INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

DEPARTMENT OF SOFTWARE ENGINEERING

COURSE: CS224-Operating Systems

PRESENTED TO: Ms. Tehmina Mehboob

Assignment-Mini Shell Design

BY:

Hajira Gul (4454-FOC/BSSE-F22-A)

Assignment Objective:

The purpose of this assignment is to help students understand process management, system calls, I/O handling, and command interpretation through the design and implementation of a basic shell interface, similar to Unix/Linux shells.

Assignment Task:

You are required to design and implement a Mini Shell that runs in a terminal/console window. The shell should accept user commands, interpret them, and execute them using appropriate system-level functions. Students may use any programming language of their choice (e.g., C, Python, Java, C++), but use of system-level functions and process control techniques is mandatory.

Core Functional Requirements:

1. **Prompt Display:** Display a user-friendly prompt (e.g., IIUI-Shell>) for each new command.
2. **Command Execution:** Execute simple built-in system commands (e.g., ls, pwd, echo, date, whoami, mkdir, etc.).
3. **Process Creation:** Use system calls like fork() or subprocess (depending on language) to spawn processes.
4. **Support for cd command:** Implement changing directories (cd) functionality within the shell.
5. **Exit Command:** Type exit to safely terminate the shell session.

Optional Advanced Features:

You can choose any 2 of the following enhancements:

1. Implement I/O Redirection (>, <)
2. Add Pipe Support (|)
3. Handle Background Processes (&)
4. Maintain Command History
5. Add basic Tab Autocomplete
6. Add Built-in commands (e.g., a custom help, clear, or about command)

Contents

1. Title

IIUI-Shell – A Custom Command Line Shell Interface in Python

2. Objective

To design and implement a basic cross-platform shell interface in Python that supports built-in commands, history management, piping, redirection, and system command execution for enhanced terminal interaction.

3. Scope

- Works on both **Windows** and **Linux/macOS** platforms.
- Supports common **shell functionalities** such as:
  - Built-in commands like cd, pwd, touch, rm, cat, clear, etc.
  - Pipe (|) and Redirection (>, <) operators.
  - Command **autocomplete** using readline.
  - Persistent **command history** using a file.

4. Technologies Used

- Python
- Standard Libraries:
  - os, sys, platform, shlex, subprocess
  - readline
- Shell Commands: Compatible with system shell commands like ls, mkdir, date, etc.

5. Description

IIUI-Shell is a Python-based command-line shell environment that mimics the behavior of traditional UNIX shells with added cross-platform functionality. It interprets user commands, executes them either internally or via the system shell, and handles advanced features like piping and redirection.

6. Features Implemented

The IIUI-Shell project is a custom command-line shell implemented in Python. It offers the following features:

- **Basic Shell Commands**:
  - cd <dir>: Change directory
  - pwd: Print working directory
  - clear: Clear the terminal screen
  - exit: Exit the shell
  - help: Show help message
  - history: Show command history
  - touch <file>: Create one or more files
  - rm <file>: Delete file(s).
  - cat <file>: Display file contents (cross-platform)
- **Advanced Functionalities**:
  - **Persistent Command History**:
    - Stored in ~/.iiui_shell_history
    - Automatically loads and appends new entries
  - **Tab Autocompletion** for built-in commands (using readline on Linux/macOS)
  - **Command Execution**:
    - System-level commands like ls, echo, mkdir, etc., are supported
    - ls mapped to dir on Windows for compatibility
  - **I/O Redirection**:
    - Output redirection using >
    - Input redirection using <
  - **Piping Support**:
    - Multiple commands can be chained using |
- **Custom Prompt**:
  - Displays current directory intelligently with ~ or relative path from home

7. How to Compile/Run the Shell

Requirements:

- Python 3.x installed
- Works on Linux**,** macOS, and Windows

Steps to Run:

1. Open terminal (or Command Prompt on Windows)
2. Navigate to the directory containing the script
3. Run the shell using: python shell.py (Assuming the file is named *shell.py*)

8. Issues Faced or Limitations

Issues Faced:

- **Tab completion** is not supported natively on Windows due to lack of readline module.
- **Command piping** required careful handling using subprocess.Popen() for chaining.
- One of the major goals was to make `IIUI-Shell` work on **all systems**, not just limited to a basic console on a single OS.
- Unlike simple console applications, this shell is meant to be **open and adaptable across all user environments**. Designing it to behave consistently and reliably across various setups while avoiding deep dependencies (e.g., Bash-only features) was a key challenge.

Limitations:

- No support for background process execution (&) or job control.
- Limited command validation — e.g., doesn't suggest fixes for typos.
- Does not yet support environment variable expansion or shell scripting.
- Cannot handle complex shell features like loops, variables, or aliases.

8. Conclusion

The IIUI-Shell is a functional mini-shell that combines usability and extendibility with features like command history, redirection, piping, and cross-platform compatibility. While it has limitations compared to full-fledged shells like Bash, it serves as a strong foundation for further development.