[Source](https://subsurface-divelog.org/documentation/contributing/ "Permalink to Contributing | Subsurface")

# Contributing | Subsurface

It might be a good idea to join our [mailing list][1]. Conversation there is in English -- even though this site (and Subsurface itself) are available in many languages, the shared language we all communicate in is English. Actually "Broken English" is just fine… :-)

We also tend to have some developers hanging out in the `#subsurface` channel on [Freenode][2].

There are many ways in which you can contribute. We are always looking for testers who are willing to test the code while it is being developed. We especially need people running Windows and Mac (as the majority of the active developers are Linux people). We are also always looking for volunteers who help reviewing and improving the documentation. And very importantly we are looking for translators willing to translate the software into different languages. Our translations are centrally handled at [Transifex][3] \-- please sign up for an account there and then request to join the [Subsurface Team][4].

If you would like to contribute patches that fix bugs or add new features, that is of course especially welcome. If you are looking for places to start, look at the open bugs in our [bug tracker][5].

Here is a very brief introduction on creating commits that you can either send as [pull requests][6] on our main GitHub repository or send as emails to the mailing list. Much more details on how to use git can be found at the [git user manual][7].

Start with getting the latest source.

`cd subsurface`

`git checkout master`

`git pull`

ok, now we know you're on the latest version. Create a working branch to keep your development in:

`git checkout -b devel`

Edit the code (or documentation), compile, test… then create a commit:

`git commit -s -a`

Depending on your OS this will open a default editor -- usually you can define which by setting the environment variable `GIT\_EDITOR`. Here you enter your commit message. The first line is the title of your commit. Keep it brief and to the point. Then a longer explanation (more on this and the fact that we insist on all contributions containing a Signed-off-by: line below).

If you want to change the commit message, "git commit --amend" is the way to go. Feel free to break your changes into multiple smaller commits. Then, when you are done there are two directions to go, which one you find easier depends a bit on how familiar you are with GitHub. You can either push your branch to GitHub and create a [pull requests on GitHub][6], or you run

`git format-patch master..devel`

Which creates a number of files that have names like 0001-Commit-title.patch, which you can then send to our developer mailing list.

When sending code, please either send signed-off patches or a pull request with signed-off commits. If you don't sign off on them, we will not accept them. This means adding a line that says "Signed-off-by: Name \<Email\>" at the end of each commit, indicating that you wrote the code and have the right to pass it on as an open source patch.

See: [Signed-off-by Lines][8]

Also, please write good git commit messages. A good commit message looks like this:

Header line: explaining the commit in one line

Body of commit message is a few lines of text, explaining things

in more detail, possibly giving some background about the issue

being fixed, etc etc.

The body of the commit message can be several paragraphs, and

please do proper word-wrap and keep columns shorter than about

74 characters or so. That way "git log" will show things

nicely even when it's indented.

Reported-by: whoever-reported-it

Signed-off-by: Your Name \<you@example.com\>

That header line really should be meaningful, and really should be just one line. The header line is what is shown by tools like gitk and shortlog, and should summarize the change in one readable line of text, independently of the longer explanation.

The preferred way to write a commit message is using imperative mood, e.g. "Make foo do xyz" instead of "This patch makes foo do xyz" or "I made foo do xyz", as if you are giving commands or requests to the code base.

https://en.wikipedia.org/wiki/Imperative\_mood

![gitk sample][9]

Example with gitk

Additionally when important changes to behaviors, fixes or new feature are introduced an entry must be added to CHANGELOG.md file. Always add new entries at the very top of the file above other existing entries. Use this layout for new entries:

Area: Details about the change [reference thread / issue]

When multiple areas are affected list the areas separated with a /:

Area1/Area2: Detail about the change [reference thread / issue]

Here is a (non exhaustive) list of Areas that can be used:

\* Bluetooth

\* Cloud-storage

\* Desktop

\* Dive pictures

\* Import

\* Libdivecomputer

\* Map-widget

\* Mobile

\* Planner

\* Printing

\* Profile

In order to make reviews simpler and have contributions merged faster in the code base, please follow Subsurface project's coding style and coding conventions described in the [CodingStyle][10] file.

[1]: http://lists.subsurface-divelog.org/cgi-bin/mailman/listinfo/subsurface

[2]: http://freenode.net/

[3]: https://www.transifex.com/

[4]: https://www.transifex.com/projects/p/subsurface/

[5]: https://github.com/Subsurface/subsurface/issues

[6]: https://github.com/Subsurface/subsurface/pulls

[7]: https://www.kernel.org/pub/software/scm/git/docs/user-manual.html

[8]: https://gerrit-review.googlesource.com/Documentation/user-signedoffby.html

[9]: https://subsurface-divelog.org/wp-content/uploads/2011/10/Screenshot-gitk-subsurface-1.png "Example with gitk"

[10]: https://github.com/Subsurface/subsurface/blob/master/CodingStyle