Project Governance and Management Documents

Project: Linux Shell

Group: Abiola, Jacky, Cameron

Date started: Jan 21, 2025

1. Code and Documentation Standards

1.1. Coding Standards

- **Language:** C
- **Naming Conventions:**
  - Variables: `snake_case`
  - Functions: `snake_case`
  - Constants: `UPPERCASE`
- **Comments:**
  - Functions: Brief description of function purpose, input, and output.
  - Code blocks: Inline comments where necessary.
  - Standard Header files for each module/file
- **Error Handling:** All system calls must have error checks. Errors should be logged with an appropriate error message.
- **Functions and Coding:**
  - Functions *should not* exceed 50-60 lines of codes
  - Functions should be generalized for tasks
  - Refactor code as necessary
  - Limit the usage of global variables
  - Appropriate Names for variables
  - Avoid code duplication where possible

1.2. Documentation Standards

- **File Documentation:** Each source file must include a header with the file name, author(s), date, and purpose.
- **Inline Comments:** Inline comments where appropriate to explain code where the code could potentially be confusing to the intended person reading the code.

2. Development Timeline

Task	Team Members	Deadline	Status
Read String	Abiola	Jan 31	Completed
String Function Library	Cameron	Jan 31	Completed
Print Tokens	Jacky	Feb 03	Completed
Struct for command	Cameron	Feb 7	Completed
Get command	Jacky	Feb 14	Completed
Run command	Abiola	Feb 14	Complete
Background Processing	Abiola	Feb 14	Complete
Job Struct	Cameron	Feb 14	Complete
Input and Output files	Jacky	Feb 19	Complete
Get Job	Abiola	Feb 20	Complete
Run Job	Abiola, Jacky	Feb 21	Complete
Error Checking	Abiola, Jacky	Feb 21	Complete
Testing	Abiola, Jacky, Cameron	Feb 21	Complete
Documentation	Abiola, Jacky, Cameron	Feb 21	Complete

3.1. Group Work Policies

- **All members must update the group via direct messages or Discord when making changes to the code.**
- **Every update should be documented in the Development Log section.**
- **If any member encounters a problem, they should inform the group as soon as possible.**
- **Communication should be constructive, with no judgment—only feedback for improvement.**
- **Meetings will have clear goals to ensure progress is tracked effectively.**
- **Members should be flexible and open to compromise when necessary.**

3.2. Division of Labour

Each member was responsible for specific tasks throughout the project:

Updated 3.2. Division of Labour

Each team member was assigned specific tasks based on their strengths and contributions to the project. Responsibilities were distributed to ensure efficiency and a balanced workload.

- **Abiola**
  - **Implemented Read String functionality.**
  - **Developed Get Job and Run Job to handle command execution.**
  - **Conducted error checking and testing to ensure stability.**
  - **Contributed to final documentation.**
- **Jacky**
  - **Designed and implemented Print Tokens for tokenizing user input.**
  - **Developed Get Command for parsing command structures.**
  - **Assisted in testing and debugging command execution.**
  - **Contributed to final documentation.**
- **Cameron**
  - **Built the String Function Library for custom string operations.**
  - **Designed the Struct for Command to store command information.**
  - **Implemented Job Struct to handle job processing.**
  - **Assisted in testing and contributed to documentation.**

---

4. Evidence of Regular Group Meetings and Development Activity

4.1. Meeting Minutes

Meeting Date: Jan 21

- **Attendees: Abiola, Jacky, Cameron**
- **Agenda: Discuss and start Week 3 tasks, establish coding standards and project scope.**
- **Actions:**
    - **Assigned initial development tasks.**
    - **Established coding standards and documentation requirements.**
    - **Set up communication and collaboration channels.**

Meeting Date: Jan 24

- **Attendees: Abiola, Jacky, Cameron**
- **Agenda: Progress check on initial implementations.**
- **Actions:**
    - **Reviewed tasks (Read String function, Tokenizer).**
    - **Identified issues with buffer size exceeding limits.**
    - **Discussed error handling strategies for system calls.**

Meeting Date: Jan 31

- **Attendees: Abiola, Jacky, Cameron**
- **Agenda: Implement String Function Library.**
- **Actions:**
    - **Created and tested `my_strlen`, `my_streq`, and `my_strcpy` functions.**
    - **Developed command struct for storing input.**
    - **Identified tokenizer issues with handling multiple spaces.**

Meeting Date: Feb 07

- **Attendees: Abiola, Jacky, Cameron**
- **Agenda: Background processing and command execution.**
- **Actions:**
    - **Fixed tokenizer issue.**
    - **Began working on command execution logic.**
    - **Discussed how to handle `exit` commands and inconsistencies.**

Meeting Date: Feb 14

- **Attendees: Abiola, Jacky, Cameron**
- **Agenda: Finalize Week 4 tasks and move to Job struct.**
- **Actions:**
    - **Implemented dynamically allocated `argv` array.**
    - **Integrated `my_strncpy` function into `str_lib`.**

    - Fixed newline issue with `cat` command.

Meeting Date: Feb 18

- Attendees: Abiola, Jacky, Cameron
- Agenda: Start Week 5 – Job processing.
- Actions:
    - Created initial Job structure.
    - Developed basic logic for handling jobs with multiple commands.
    - Addressed error handling for invalid commands.

Meeting Date: Feb 19

- Attendees: Abiola, Jacky, Cameron
- Agenda: Input and Output redirection.
- Actions:
    - Implemented `>` and `<` redirection for commands.
    - Tested background processing with `&`.
    - Identified need for centralized error-checking function.

Meeting Date: Feb 20

- Attendees: Abiola, Jacky, Cameron
- Agenda: Finalizing Job execution.
- Actions:
    - Completed `get_job()` function.
    - Optimized fork execution and signal handling.
    - Debugged segmentation fault when running multiple jobs in parallel.

Meeting Date: Feb 21

- Attendees: Abiola, Jacky, Cameron
- Agenda: Final Testing and Documentation.
- Actions:
    - Finished `run_job()`function.
    - Completed all test cases and bug fixes.
    - Finalized documentation for all functions.
    - Prepared project submission.

4.2. Development Logs

- Log 1 (Jan 24): Project Functionality Document Created.
- Log 2 (Jan 31): Initial implementation of `read_string()` and tokenizer().
- Log 3 (Feb 03): Created and tested `strlen` function; implemented command struct.
- Log 4 (Feb 07): Fixed tokenizer issues; started command execution logic.
- Log 5 (Feb 14): Completed `argv` allocation; integrated `my_strncpy()`.
- Log 6 (Feb 18): Implemented Job structure and background processing.
- Log 7 (Feb 19): Finished input/output redirection; added error handling for invalid

commands.

- **Log 8 (Feb 20): Completed `get_job()` function; improved process execution.**
- **Log 9 (Feb 21): Finalized testing and documentation.**

```
Unfinished:
Enhancement 1&3&4
For Enhancement 2, unable to get ctrl+Z to work
Couldn’t get the exceed buffer to work
usr/bin/wc < infile > outfile doesn’t work
```