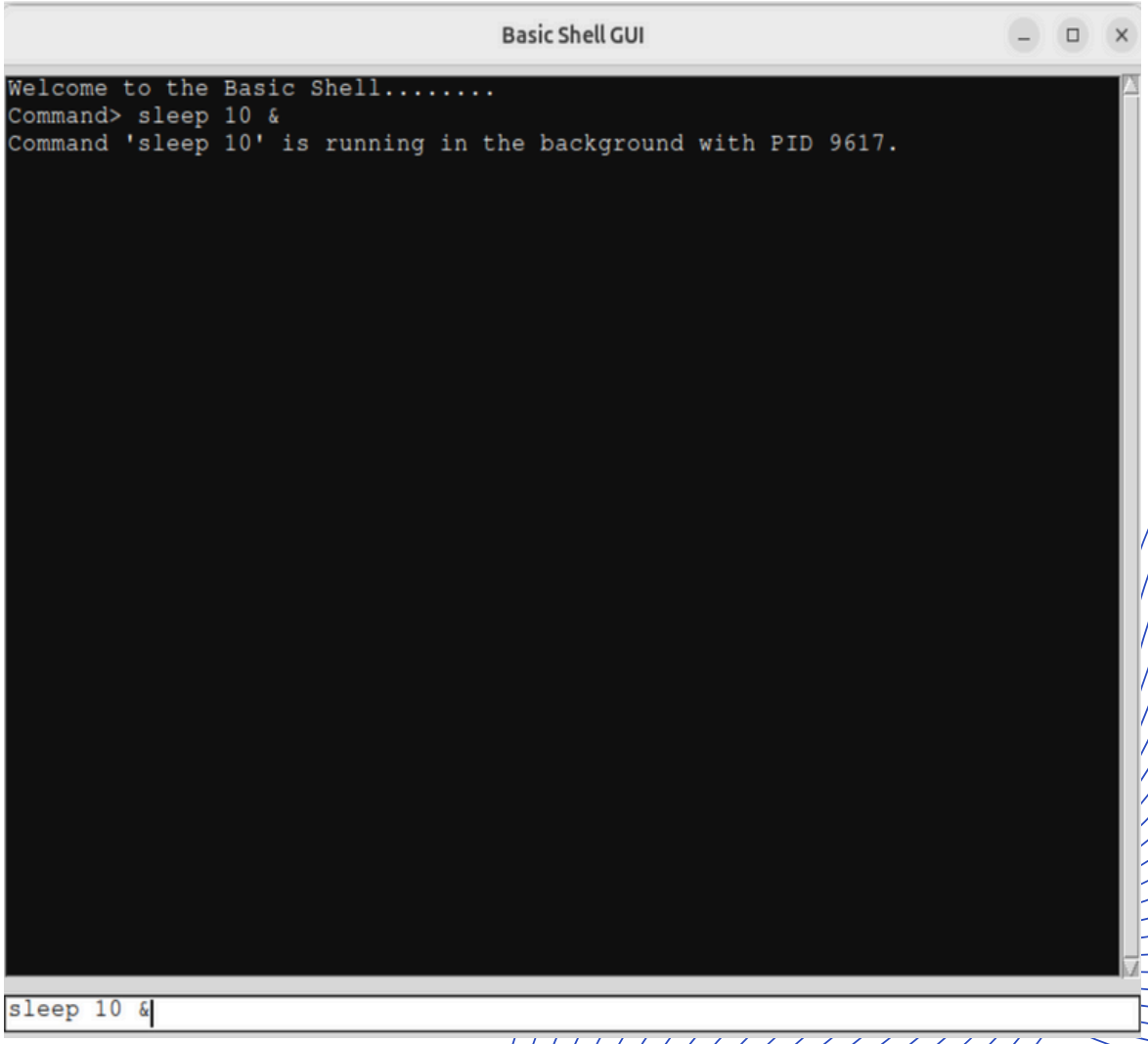
fork() and exec() system calls to execute a command like python3 --version



3. BACKGROUND PROCESSES USING &

Commands followed by '&' should be executed in the background.

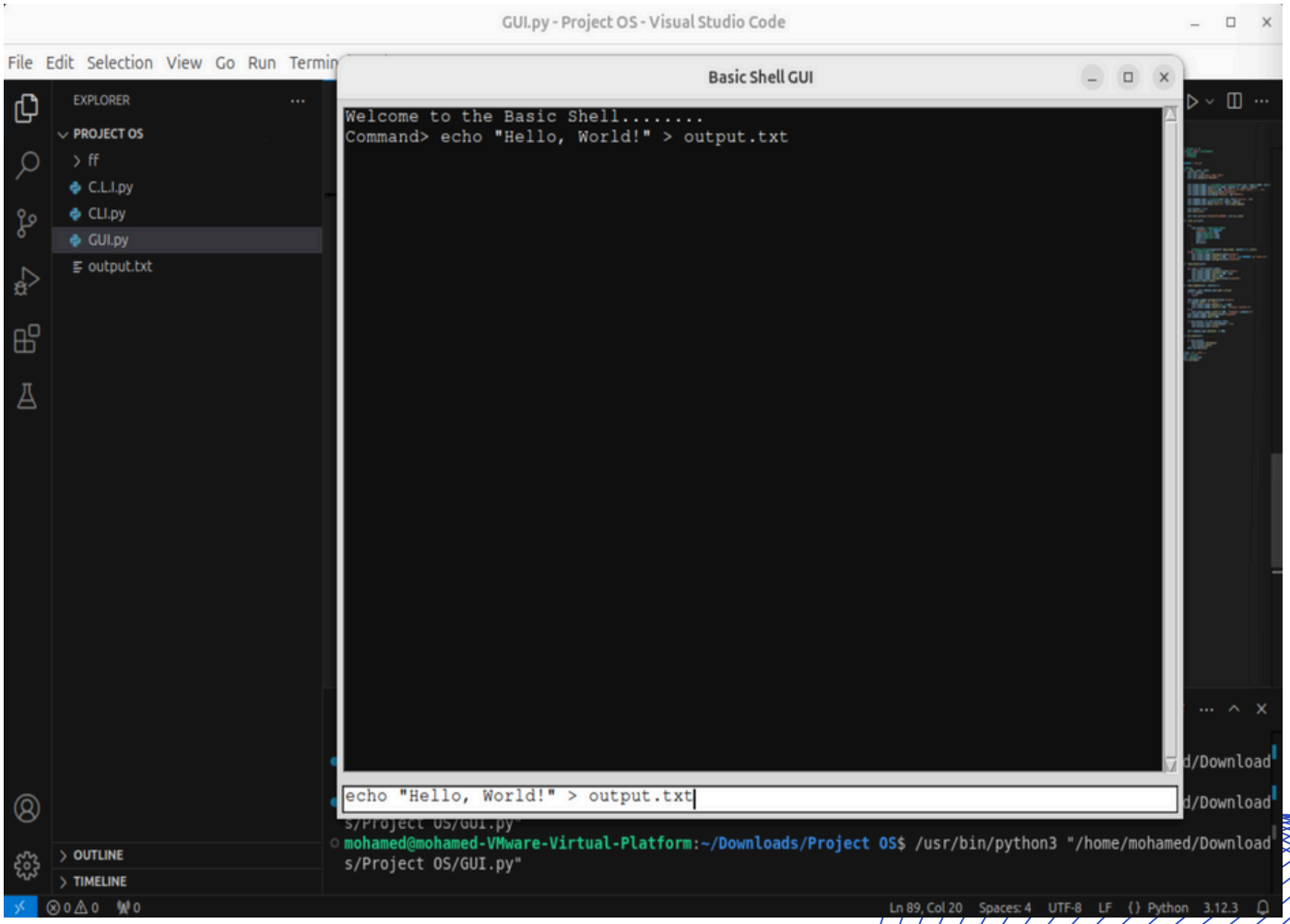
Example: sleep 10 &



4. INPUT/OUTPUT REDIRECTION (<, >)

Input redirection ('<') reads input from a file, and output redirection ('>') sends output to a file.

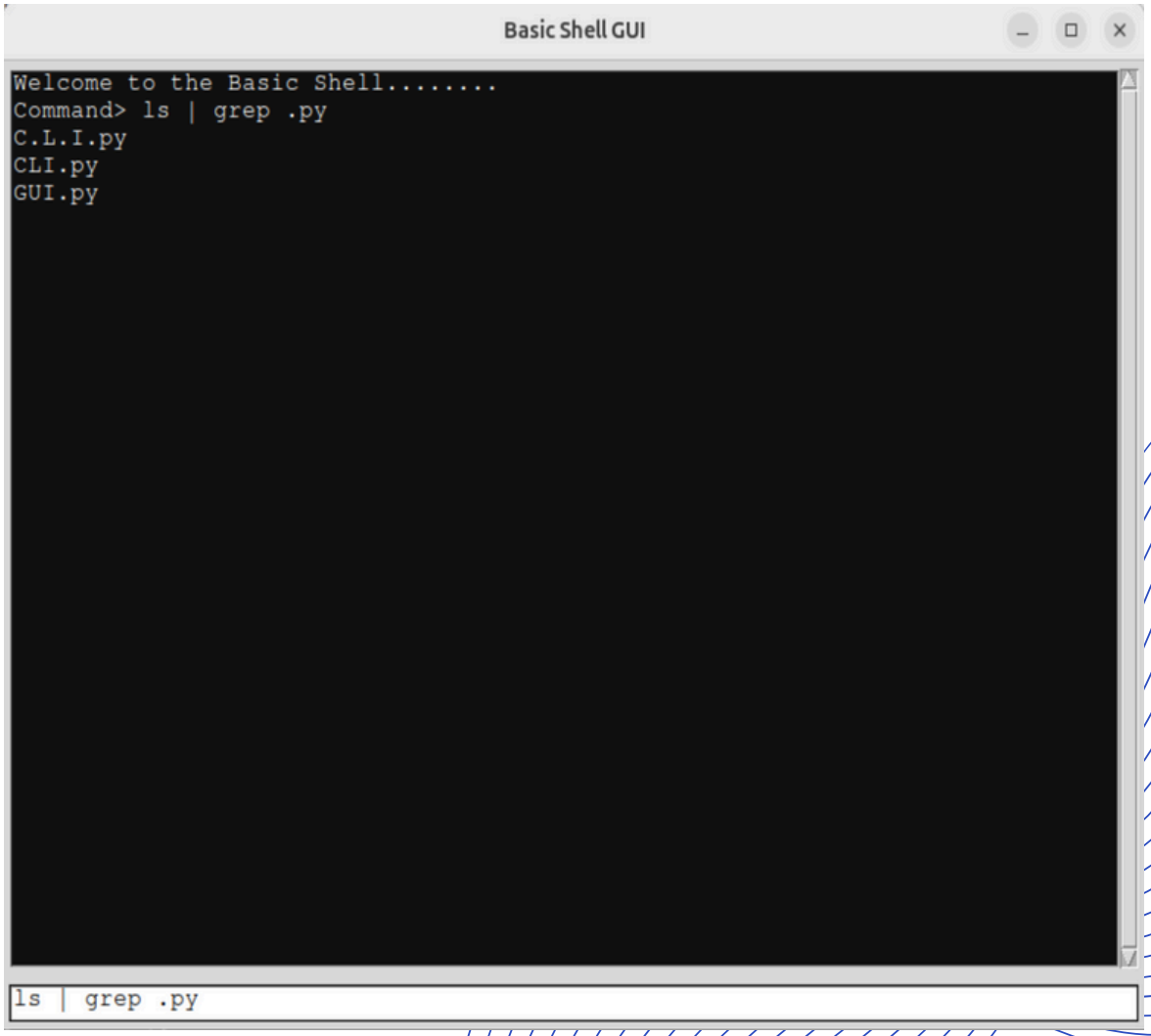
Example: echo "Hello, World!" > output.txt



5. PIPING COMMANDS (|)

The pipe ('|') allows the output of one command to be passed as the input to another.

Example: ls | grep .py



6. SIGNAL HANDLING:

Gracefully handles interruptions (e.g., Ctrl+C) with a custom message.

POSSIBLE EXTENSIONS:

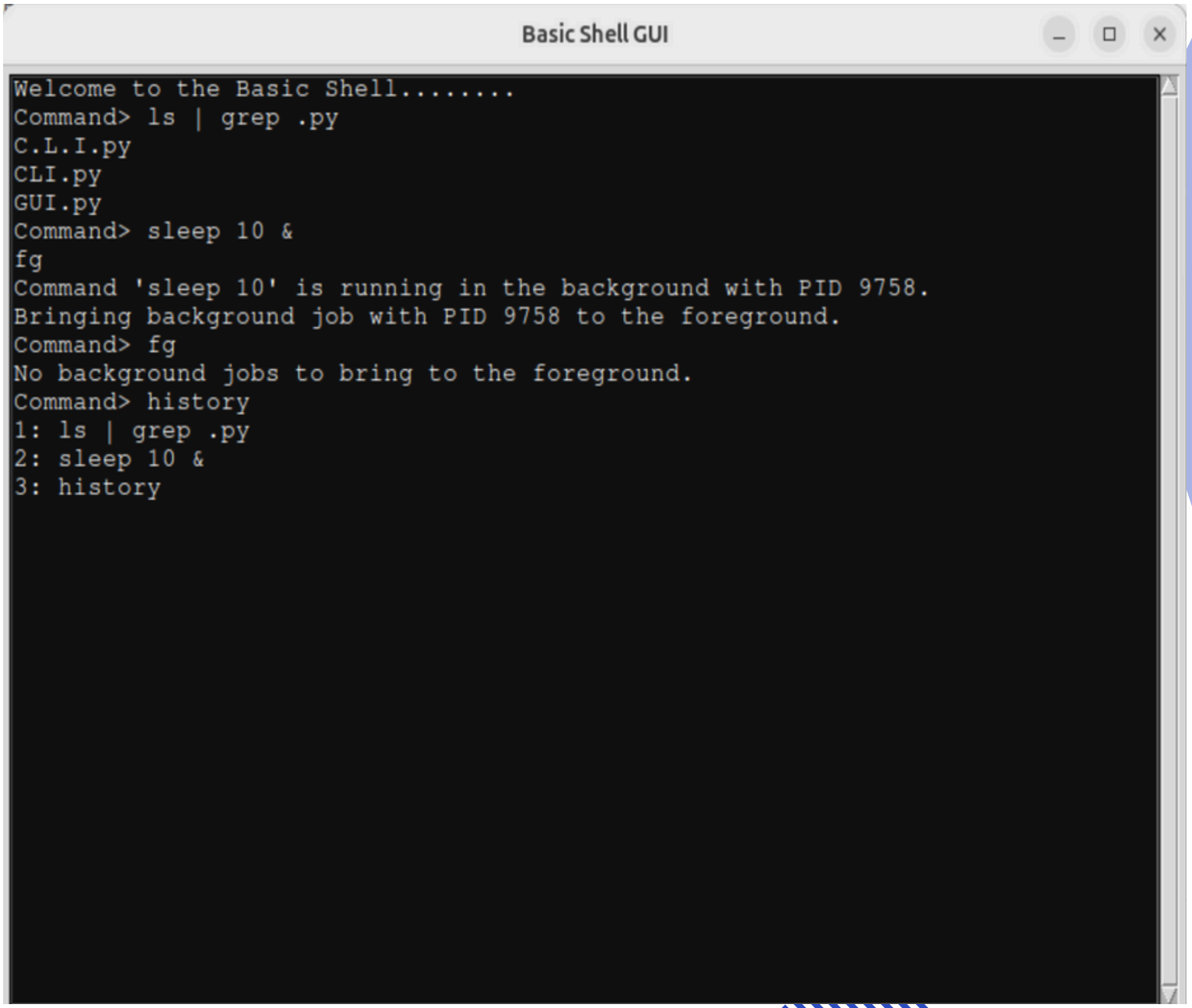
7

- Job control (allowing users to bring background jobs to the foreground).

- Support for environment variables (`$PATH`).

- Command history with navigation.

POSSIBLE EXTENSIONS:



Basic Shell GUI

```
                    Welcome to the Basic Shell
Command> ls -l
total 20
-rw-r--r-- 1 mohamed mohamed 6638 Dec 20 13:57 C.L.I.py
-rw-r--r-- 1 mohamed mohamed 6616 Dec 20 14:59 CLI.py
-rw-r--r-- 1 mohamed mohamed 2757 Dec 20 15:10 GUI.py
Command> pwd
/home/mohamed/Downloads/Project OS
Command> python3 --version
Python 3.12.3
Command> ls | grep .py
C.L.I.py
CLI.py
GUI.py
Command> sleep 10 &
fg
Command 'sleep 10' is running in the background with PID 17694.
Bringing background job with PID 17694 to the foreground.
Command> echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
local/games:/snap/bin:/snap/bin
Command> echo "Hello, World!" > output.txt
Command> echo "Hello, World!" >> output.txt
Command> cat output.txt
"Hello, World!"
"Hello, World!"
```

Basic Shell GUI

```
Welcome to the Basic Shell........
Command> exit
Exiting the shell.
```

exit

THANK YOU!