What's the point in adding a new line to the end of a file?

Asked 12 years, 4 months ago Modified 1 year, 3 months ago Viewed 297k times

316

Some compilers (especially C or C++ ones) give you warnings about:

```
No new line at end of file
```

I thought this would be a C-programmers-only problem, but github displays a message in the commit view:

```
\ No newline at end of file
```

for a PHP file.

I understand the preprocessor thing explained in this thread, but what has this to do with PHP? Is it the same `include()` thing or is it related to the `\r\n` vs `\n` topic?

What is the point in having a new line at the end of a file?

23, 2017 at 12:40
nmunity Bot

asked Aug 14, 2011 at 19:29
Philipp Stephan
3,270 2 15 8

– AlikElzin-kilaka Dec 31, 2018 at 12:55

final "line" if it does not end with a newline.

Sorted by: Highest score (default)

It's not about adding an extra newline at the end of a file, it's about not removing the newline that should be there.

A text file, under unix, consists of a series of lines, each of which ends with a newline character (`\n`). A file that is not empty and does not end with a newline is therefore not a text file.

Utilities that are supposed to operate on text files may not cope well with files that don't end with a newline; historical Unix utilities might ignore the text after the last newline, for example. GNU utilities have a policy of behaving decently with non-text files, and so do most other modern utilities, but you may still encounter odd behavior with files that are missing a final newline[1].

With GNU diff, if one of the files being compared ends with a newline but not the other, it is careful to note that fact. Since diff is line-oriented, it can't indicate this by storing a newline for one of the files but not for the others — the newlines are necessary to indicate where each line *in the diff file* starts and ends. So diff uses this special text `\ No newline at end of file` to differentiate a file that didn't end in a newline from a file that did.

By the way, in a C context, a source file similarly consists of a series of lines. More precisely, a translation unit is viewed in an implementation-defined as a series of lines, each of which must end with a newline character (n1256 §5.1.1.1). On unix systems, the mapping is straightforward. On DOS and Windows, each CR LF sequence (`\r\n`) is mapped to a newline (`\n`; this is what always happens when reading a file opened as text on these OSes). There are a few OSes out there which don't have a newline character, but instead have fixed- or variable-sized records; on these systems, the mapping from files to C source introduces a `\n` at the end of each record. While this isn't directly relevant to unix, it does mean that if you copy a C source file that's missing its final newline to a system with record-based text files, then copy it back, you'll either end up with the incomplete last line truncated in the initial conversion, or an extra newline tacked onto it during the reverse conversion.

[1] Example: the output of GNU `sort` on non-empty files always ends with a newline. So if the file `foo` is missing its final newline, you'll find that `sort foo | wc -c` reports one more byte than `cat foo | wc -c`. The `read` builtin of `sh` is required to return *false* if the end-of-file is reached before the end of the line is reached, so you'll find that loops such as `while IFS= read -r line; do ...; done` skip an unterminated line altogether.

Share Improve this answer Follow

edited Sep 6, 2022 at 11:46
phane Chazelas
92 1029

answered Aug 15, 2011 at 16:10
Gilles 'SO- stop being evil'
818k 195 1696 2189

a newline character (n1256 §5.1.1.1)" --> In re-viewing a ther than maybe: "A source file that is not empty shall preceded by a backslash character before any such cted to splicing specifications. – chux - Reinstate Monica

must end with a newline character. Lines that are not the indicate that that line ends and the next line begins. SO- stop being evil' Aug 4, 2016 at 16:40

e." could be limited to how splicing considerations and Perhaps I'll look for a post that focuses on that.

>"So diff uses this special text \ No newline at end of file to differentiate a file that didn't end in a newline from a file that did." Git shows this text not only when it compares files. But even when new file added to git. So this argument is not valid, I suppose. – Viktor Kruglikov Jan 12, 2018 at 15:54

3 Why not just update the archaic tools instead so they don't break if there's a missing new line... – Andrew May 11, 2019 at 14:27

79

Not necessarily the reason, but a practical consequence of files not ending with a new line:

Consider what would happen if you wanted to process several files using `cat`. For instance, if you wanted to find the word `foo` at the start of the line across 3 files:

```
cat file1 file2 file3 | grep -e '^foo'
```

If the first line in file3 starts with `foo`, but file2 does not have a final `\n` after its last line, this occurrence would not be found by grep, because the last line in file2 and the first line in file3 would be seen by grep as a single line.

So, for consistence and in order to avoid surprises I try to keep my files always ending with a new line.

Share Improve this answer Follow

answered Aug 18, 2011 at 19:01

Sergio Acosta
1,104 7 12

3 But is it business of git to care about files concatenation? – Viktor Kruglikov Jan 15, 2018 at 9:33

3 Doesn't it stand to reason that you should just put `'\n'`'s in the cat operation... – Andrew May 11, 2019 at 14:28

10 That's like saying, "Sometimes I append Strings together that have `\n` or whitespace at the ends, so in order to keep things consistent, I always put `\n` _____ at both ends of my strings." Well, no, the right thing to do there is to have your Strings trimmed and then concatenate them properly. – Andrew May 11, 2019 at 14:29

4 @ViktorKruglikov It is git's business to care about your file content, yes. Git is there to help you keep your data consistent. If it were to ignore newlines it would fail in that purpose. In that same line of thought, it is cat's business to exactly reproduce the data you give it ; It must not add random characters inbetween that you did not specifically ask for. – zaTricky Sep 21, 2020 at 16:35

egrity of text files. If you're checking in an invalid text
021 at 14:34

e the last line if it does not end with a newline.
ith a newline (C11, 5.1.1.2, 2.) and that a last line
.2, 2nd item). Perhaps for historic reasons,
of the committee when the first standard was

) show line by line differences between files.
ds with a newline because else you would not
ce between two files is the presence of the last
the both files were the same, when `diff` and

`cmp` return an exit-code unequal success and the checksums of the files (e.g. via `md5sum`) don't match.

Share Improve this answer Follow

edited Sep 9, 2016 at 6:13

answered Aug 14, 2011 at 20:29

maxschlepzig
57k 51 209 281

make sense with diff program – Thamaraiselvam May 7, 2019 at 6:36

1 Sounds like diffs should just be smarter. – Andrew May 11, 2019 at 14:32

3 @Andrew, no, it doesn't. `diff` is expected to print differences if there are any. And if one file has a newline as last character while the other hasn't then that difference must be somehow noticeable in the output. – maxschlepzig May 11, 2019 at 14:42

Your latter statement is correct. However, the diff viewer does not have to display "newlines" (`\n`) to begin with, it can instead simply show "new lines". – Andrew May 11, 2019 at 14:45

14

The `\ No newline at end of file` you get from *github* appears at the end of a patch (in `diff` format, see the note at the end of the "Unified Format" section).

Compilers don't care whether there is a newline or not at the end of a file, but `git` (and the `diff`/`patch` utilities) have to take those in account. There are many reasons for that. For example, forgetting to add or to remove a newline at the end of a file would change its hashsum (`md5sum`/`sha1sum`). Also, files are not always programs, and a final `\n` might make some difference.

Note: About the warning from C compilers, I guess they insist for a final newline for backward compatibility purposes. Very old compilers might not accept the last line if doesn't end with `\n` (or other system-dependent end-of-line char sequence).

Share Improve this answer Follow

edited Aug 14, 2011 at 20:17

answered Aug 14, 2011 at 20:03

Stéphane Gimenez
28.7k 3 76 87

10 *"I guess they insist for a final newline for backward compatibility purposes"* - Nope, they insist on it because the C standard *mandates* it. – MestreLion Aug 28, 2013 at 9:25

C11 §5.1.1.2 2). Note that for *text* file I/O, C has cter is implementation-defined." §7.21.2 2

v May 11, 2019 at 14:31

tes it… – Stéphane Gimenez Aug 1, 2019 at 10:11

nteroperability among different OSes (POSIX also 7

ds without a newline character, then adding es as changing that last line (because `\n` is

n as `git blame` and `hg annotate`.

Share Improve this answer Follow

answered Sep 9, 2015 at 18:25

Hosam Aly

290 2 6

Sounds like diffs just need to be smarter. – Andrew May 11, 2019 at 14:32

2 The diffing tools are being smart. They notice the subtle change to the file (which is important because it will inevitably change the file's hash). And both GNU diff and git diff accept a `-w` option to ignore whitespace changes when outputting data for humans. – joeytwiddle Oct 16, 2019 at 3:22

@andrew Smart enough to follow any ad-hoc nonstandard text? Where to end with that? Maybe ending text files with a newline was a bad standard, but is it really a good idea to change that now? More things than `diff` would be affected. – Jeff Learman Dec 7, 2021 at 14:39

8

POSIX, this is a set of standards specified by IEEE to maintain compatibility between operating systems.

One of which is the definition of a "line" being a sequence of zero or more non- characters plus a terminating newline character.

So for that last line to be recognised as an actual "line" it should have a terminating new line character.

This is important if you depend on OS tools to say line count or split / help parse your file. Given PHP is a script language, its entirely possible especially in its early days or even now (I have no idea / postulating) it had OS dependencies like that.

In reality, most operating systems are not fully POSIX compliant and humans are not that machine like or even caring about terminating new lines. So for most things its a smorgasbord of everything either caring about it, warning or just going that last bit of text is really a line so just include it.

Share Improve this answer Follow

answered Nov 15, 2017 at 13:26

user3379747

81 1 1

Here is an additional reason. Say you have a file `file.txt` containing a list of names, with one name per line (or consider a file such as a .gitignore file).

e:

line (no matter if your file is empty or not).

you will end up with:

```
AliceJohn
<new_line_here>
```

which is definitely not what you expected. So having **all** text files terminated with a EOL makes life much easier.

Share Improve this answer Follow

edited Sep 6, 2022 at 10:41

answered Aug 12, 2021 at 10:34

Chevdor

161 1 3