# Convert interactive rebase to C

Alban Gruin

**Abstract**

git is a modular source control management software, and all of its subcommands are programs on their own. A lot of them are written in C, but a couple of them are shell or Perl scripts. This is the case of `git-rebase--interactive` (or interactive rebase), which is a shell script. Rewriting it in C would improve its performance, its portability, and maybe its robustness.

## 1 About `git-rebase` and `git-rebase--interactive`

`git-rebase` allows to re-apply changes on top of another branch. For instance, when a local branch and a remote branch have diverged, `git-rebase` can re-unify them, applying each change made on the local branch on top of the remote branch.

`git-rebase--interactive` is used to reorganize commits by reordering, rewording, or squashing them. To achieve this purpose, `git` opens the list of commits to be modified in a text editor (hence the interactivity), as well as the actions to be performed for each of them.

## 2 Project goals

Back in 2016, Johannes Schindelin discussed[1] about retiring `git-rebase.sh` (described here as a "hacky shell script") in favour of a builtin. He explained that, as it's only a command-line parser for `git-rebase--am.sh`, `git-rebase--merge.sh`, and `git-rebase--interactive.sh`, these 3 scripts should be rewritten first, preferably starting with the latter. Since then, there have been some effort ongoing to progressively rewrite this script in C, but a lot remains to be done.

[1] `https://public-inbox.org/git/alpine.DEB.2.20.1609021432070.129229@virtualbox/`

The goal of this project is to rewrite `git-rebase--interactive.sh` in C.

# 3 Benefits to the community

## 3.1 Performance improvements

Shell scripts are inherently slow. That's because each command is a program by itself. So, for each command, the shell interpreter has to spawn a new process and to load a new program (with `fork()` and `exec()` syscalls), which is an expensive process.

Those commands can be other `git` commands. Sometimes, they are wrappers to call internal C functions (eg. `git-rebase--helper`), something shell scripts can't do natively. These wrappers basically parse the parameters, then start the appropriate function, which is obviously slower than just calling a function from C.

Other commands can be POSIX utilities (eg. `sed`, `cut`, etc.). They suffer from the same performance problem, which worsens on non-POSIX platforms such as Windows because they have to rely on an emulation layer.

Performance improvement is not anecdotal : when Johannes Schindelin began to rewrite some parts of `git-rebase--interactive` in C, its performance increased from 3 to 5 times, depending on the platform[2], Windows benefiting the most.

POSIX utilities have their own problems, speed aside, namely portability.

## 3.2 Portability improvements

Shell scripts often rely on many of those POSIX utilities, which are not necessarily available natively on all platforms, or may have more or less features depending on the implementation. On non-POSIX platforms, this requires the inclusion of those utilities, as well as the aforementioned emulation layer.

Although C is not perfect portability-wise, it's still better than shell scripts. For instance, the resulting binaries will not necessarily depend on third-party programs or libraries.

[2] `https://public-inbox.org/git/cover.1472457609.git.johannes.schindelin@gmx.de/`

# 4 Risks

Of course, rewriting a piece of software takes time, and can lead to regressions (ie. new bugs). To mitigate this risk, I should understand well the functions I want to rewrite, run tests on a regular basis, and write new if needed, and of course discuss about my code with the community during reviews.

# 5 Related work

There is another project idea consisting in converting Perl or shell scripts to builtins. Two others GSoC students therefore proposed to rewrite `git-stash`.

# 6 Approximative timeline

Normally, I would be able to work 35 to 40 hours a week. When I have courses or exams at university, I could work between 20 and 25 hours a week.

During this period, I will communicate weekly on the state and progress of my work on the mailing list.

Once the GSoC is over, I intend to continue contributing to git. I hope that this experience will give me the necessary skills to do it efficiently.

## 6.1 Community bonding — April 23, 2018 – May 14, 2018

*I will still have courses at the university during this period.*

During the community bonding, I would like to dive into `git`'s codebase, and to understand what `git-rebase--interactive` does under the hood. At the same time, I'd communicate with the community and my mentor, seeking for clarifications, and asking questions about how things should or should not be done.

## 6.2 Weeks 1 & 2 — May 14, 2018 – May 27, 2018

*From May 14 to 18, I have exams at the university, so I won't be able to work full time.*

I would search for edge cases not covered by current tests and write some if needed.

### 6.3 Week 3 — May 28, 2018 – June 3, 2018

At the same time, I would refactor `--preserve-merges` in its own shell script, if it has not been deprecated or moved in the meantime. Johannes Schindelin explained that this would be the first step of the conversion[1]. This operation is not really tricky by itself, as `--preserve-merges` is about only 50 lines of code into `git_rebase__interactive()`, plus some specific functions (eg. `pick_one()`).

### 6.4 Weeks 4 to 7 — June 4, 2018 – July 1, 2018

Then, I would start to incrementally rewrite `git-rebase--interactive.sh` functions in C, and move them `git-rebase--helper.c`. Newly-rewritten C functions are then associated to command-line parameters to be able to use them from shell scripts.

Examples of such conversion can be found in commits `0cce4a2756`[3] (rebase -i -x: add exec commands via the rebase–helper) and `b903674b35`[4] (bisect–helper: `‘is_expected_rev‘` & `‘check_expected_revs‘` shell function in C).

There is a lot of functions into `git-rebase--interactive.sh` to rewrite. Most of them are small, and some of them are even wrappers for a single command (eg. `do_next()`), so they shouldn't be really problematic.

A couple of them are quite long (eg. `pick_one()`), and will probably be even longer once rewritten in C due to the low-level nature of the language. They also tend to depend a lot on other smaller functions.

The plan here would be to start rewriting the smaller functions when applicable (ie. they're not a simple command wrapper) before working on the biggest of them.

Of course, rewriting a function should not cause any breakage in the test suite. C also brings another class of problems related to memory management, such as memory leaks. To avoid them, I will analyze my code with `valgrind`.

I will also document my code as much as possible.

### 6.5 Week 8 — July 2, 2018 – July 8, 2018

Then, I plan to polish the new code, in order to improve its performance or to make it more readable.

Benchmarking will be done using the existing framework.

[3] `https://git.kernel.org/pub/scm/git/git.git/commit/?id=0cce4a2756`
[4] `https://git.kernel.org/pub/scm/git/git.git/commit/?id=b903674b35`

### 6.6 Weeks 9 & 10 — July 9, 2018 – July 22, 2018

When all majors functions from `git-rebase--interactive.sh` have been rewritten in C, I would retire the script in favour of a builtin.

### 6.7 Weeks 11 & 12 — July 23, 2018 – August 5, 2018

In the last two weeks, I would improve the code coverage where needed.

I also plan to use this time as a backup if I am late on my planning.

### 6.8 Wrapping up — August 6 – August 12, 2018

Lastly, I will submit my code for evaluation. I may begin to work on my wish list below.

## 7 If times permits

- Add an option to include the patch in the message of commits to be reworded, as proposed by Ævar Arnfjörð Bjarmason[5].

## 8 About me

My name is Alban Gruin. I am an undergraduate at the Toulouse III — Paul Sabatier University, France, where I have been studying Computer Sciences for a year and a half. My timezone is UTC+02:00.

I have been programming in C for the last 5 years. I learned using freely available resources online, and by attending class ever since last year.

I am also quite familiar with shell scripts, and I have been using `git` for the last 3 years.

My e-mail address is `alban.gruin@gmail.com`, my IRC nick is `abngrn`. My personnal website is located at `https://pa1ch.fr/` (in French), and some of my projects can be found on Github (`agrn`) as well as on my website.

My micro-project was "userdiff: add built-in pattern for golang"[6, 7].

[5] `https://public-inbox.org/git/87in9ucsbb.fsf@evledraar.gmail.com/`

[6] `https://public-inbox.org/git/20180228172906.30582-1-alban.gruin@gmail.com/`

[7] `https://git.kernel.org/pub/scm/git/git.git/commit/?id=1dbf0c0a`