



The background features several decorative elements: a large solid blue circle in the top-left corner, a smaller circle with blue diagonal stripes in the top-right corner, a solid blue circle on the right edge, and a circle with blue diagonal stripes in the bottom-left corner. At the bottom, there are overlapping wavy shapes in light blue and medium blue.

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:

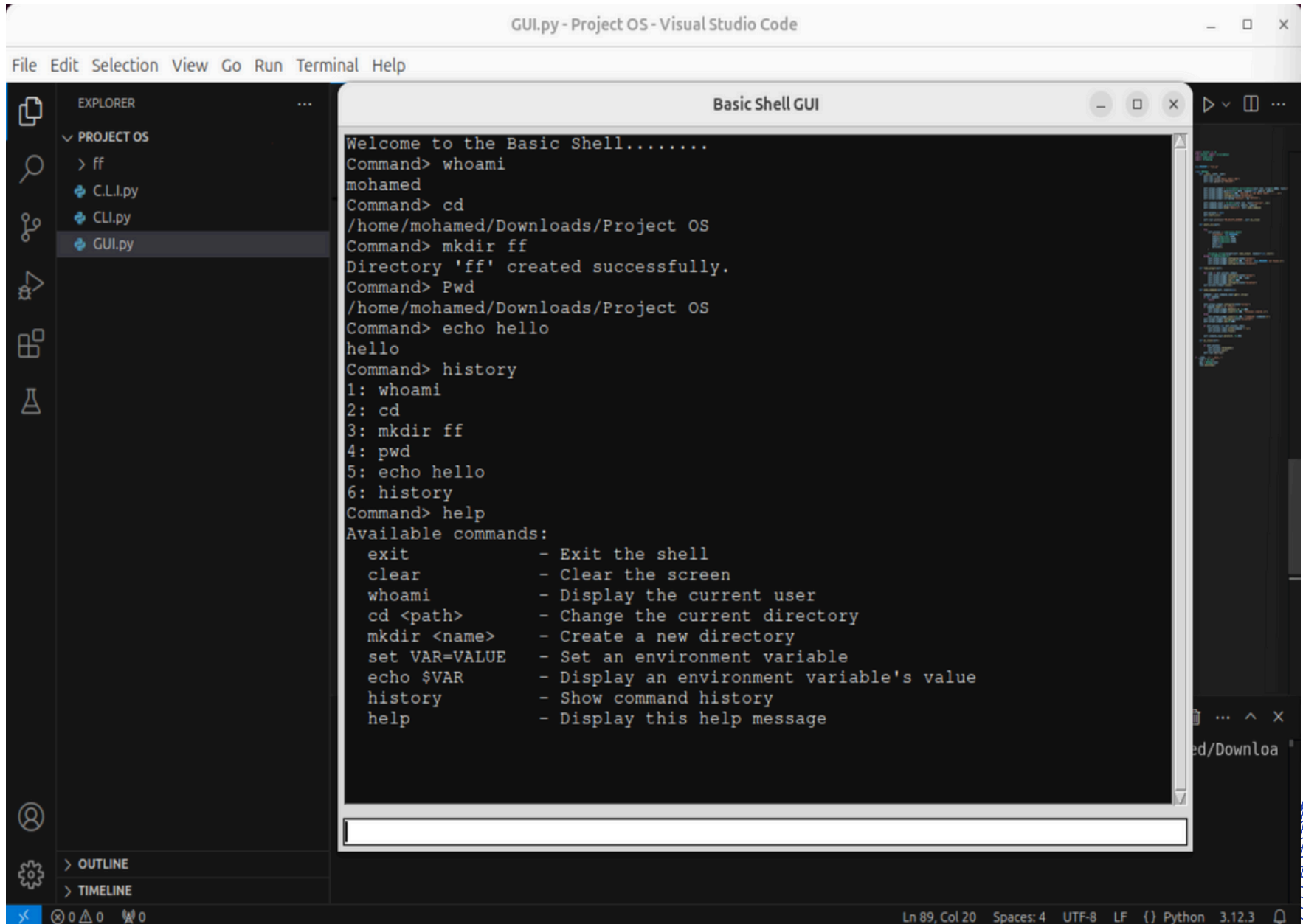
- 1 Parse user input.
- 2 Implement built-in commands like `cd`, `exit`.
- 3 Use `fork()` and `exec()` to execute other programs.
- 4 Add support for background processes (using `&`).
- 5 Implement I/O redirection for commands.
- 6 Implement piping between commands (`command1 | command2`).

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,...)



GUI.py - Project OS - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

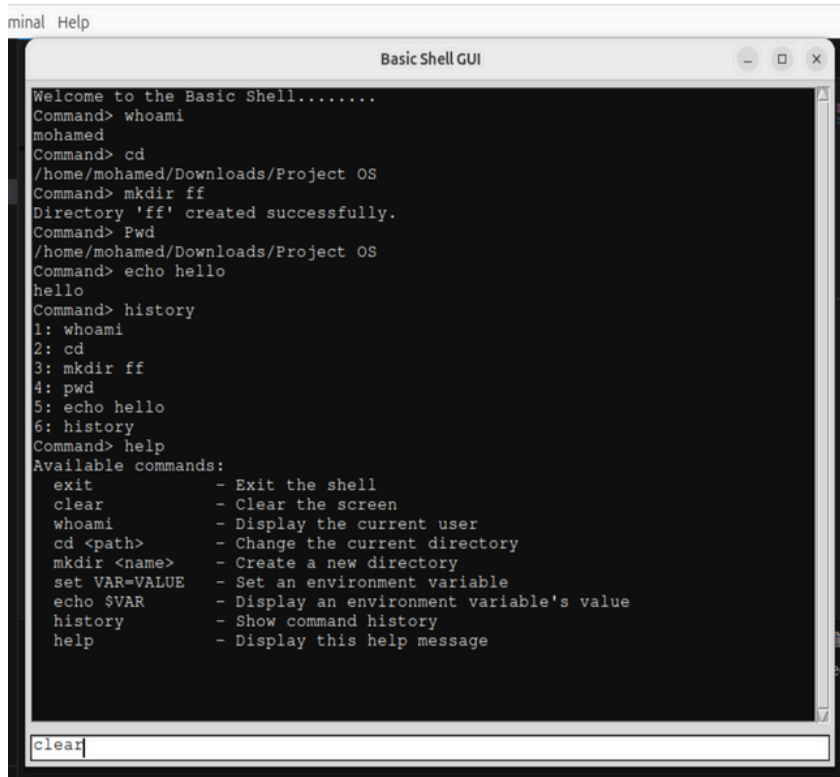
- PROJECT OS
 - ff
 - C.L.I.py
 - CLI.py
 - GUI.py

Basic Shell GUI

```
Welcome to the Basic Shell.....
Command> whoami
mohamed
Command> cd
/home/mohamed/Downloads/Project OS
Command> mkdir ff
Directory 'ff' created successfully.
Command> Pwd
/home/mohamed/Downloads/Project OS
Command> echo hello
hello
Command> history
1: whoami
2: cd
3: mkdir ff
4: pwd
5: echo hello
6: history
Command> help
Available commands:
  exit          - Exit the shell
  clear         - Clear the screen
  whoami        - Display the current user
  cd <path>     - Change the current directory
  mkdir <name>  - Create a new directory
  set VAR=VALUE - Set an environment variable
  echo $VAR     - Display an environment variable's value
  history       - Show command history
  help         - Display this help message
```

Ln 89, Col 20 Spaces: 4 UTF-8 LF {} Python 3.12.3

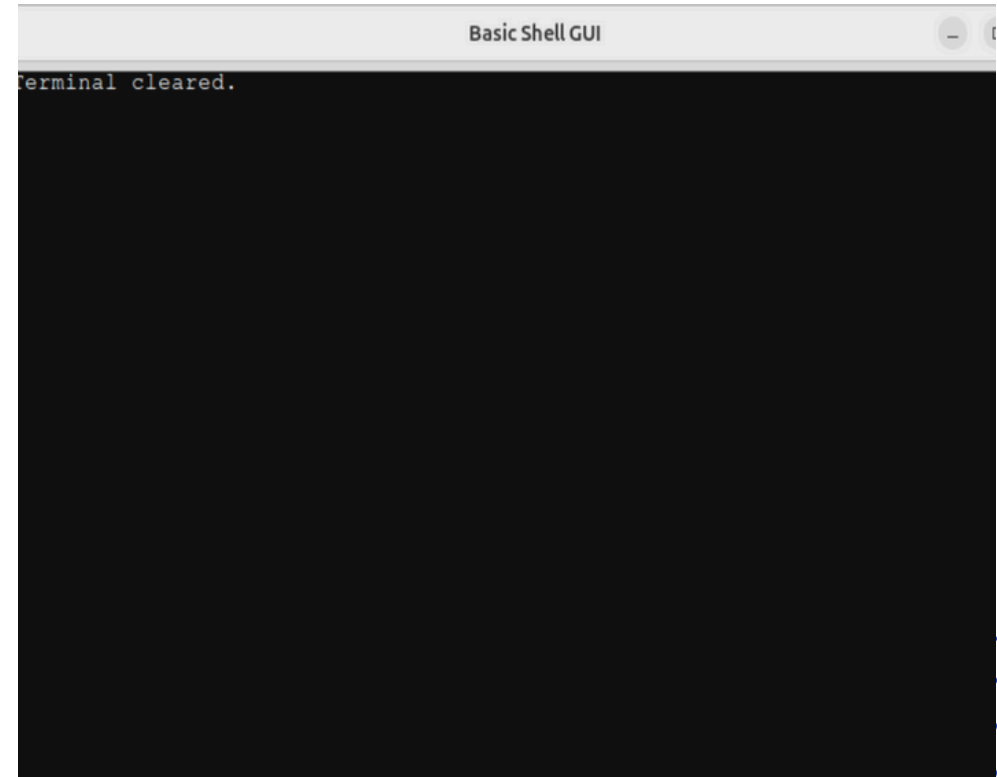
1. IMPLEMENT BUILT-IN COMMANDS (CD, EXIT,...)



The screenshot shows a window titled "Basic Shell GUI" with a menu bar containing "minal" and "Help". The terminal displays the following text:

```
Welcome to the Basic Shell.....
Command> whoami
mohamed
Command> cd
/home/mohamed/Downloads/Project OS
Command> mkdir ff
Directory 'ff' created successfully.
Command> Pwd
/home/mohamed/Downloads/Project OS
Command> echo hello
hello
Command> history
1: whoami
2: cd
3: mkdir ff
4: pwd
5: echo hello
6: history
Command> help
Available commands:
  exit      - Exit the shell
  clear     - Clear the screen
  whoami    - Display the current user
  cd <path> - Change the current directory
  mkdir <name> - Create a new directory
  set VAR=VALUE - Set an environment variable
  echo $VAR - Display an environment variable's value
  history   - Show command history
  help      - Display this help message
```

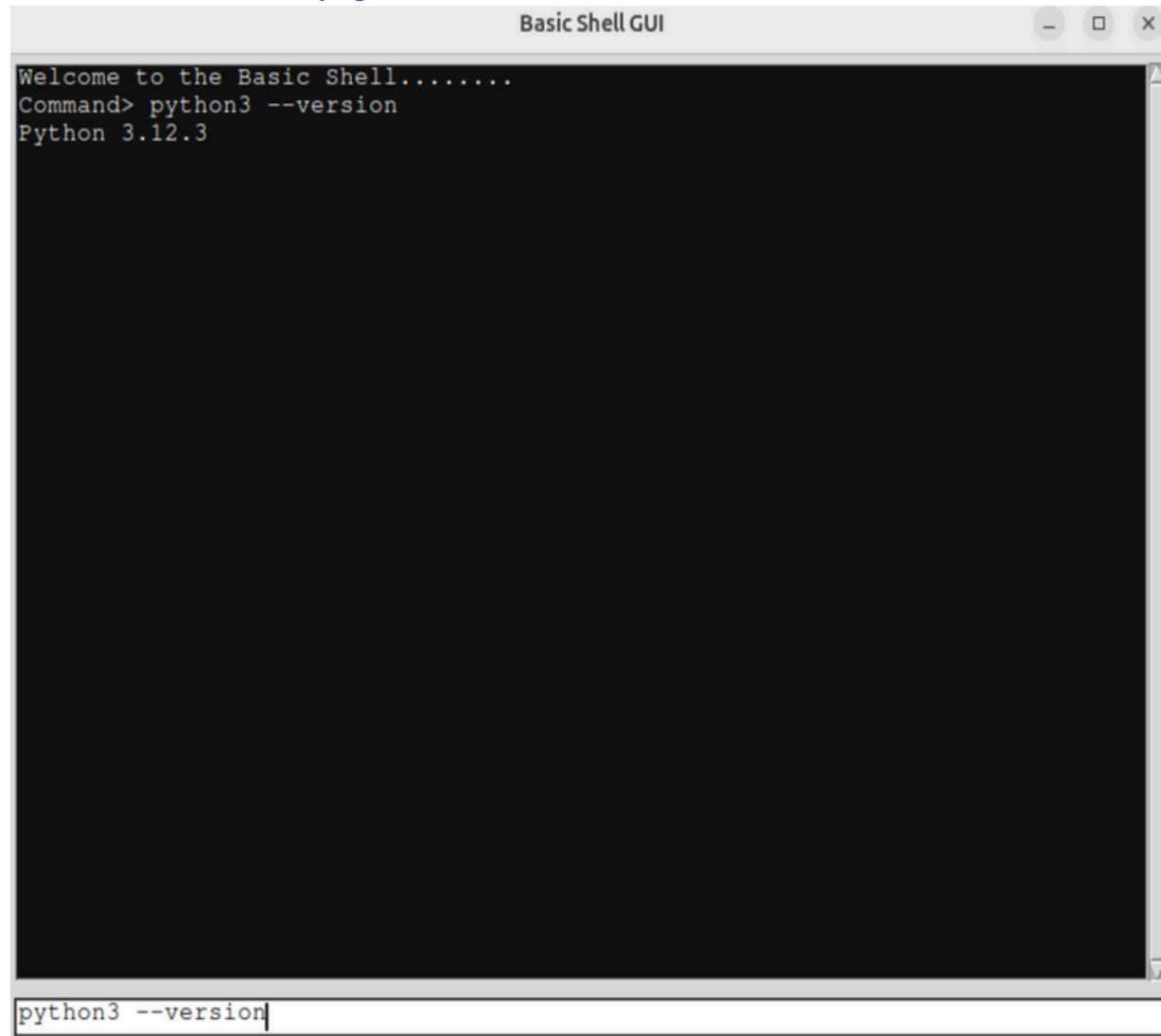
The input field at the bottom contains the text "clear".



The screenshot shows the same "Basic Shell GUI" window, but the terminal content has been cleared, displaying only the text "terminal cleared.".

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

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

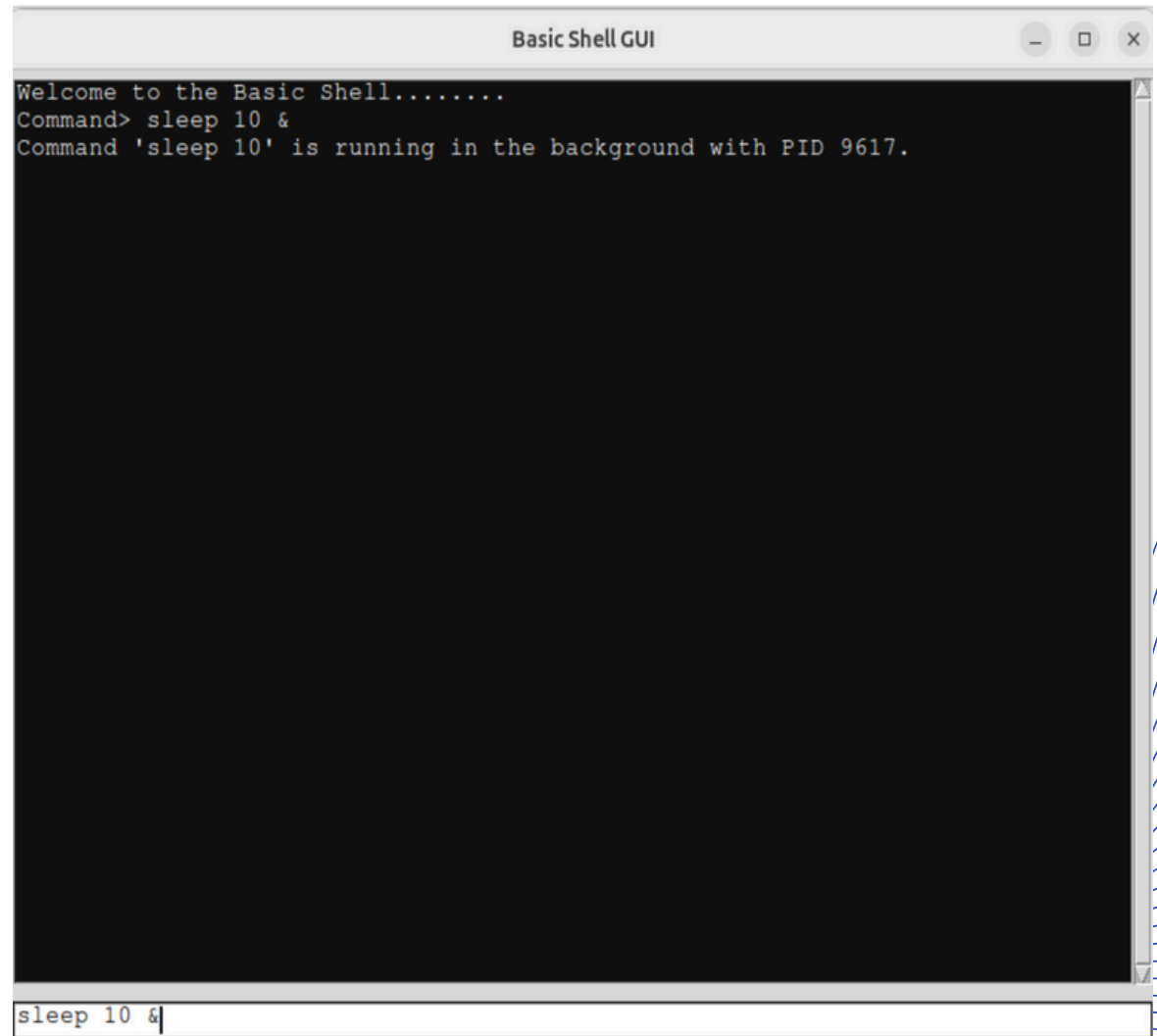
A screenshot of a window titled "Basic Shell GUI". The window has a black background with white text. The text inside the window reads: "Welcome to the Basic Shell.....", "Command> python3 --version", and "Python 3.12.3". At the bottom of the window, there is a white input field containing the text "python3 --version".

```
Basic Shell GUI
Welcome to the Basic Shell.....
Command> python3 --version
Python 3.12.3
python3 --version
```

3. BACKGROUND PROCESSES USING &

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

Example: sleep 10 &



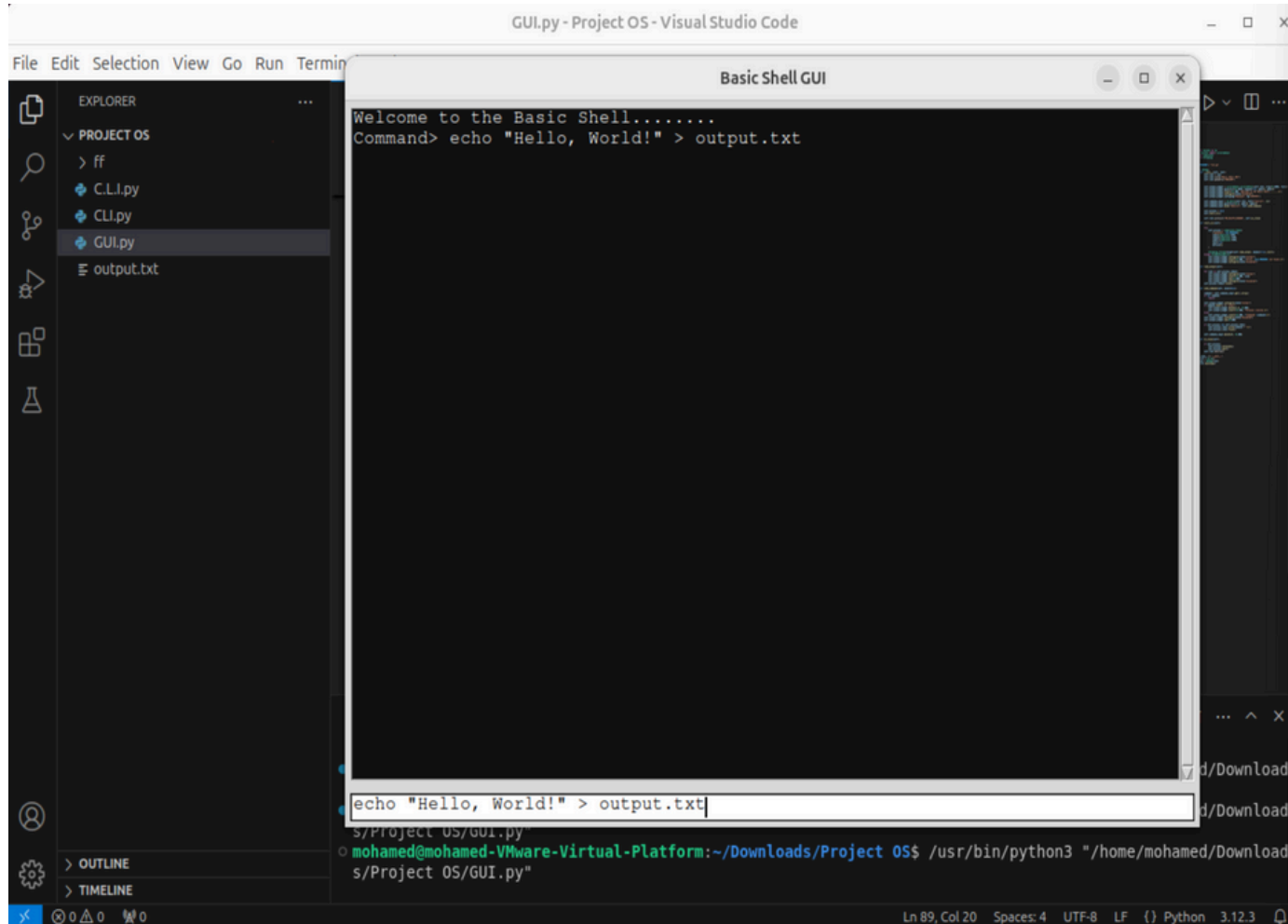
The screenshot shows a window titled "Basic Shell GUI" with a black terminal area. The text inside the terminal reads: "Welcome to the Basic Shell.....", "Command> sleep 10 &", and "Command 'sleep 10' is running in the background with PID 9617." At the bottom of the window, there is a white input field containing the text "sleep 10 &".

```
Basic Shell GUI
Welcome to the Basic Shell.....
Command> sleep 10 &
Command 'sleep 10' is running in the background with PID 9617.
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`

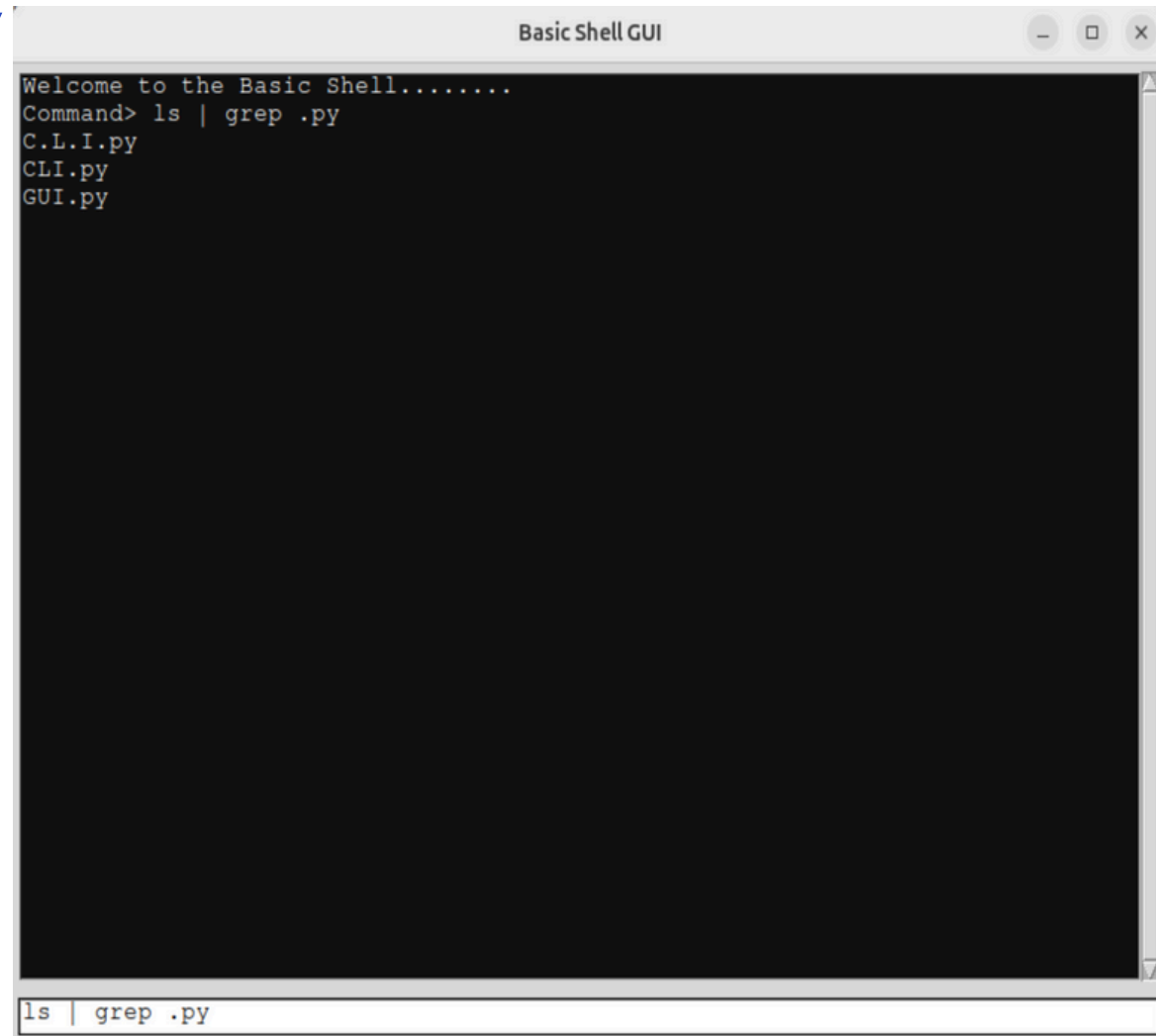


The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays the file structure of a project named 'PROJECT OS', which includes files 'C.LI.py', 'CLI.py', 'GUI.py', and 'output.txt'. The main editor area shows the 'GUI.py' file. A terminal window titled 'Basic Shell GUI' is open, displaying the command `echo "Hello, World!" > output.txt` and its output, `Hello, World!`. The terminal window also shows the prompt `Command>` and the file `output.txt`. The status bar at the bottom indicates the current file is `GUI.py` at line 89, column 20, using UTF-8 encoding with LF line endings.

5. PIPING COMMANDS (|)

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

Example: `ls | grep .py`



```
Basic Shell GUI
Welcome to the Basic Shell.....
Command> ls | grep .py
C.L.I.py
CLI.py
GUI.py
ls | grep .py
```

The screenshot shows a window titled "Basic Shell GUI" with a black terminal area. The text "Welcome to the Basic Shell....." is displayed at the top. Below it, the command "Command> ls | grep .py" has been entered and executed. The output of the command is listed: "C.L.I.py", "CLI.py", and "GUI.py". At the bottom of the window, there is a text input field containing the command "ls | grep .py".

6. SIGNAL HANDLING:

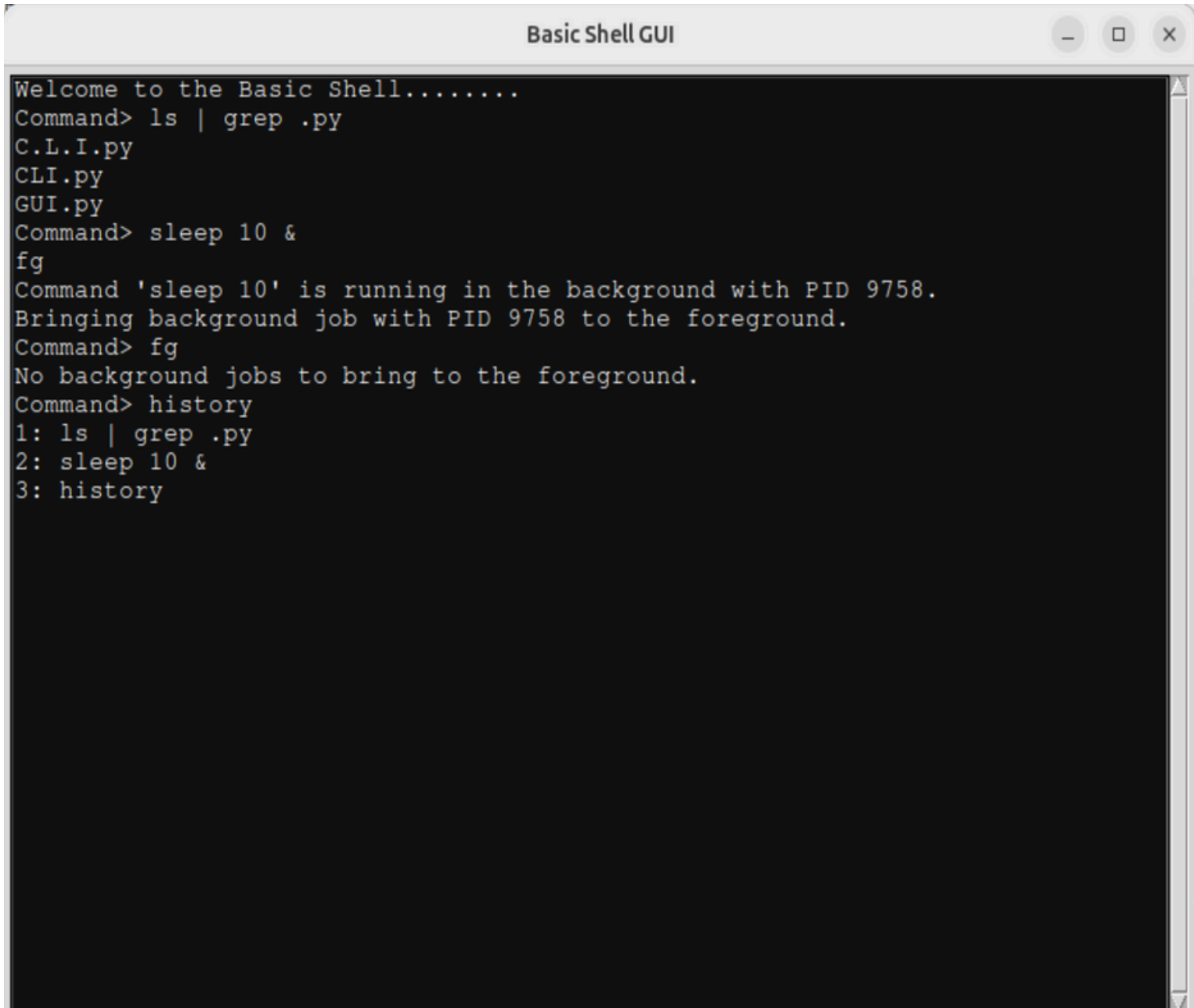
Gracefully handles interruptions (e.g., Ctrl+C) with a custom message.

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 | grep .py
C.L.I.py
CLI.py
GUI.py
Command> sleep 10 &
fg
Command 'sleep 10' is running in the background with PID 9758.
Bringing background job with PID 9758 to the foreground.
Command> fg
No background jobs to bring to the foreground.
Command> history
1: ls | grep .py
2: sleep 10 &
3: history
```

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!**

