Contributing

============

The simdjson library is an open project written in C++. Contributions are invited. Contributors

agree to the project's license.

We have an extensive list of issues, and contributions toward any of these issues is invited.

Contributions can take the form of code samples, better documentation or design ideas.

In particular, the following contributions are invited:

- The library is focused on performance. Well-documented performance optimization are invited.

- Fixes to known or newly discovered bugs are always welcome. Typically, a bug fix should come with

a test demonstrating that the bug has been fixed.

- The simdjson library is advanced software and maintainability and flexibility are always a

concern. Specific contributions to improve maintainability and flexibility are invited.

We discourage the following types of contributions:

- Code refactoring. We all have our preferences as to how code should be written, but unnecessary

refactoring can waste time and introduce new bugs. If you believe that refactoring is needed, you

first must explain how it helps in concrete terms. Does it improve the performance?

- Applications of new language features for their own sake. Using advanced C++ language constructs

is actually a negative as it may reduce portability (to old compilers, old standard libraries and

systems) and reduce accessibility (to programmers that have not kept up), so it must be offsetted

by clear gains like performance or maintainability. When in doubt, avoid advanced C++ features

(beyond C++11).

- Style formatting. In general, please abstain from reformatting code just to make it look prettier.

Though code formatting is important, it can also be a waste of time if several contributors try to

tweak the code base toward their own preference. Please do not introduce unneeded white-space

changes.

In short, most code changes should either bring new features or better performance. We want to avoid unmotivated code changes.

Specific rules

----------

We have few hard rules, but we have some:

- Printing to standard output or standard error (`stderr`, `stdout`, `std::cerr`, `std::cout`) in the core library is forbidden. This follows from the [Writing R Extensions](https://cran.r-project.org/doc/manuals/R-exts.html) manual which states that "Compiled code should not write to stdout or stderr".

- Calls to `abort()` are forbidden in the core library. This follows from the [Writing R Extensions](https://cran.r-project.org/doc/manuals/R-exts.html) manual which states that "Under no circumstances should your compiled code ever call abort or exit".

- All source code files (.h, .cpp) must be ASCII.

Tools, tests and benchmarks are not held to these same strict rules.

General Guidelines

----------

Contributors are encouraged to :

- Document their changes. Though we do not enforce a rule regarding code comments, we prefer that non-trivial algorithms and techniques be somewhat documented in the code.

- Follow as much as possible the existing code style. We do not enforce a specific code style, but we prefer consistency.

- Modify as few lines of code as possible when working on an issue. The more lines you modify, the harder it is for your fellow human beings to understand what is going on.

- Tools may report "problems" with the code, but we never delegate programming to tools: if there is a problem with the code, we need to understand it. Thus we will not "fix" code merely to please a static analyzer if we do not understand.

- Provide tests for any new feature. We will not merge a new feature without tests.

Pull Requests

--------------

Pull requests are always invited. However, we ask that you follow these guidelines:

- It is wiser to discuss your ideas first as part of an issue before you start coding. If you omit this step and code first, be prepare to have your code receive scrutiny and be dropped.

- Users should provide a rationale for their changes. Does it improve performance? Does it add a feature? Does it improve maintainability? Does fix a bug? This must be explicitly stated as part of the pull request. Do not propose changes based on taste or intuition. We do not delegate programming to tools: that some tool suggested a code change is not reason enough to change the code.

1. When your code improves performance, please document the gains with a benchmark using hard numbers.

2. If your code fixes a bug, please be either fix a failing test, or propose a new test.

3. Other types of changes must be clearly motivated. We openly discourage changes with no identifiable benefits.

- Changes should be focused and minimal. You should change as few lines of code as possible. Please do not reformat or touch files needlessly.

- New features must be accompanied of new tests, in general.

- Your code should pass our continuous-integration tests. It is your responsability to ensure that your proposal pass the tests. We do not merge pull requests that would break our build.

If the benefits of your proposed code remain unclear, we may choose to discard your code: that is not an insult, we frequently discard our own code. We may also consider various alternatives and choose another path. Again, that is not an insult or a sign that you have wasted your time.

Style

-----

Our formatting style is inspired by the LLVM style.

The simdjson library is written using the snake case: when a variable or a function is a phrase, each space is replaced by an underscore character, and the first letter of each word written in lowercase. Compile-time constants are written entirely in uppercase with the same underscore convention.

Code of Conduct

---------------

Though we do not have a formal code of conduct, we will not tolerate bullying, bigotry or

intimidation. Everyone is welcome to contribute. If you have concerns, you can raise them privately with the core team members (e.g., D. Lemire, J. Keiser).

We welcome contributions from women and less represented groups. If you need help, please reach out.

Consider the following points when engaging with the project:

- We discourage arguments from authority: ideas are discusssed on their own merits and not based on who stated it.

- Be mindful that what you may view as an aggression is maybe merely a difference of opinion or a misunderstanding.

- Be mindful that a collection of small aggressions, even if mild in isolation, can become harmful.

Getting Started Hacking

-----------------------

An overview of simdjson's directory structure, with pointers to architecture and design

considerations and other helpful notes, can be found at [HACKING.md](HACKING.md).