**Basic Shell Implementation with Scripting Capabilities**

**Team Members:**

|  |  |  |
| --- | --- | --- |
| S.No. | Name | USN |
| 1. | Saksham Gupta | 1RV23CY047 |
| 2. | Sarthak Lakhotia | 1RV23CY049 |

**Objective:**

The project aims to design and implement a basic shell that simulates command-line functionalities similar to Unix/Linux shells. It will include support for basic commands, scripting capabilities, piping, redirection, and process management. The project will demonstrate an understanding of process creation, inter-process communication, and file manipulation using OS system calls.

**Problem Definition:**

Modern operating systems rely on command-line shells for efficient process and file management. The implementation of a basic shell with scripting capabilities will enhance the understanding of core OS concepts, such as process handling, system calls, and file I/O, while providing a simplified platform for users to interact with the OS.

**Proposed Features:**

1. **Command Execution:** Execute basic Unix/Linux commands using fork(), exec(), and wait() system calls.
2. **Scripting Support:** Allow users to create and execute custom scripts with multiple commands.
3. **Input/Output Redirection:** Support redirection operators (>, <, >>) to manage input and output streams.
4. **Piping:** Enable inter-process communication using pipes (|) to pass output of one command as input to another.
5. **Error Handling:** Provide detailed error messages for invalid commands or failed operations.
6. **Interactive and Non-Interactive Modes:** Allow the shell to operate interactively or run predefined scripts in batch mode.

**System Calls and APIs to be Used**:

1. **Process Management**: fork(), execvp(), wait(), kill()
2. **File I/O**: open(), close(), read(), write()
3. **Pipes and Redirection**: pipe(), dup(), dup2()
4. **Environment Management**: getenv(), setenv()

**Implementation Steps**:

1. **Environment Setup**: Set up a Linux environment for development and testing.
2. **Basic Shell Implementation**:

* Create a loop to accept user input.
* Parse and execute basic commands.

1. **Advanced Features**:

* Add support for I/O redirection and piping.
* Implement a scripting engine to execute commands from a script file.

1. **Error Handling and Optimization**: Ensure robust error handling and optimize command execution.
2. **Testing and Debugging**: Test the shell with various commands, scripts, and edge cases.

**Expected Outcomes**:

* A functional shell that can execute user commands interactively and process scripts.
* Improved understanding of OS system calls, process management, and file I/O.
* Insight into designing user-friendly command-line tools.

**Tools and Platform**:

* Development: GCC, Makefile
* Platform: Linux (Ubuntu, Fedora, or Arch Linux)
* Tools: gdb for debugging

**Conclusion**:

This project will serve as a practical implementation of core OS concepts and system calls. It will provide a foundation for understanding the inner workings of shells and scripting languages while preparing students for advanced OS-level programming challenges.