Writing a Simple Shell in C

A **shell** is a command-line interpreter that provides a user interface for accessing an operating system's services. Linux shells like `bash`, `zsh`, and `fish` are sophisticated programs with features like job control, redirection, and scripting. In this writeup, we'll explore how to **write a basic shell in C**, explaining the core components involved.

🛠️ Required System Calls

- `fork()`: Creates a new process (child).
- `execvp()`: Replaces the current process image with a new one.
- `waitpid()`: Parent waits for child to terminate.
- `getline()`: Reads a line of input from the user.

🔧 Enhancements to Explore

- **Pipelines (`|`)** – Use `pipe()` and `dup2()`.
- **Redirection (`>`, `<`)** – Use `open()` and `dup2()`.
- **Built-in commands** – Implement `cd`, `exit`, etc. without `execvp`.
- **Job control** – Handle background execution with `&`.

🧪 Example Run

```
mysh> echo Hello, World!
Hello, World!
mysh> ls -l
(total output)
mysh> exit
```

🧩 Conclusion

Writing a shell helps you deeply understand:

- Process creation
- Inter-process communication
- System calls
- Input parsing and error handling

This exercise is an essential step for systems programmers and anyone interested in OS internals or building minimal containers, scripting environments, or DevOps tools.