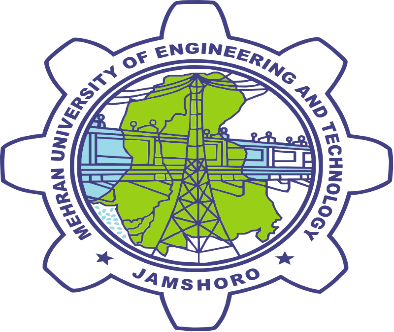
**[Mehran University of Engineering & Technology, Jamshoro Sindh Pakistan](https://www.google.com/url?sa=t&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwi7vq2wns7lAhXCa1AKHcjlDHgQFjAAegQIBRAD&url=https%3A%2F%2Fwww.muet.edu.pk%2F&usg=AOvVaw09hDIJA3HmLIX2gGhS8Xxd)**



**Complex Engineering Problem Report:  
 Operating Systems**

**SUBJECT TEACHER:**

**Dr. Bushra Naz**

**SUBMITTED BY**:  
**22CS019, 22CS041 and 22CS109**

***DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING***



|  |  |  |  |
| --- | --- | --- | --- |
| **Course: Operating Systems (CS-261)** | | | |
| **Instructor** | Dr. Bushra Naz | Assignment Type | Complex Engineering Problem |
| **Semester** | 4th | Year | 2nd |
| **Submission Deadline** | 5-11-2024 | Assessment Score | 05 |

Semester project is designed in a way to able students to solve the complex engineering problem using the Operating systems. Following characteristics of complex engineering problem are targeted in this semester project of OS.

|  |  |  |
| --- | --- | --- |
| **Complex Engineering Problem – Characteristics** | | |
| 1 | Depth of knowledge Required | Checkbox Checked with solid fill |
| 2 | Range of Conflicting Requirements | Checkbox Checked with solid fill |
| 3 | Depth of Analysis Required | Checkbox Checked with solid fill |
| 4 | Infrequently Encountered Issues Involved | □ |
| 5 | Beyond codes/standards of practice | □ |
| 6 | Diverse groups of stakeholders with widely varying needs involved | □ |
| 7 | Interdependence (high level problems including many components parts/sub-problems) | □ |
| 8 | Have significant consequences in a range of contexts | □ |
| 9 | Judgement (Require judgement in decision making) | □ |

**Isekai Shell - Gateway to a parallel digital universe**

**Technical Report: OS Design, Architecture, and Implementation**

**Overview**  
This technical report provides a comprehensive overview of the design, architecture, and implementation of the Isekai Shell, a lightweight custom shell developed to enhance the command-line interface experience. The Isekai Shell is designed to handle basic command execution, process management, job control, and I/O redirection while maintaining an intuitive user interface.

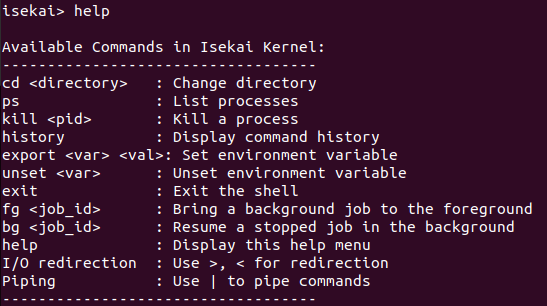
**Design and Architecture**  
The architecture of Isekai Shell is modular, comprising various components that facilitate its functionality:

* **Command Execution**: The shell utilizes the execvp function to execute external commands, with a mechanism to handle built-in commands such as cd, exit, history, and job control commands (fg, bg). This separation allows efficient management of command processing.
* **Process Management**: The Shell maintains a job table to keep track of background and foreground processes. Each job is represented by a Job struct containing the process ID (PID), command string, and status (running or stopped). This structure aids in managing multiple processes and their states.
* **Signal Handling**: The shell implements signal handling to manage interruptions gracefully. Specifically, it captures the SIGINT signal to prevent abrupt termination, instead providing a user-friendly message.
* **I/O Redirection**: Isekai Shell supports input and output redirection using > and < operators. The implementation includes the redirect\_io function, which modifies file descriptors to enable redirection, allowing users to manage file operations seamlessly.
* **Piping**: The shell supports command piping via the | operator, enabling the output of one command to be used as the input for another. This is accomplished by creating a pipe and forking processes for each command.
* **History Management**: The shell maintains a history of executed commands, allowing users to review and reuse previous commands efficiently.

**Key Algorithms and Methodologies**

* **Command Parsing**: The command input is parsed using strtok, which tokenizes the input string into arguments for further processing. This facilitates handling of various commands, including those with pipes and redirection.
* **Forking and Executing**: The core algorithm for executing commands involves forking a new process using fork(), followed by the execution of the command in the child process. The parent process waits for the child to complete, ensuring proper synchronization.
* **Signal Handling**: The use of signal to register the SIGINT handler exemplifies a methodology for enhancing user experience by handling interruptions gracefully.

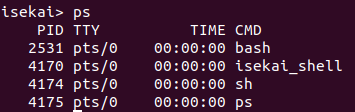
**Implementation:  
User Interface:  
A screenshot of a computer

Description automatically generated  
Help Command:  
**

**Cd command:**

****

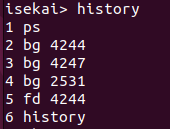
**Ps command:**

****

**Kill command:**

****

**History command:**

****

**Export/Unset command:**

****

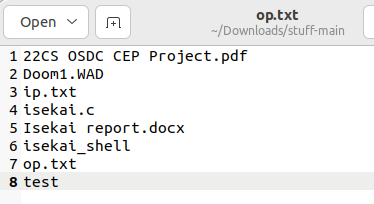
**Exit command:**

****

**I/O redirection command:**

**For >:**

****

**ls**

**For <:**

**A black background with white text

Description automatically generated**

**Piping command:**

****

**User Documentation:ls User Guides and Manuals**

**Introductionsort**  
Welcome to the Isekai Shell, a lightweight custom shell designed for an enhanced command-line experience. This user documentation provides comprehensive guidance on installation, functionality, and usage of the Isekai Shell.

**Installation Instructions**  
To install the Isekai Shell, follow these steps:

1. **Navigate to the Directory**: Change to the project directory:

cd /path/to/shell-directory

1. **Compile the Code**: Use a C compiler (like GCC) to compile the source code:

gcc isekai.c -o isekai\_shell

1. **Run the Shell**: Execute the compiled binary to start the shell:

./isekai\_shell

**Usage Instructions**

* To execute a command, simply type the command at the prompt (isekai>) and press Enter.
* For commands requiring arguments, ensure to follow the syntax outlined above (e.g., kill <pid>, cd <directory>).
* To view the available commands, type **help** to access a brief description of each command and its usage.

**Functionalities**  
The Isekai Shell supports a variety of commands and functionalities:

* **Basic Commands**: Users can execute standard commands (e.g., ls, pwd, echo) directly in the shell.
* **Changing Directories**: Use the cd <directory> command to navigate between directories. If no directory is specified, it defaults to the home directory.
* **Process Management**: Users can list running processes using the ps command and terminate processes with kill <pid>.
* **Job Control**: Background jobs can be managed with bg <job\_id> to resume a job in the background and fg <job\_id> to bring a job to the foreground.
* **History Management**: The history command displays the list of previously executed commands, allowing users to quickly recall and reuse them.
* **I/O Redirection**: Users can redirect output to a file using > and read input from a file using <.
* **Piping**: Combine commands using the | operator to send the output of one command as input to another.

**Conclusion**  
The Isekai Shell is designed for efficiency and ease of use, empowering users to navigate the command line with advanced functionalities while maintaining simplicity. For further assistance, refer to the help command or consult the technical report for deeper insights into the system's design and architecture.

This documentation serves as a complete guide to utilizing the Isekai Shell effectively, ensuring users can leverage its full capabilities.

Projects would be evaluated on the following criteria: -

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rubrics | | | | CEP  characteristics | Marks distribution |
|  | Unacceptable 2 | Acceptable 8 | Proficient 10 |
| R1: Idea/Initial Study | □ | □ | □ | WP2 | 20% |
| R2: Project Proposal | □ | □ | □ | WP1, WP3 | 20% |
| R3: Project Progress | □ | □ | □ | WP3, WP2 | 20% |
| R4: Final Report | □ | □ | □ | WP3, WP1 | 40% |