Git Commit Messages: 50/72 Formatting

Asked 12 years, 7 months ago Modified 1 year, 3 months ago Viewed 127k times

365

173

Tim Pope argues for a particular Git commit message style in his blog post: http://www.tpope.net/node/106.

Here is a quick summary of what he recommends:

- First line is 50 characters or less.
- Then a blank line.
- Remaining text should be wrapped at 72 characters.

His blog post gives the rationale for these recommendations (which I will call "50/72 formatting" for brevity):

- In practice, some tools treat the first line as a subject line and the second paragraph as a body (similar to email).
- `git log` does not handle wrapping, so it is hard to read if lines are too long.
- `git format-patch --stdout` converts commits to email — so to play nice it helps if your commits are already wrapped nicely.

A point I would like to add that I think Tim would agree with:

- The act of summarizing your commit is a good practice inherently in any version control system. It helps others (or a later you) find relevant commits more quickly.

So, I have a couple of angles to my question:

- What chunk (roughly) of the "thought leaders" or "experienced users" of Git embrace the 50/72 formatting style? I ask this because sometime newer users don't know or don't care about community practices.
- For those that don't use this formatting, is there a principled reason for using a different formatting style? (Please note that I'm looking for an argument on the merits, not "I've never heard of it" or "I don't care.")
- Empirically speaking, what percentage of Git repositories embrace this style? (In case someone wants to do an analysis on GitHub repositories… hint, hint.)

My point here is not to recommend the 50/72 style or shoot down other styles. (To be open about it, I do prefer it, but I am open to other ideas.) I just want to get the rationale for why people like or oppose various Git commit message styles. (Feel free to bring up points that haven't been mentioned, too.)

Share Improve this question edited Feb 20, 2020 at 3:03 asked Feb 18, 2010 at 16:02

Follow ib. 26.7k 10 77 99 David J. 30.3k 21 116 171

16 I just noticed that Github's web interface will warn you if your first line is longer than 50 characters by saying "ProTip: Great commit summaries are 50 characters or less. Place extra information in the extended description." – David J. Jun 1, 2014 at 15:22

5 Answers

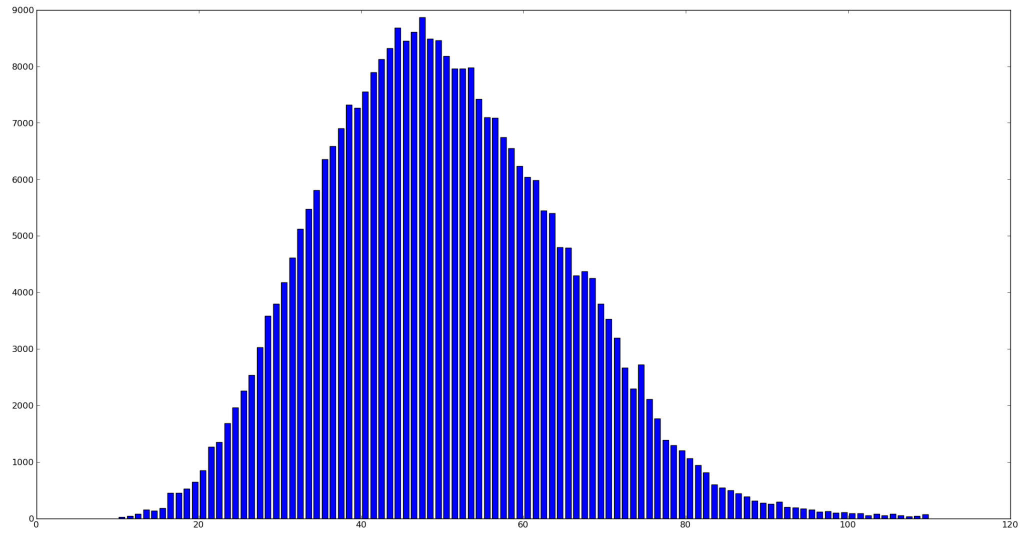
Sorted by: Trending sort available Highest score (default)

321

Regarding the “summary” line (the 50 in your formula), the Linux kernel documentation has this to say:

```
For these reasons, the "summary" must be no more than 70-75
characters, and it must describe both what the patch changes, as well
as why the patch might be necessary.  It is challenging to be both
succinct and descriptive, but that is what a well-written summary
should do.
```

That said, it seems like kernel maintainers do indeed try to keep things around 50. Here’s a histogram of the lengths of the summary lines in the git log for the kernel:



(view full-sized)

There is a smattering of commits that have summary lines that are longer (some much longer) than this plot can hold without making the interesting part look like one single line. (There’s probably some fancy statistical technique for incorporating that data here but oh well... :-)

If you want to see the raw lengths:

```
cd /path/to/repo
git shortlog  | grep -e '^      ' | sed 's/[[:space:]]\+\(.*\)$/\1/' | awk '{print
length($0)}'
```

or a text-based histogram:

Share Improve this answer Follow

edited Feb 20, 2020 at 3:06
ib.
26.7k 10 77 99

answered Aug 16, 2012 at 18:12
mgalgs
14.7k 9 59 66

27 How did you generate your histogram, out of curiosity? – anarchivist Aug 17, 2012 at 0:11

45 matplotlib in python. Something like this but with the output from one of the commands in my answer instead of the random data. – mgalgs Aug 17, 2012 at 4:11

3 Using GNU AWK: `git shortlog | awk '/^      / {gensub(/[[:space:]]\+\(.*\)$/, "\\1", ""); print length()}'` – Dennis Williamson Dec 8, 2013 at 15:42

6 Github will hide commit message text after the 70th character. – Peeter Kokk Jul 18, 2016 at 9:21

3 concerning the "fancy statistical technique", you could simple make the last bin e.g. "≥ 100" – Tobias Kienzler Mar 23, 2017 at 8:24

90

Regarding “thought leaders”: Linus emphatically advocates line wrapping for the full commit message:

> [...] we use 72-character columns for word-wrapping, except for quoted material that has a specific line format.

The exceptions refers mainly to “non-prose” text, that is, text that was not typed by a human for the commit — for example, compiler error messages.

Share Improve this answer Follow

edited Feb 20, 2020 at 2:37
ib.
26.7k 10 77 99

answered Jul 22, 2013 at 16:12
leonbloy
70.4k 20 136 187

33 +1 for bringing up difference between "prose" and "non-prose". And "except for quoted material that has a specific line format". Excellent rule of thumb. – Alois Mahdal Nov 6, 2014 at 15:12

49

Separation of presentation and data drives my commit messages here.

Your commit message should not be hard-wrapped at *any* character count and instead line breaks should be used to separate thoughts, paragraphs, etc. as part of the data, not the presentation. In this case, the "data" is the message you are trying to get across and the "presentation" is how the user sees that.

I use a single summary line at the top and I try to keep it short but I don't limit myself to an arbitrary number. It would be far better if Git actually provided a way to store summary messages as a separate entity from the message but since it doesn't I have to hack one in and I use the first line break as the delimiter (luckily, many tools support this means of breaking apart the data).

For the message itself newlines indicate something meaningful in the data. A single newline indicates a start/break in a list and a double newline indicates a new thought/idea.

```
This is a summary line, try to keep it short and end with a line break.
This is a thought, perhaps an explanation of what I have done in human readable format.  It
may be complex and long consisting of several sentences that describe my work in essay
format.  It is not up to me to decide now (at author time) how the user is going to consume
this data.

Two line breaks separate these two thoughts.  The user may be reading this on a phone or a
wide screen monitor.  Have you ever tried to read 72 character wrapped text on a device
that only displays 60 characters across?  It is a truly painful experience.  Also, the
opening sentence of this paragraph (assuming essay style format) should be an intro into
the paragraph so if a tool chooses it may want to not auto-wrap and let you just see the
start of each paragraph.  Again, it is up to the presentation tool not me (a random author
at some point in history) to try to force my particular formatting down everyone else's
throat.

Just as an example, here is a list of points:
* Point 1.
* Point 2.
* Point 3.
```

Here's what it looks like in a viewer that soft wraps the text.

> This is a summary line, try to keep it short and end with a line break.
>
> This is a thought, perhaps an explanation of what I have done in human readable format. It may be complex and long consisting of several sentences that describe my work in essay format. It is not up to me to decide now (at author time) how the user is going to consume this data.

```
Also, the opening sentence of this paragraph (assuming essay style format) should be
an intro into the paragraph so if a tool chooses it may want to not auto-wrap and let
you just see the start of each paragraph. Again, it is up to the presentation tool not
me (a random author at some point in history) to try to force my particular formatting
down everyone else's throat.

Just as an example, here is a list of points:
* Point 1.
* Point 2.
* Point 3.
```

My suspicion is that the author of Git commit message recommendation you linked has never written software that will be consumed by a wide array of end-users on different devices before (i.e., a website) since at this point in the evolution of software/computing it is well known that storing your data with hard-coded presentation information is a bad idea as far as user experience goes.

Share Improve this answer Follow edited Oct 16, 2017 at 3:15 answered Aug 20, 2013 at 2:57

Micah Zoltu
6,270 3 43 69

64 Wow, that commit message is painful to read even on a webpage like SO. I don't need *responsive* commit messages, but something which works well with `tig`, `git log` or `gitk`, and maybe also github. – Benjamin Bannier Aug 20, 2013 at 4:07

34 The message would be easy to read with any viewer that word wraps. I put it in a non-wrapping code block as an example. – Micah Zoltu Aug 20, 2013 at 7:41

23 Thanks for a different perspective. In theory, your answer sounds fine. In practice, I like line breaks for current command line tools. – David J. Aug 21, 2013 at 1:44

20 The character sequence `\n\n` is a thought separator. `\n*` is a list item indicator. How those are rendered is up to the view. The problem with artificial line breaks is that they are associated with nothing *except* the presentation. There isn't any data-related information being transmitted by putting a line break at 70 characters. My choice of `\n\n` and `\n*` is the same as why markdown chose it, because it is a form of encoding data that also happens to look somewhat reasonable in a plain text view. – Micah Zoltu Nov 6, 2014 at 20:38

21 Hard wraps are difficult to read on devices with small screens (mobile). The message will be difficult to read somewhere no matter what you do. I would rather follow modern best practices than cater to legacy software that doesn't have some of the most basic rendering capabilities. – Micah Zoltu Jun 15, 2015 at 14:06

9

Is the maximum recommended title length really 50?

I have believed this for years, but as I just noticed the documentation of "git commit" actually states

```
$ git help commit | grep -C 1 50
      Though not required, it’s a good idea to begin the commit message with
      a single short (less than 50 character) line summarizing the change,
      followed by a blank line and then a more thorough description. The text

$  git version
git version 2.11.0
```

One could argue that "less then 50" can only mean "no longer than 49".

Share Improve this answer Follow

answered Jul 4, 2019 at 11:34
Guenther Bruntthaler
649 5 10

5 On the other hand, the default highlighting highlights the first 50 characters. This appears to be an indeliberate discrepancy. – August Janse Sep 2, 2019 at 11:35

6

I'd agree it is interesting to propose a particular style of working. However, unless I have the chance to set the style, I usually follow what's been done for consistency.

Taking a look at the Linux Kernel Commits, the project that started git if you like, http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=bca476139d2ded86be146dae09b06e22548b67f3, they don't follow the 50/72 rule. The first line is 54 characters.

I would say consistency matters. Set up proper means of identifying users who've made commits (user.name, user.email - especially on internal networks. User@OFFICE-1-PC-10293982811111 isn't a useful contact address). Depending on the project, make the appropriate detail available in the commit. It's hard to say what that should be; it might be tasks completed in a development process, then details of what's changed.

I don't believe users should use git one way because certain interfaces to git treat the commits in certain ways.

I should also note there are other ways to find commits. For a start, `git diff` will tell you what's changed. You can also do things like `git log --pretty=format:'%T %cN %ce'` to format the options of `git log` .

Share Improve this answer Follow

answered Feb 18, 2010 at 16:24
user257111

For reference he says "As the example indicates, you should shoot for about 50 characters (though this isn't a hard maximum)", but I suppose you have a point in that you shouldn't have to work around your tools. – Omni5cience Feb 23, 2011 at 0:13