5118008 English for Software Developer

# Writing Git Messages

14 May 2024

Shin Hong

## Git Commit Message Guideline (1/2)

- Start with a short subject line (less than 50 characters)
  - "If applied, this commit will THE SUBJECT LINE."
- Following an empty line after the subject, the body follows to give details.
  - explain what and why, rather than how
  - bullet items are recommended
- Use imperative mood
  - Ex. “Fix bugs” rather than “Fixed bugs”
  - Ex. “Add password length validation”, rather than “Password length validation”
- Refer to related issues or commits if exist
  - Ex. Fix Issue #12, Continue `a1b2c3d4`

## Example

```
Add password length validation


This commit adds validation required in Task-123

using the abc validation library.


It closes issue #456.
```

https://gist.github.com/rsp/057481db4dbd999bb7077f211f53f212

# Git Commit Message Guideline (2/2)

- Specify the type of commit.
  - Ex. Fix, Refactor, Feature, Docs, Style, Test, Perf, Build, Chore

- To come up with thoughtful commits, consider the following:
  - Why have I made these changes?
  - What effect have my changes made?
  - Why was the change needed?
  - What are the changes in reference to?

- Utilize `BREAKING CHANGE: <description>` to note the reason for a breaking change within the commit.

https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/

# Example: Good and Bad Subject

- Good
  - `feat: improve performance with lazy load implementation for images`
  - `chore: update npm dependency to latest version`
  - `Fix bug preventing users from submitting the subscribe form`
  - `Update incorrect client phone number within footer body per client request`
- Bad
  - `fixed bug on landing page`
  - `Changed style`
  - `oops`
  - `I think I fixed it this time?`
  - `empty commit messages`

## Conventional Commit Structure

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

```
fix: fix foo to enable bar

This fixes the broken behavior of the component by doing xyz.

BREAKING CHANGE
Before this fix foo wasn't enabled at all, behavior changes from <old> to <new>

Closes D2IQ-12345
```

## How to Write Good Commit Message

- Describe exactly what the commit does
- Describe a single, complete solution
  - break down a story ideally it should be the smallest unit of work that delivers value so should be your commits
- Be self-contained
  - Do not assume the reviewer understands what the original problem was, ensure you add it
- Clarify the scope of commit
  - Ex. `feat(profile): add button for update call`

## GitHub Issue

- Issues are means to structure ideas and collaboration
- Issues should focus on ideas, problems, and solutions
- Issue queue is where real collaboration is done.
  - First places for tracking collaboration
  - Issues must be written in a clean, structured, and transparent manner.

https://wiredcraft.com/blog/how-we-write-our-github-issues/

# How To Write Issue

- Context: explain the conditions which led you to write this issue.
  - Ex. “Since we’ve moved to the latest version of Express.js, we’ve experienced a few performance issues (#14 and #15) on production.”

- Problem or idea: the context should lead to something, an idea or a problem that you’re facing.
  - Ex. “We’ve had no way of easily seeing the performance impact before releasing our changes to production.”

- Solution or next step
  - Propose a next step towards solving the issue.
  - Engage others (request feedback), assign somebody else to the issue,
  - Ex. “@bobby see with @johnny if he has a tool that could provide us insights on the performances in the development environment. We should include something similar in our development and deployment processes for future projects.”

## Guideline

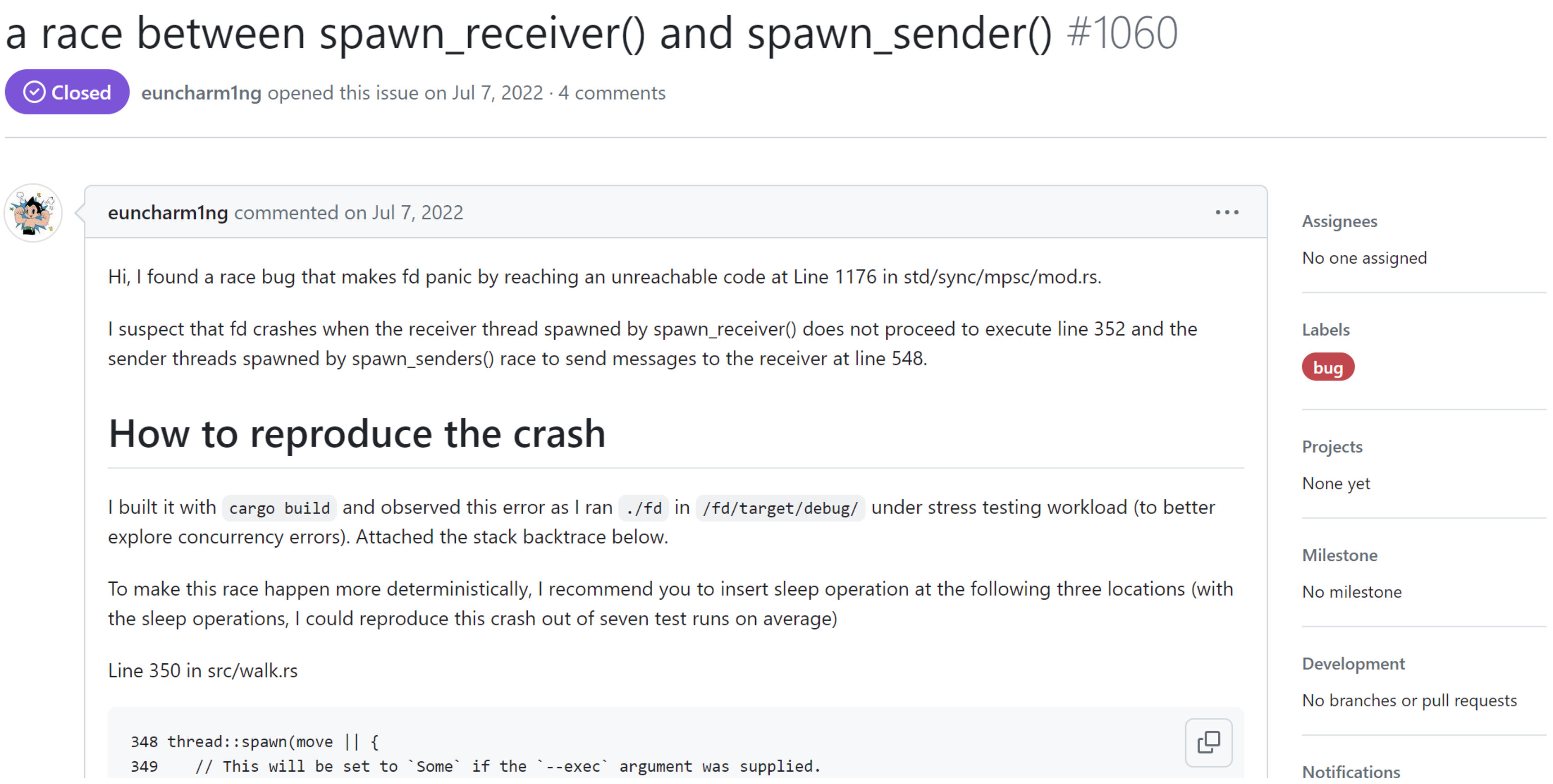
- Keep titles short and descriptive.
- Keep your messages as short and to the point as possible.
    - Bullet points are often a great way of quickly structuring complex ideas or solutions.
- Include the right people in your discussion.
    - Mentioning people in the issue (Ex. @bob @john in our issues).
- Format your messages.
    - Help the reader focus on what matters and understand your message
    - GitHub flavored markdown has a simple but effective syntax
- Add links to you references

## Bug Report Guideline

```
#### Description

A clear and concise description of what the issue is about.


#### Screenshots

![Downhill Windmills](http://i.giphy.com/K08AG2EByqkFi.gif)


#### Files

A list of relevant files for this issue. This will help people navigate the project and offer some clues of where to
start.


#### To Reproduce

If this issue is describing a bug, include some steps to reproduce the behavior.


#### Tasks

Include specific tasks in the order they need to be done in. Include links to specific lines of code where the task
should happen at.

- [ ] Task 1

- [ ] Task 2

- [ ] Task 3
```

# Example

- https://github.com/sharkdp/fd/issues/1060



a race between spawn_receiver() and spawn_sender() #1060

Closed euncharm1ng opened this issue on Jul 7, 2022 · 4 comments

euncharm1ng commented on Jul 7, 2022

Hi, I found a race bug that makes fd panic by reaching an unreachable code at Line 1176 in std/sync/mpsc/mod.rs.

I suspect that fd crashes when the receiver thread spawned by spawn_receiver() does not proceed to execute line 352 and the sender threads spawned by spawn_senders() race to send messages to the receiver at line 548.

## How to reproduce the crash

I built it with `cargo build` and observed this error as I ran `./fd` in `/fd/target/debug/` under stress testing workload (to better explore concurrency errors). Attached the stack backtrace below.

To make this race happen more deterministically, I recommend you to insert sleep operation at the following three locations (with the sleep operations, I could reproduce this crash out of seven test runs on average)

Line 350 in src/walk.rs

```
348 thread::spawn(move || {
349     // This will be set to `Some` if the `--exec` argument was supplied.
```

Assignees

No one assigned

Labels

bug

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

Notifications

## Personal Homework 2

- Add at least one new issue and make at least one commit in your GitHub repository
  - Write commit and issue messages following the guideline

- Write one-page summary report of your issue and commit
  - including the links to the issue and commit

- Submission
  - By 9 PM, May 24 Fri
  - Via LMS