Why is it recommended to have empty line in the end of file?

Asked 9 years, 6 months ago Active 1 month ago Viewed 32k times

194

Some code style tools recommend this and I remember seeing some unix command line tools warning about missing empty line.

What is the reasoning for having an extra empty line?

language-agnostic coding-style eof

26

asked Feb 18 '10 at 10:52

Petteri Hietavirta
6,142 11 56 92

6 Some tools fail to work if the file doesn't end with a newline. That is different than having an empty line at the end (which would be 2 newlines). – William Pursell Feb 18 '10 at 12:02

The text editors Gedit and Nano (and reportedly Vim) will append an empty line to any documents you save. – Colonel Panic Sep 14 '12 at 13:48

2 Do you mean empty line (`\n\n`) or new line `\n` ? – Ciro Santilli 新疆改造中心996ICU六四事件 Sep 13 '14 at 20:10

12 `cat` the file on a shell and you'll know why. If your file makes my shell's prompt appear in any other place than the one it should be (at the beginning of the line) I will probably hate you. ;) – ThiefMaster Sep 25 '14 at 16:21

1 Came across this old question and simply cannot believe that every single answer tries to justify the failings and shortcomings of other tools and systems by saying that modern coders should add a character that has no value in the code itself. Talk about 5 monkeys in a cage! :-D – Amos M. Carpenter Jan 21 at 8:01

8 Answers

158

Many older tools misbehave if the last line of data in a text file is not terminated with a newline or carriage return / new line combination. They ignore that line as it is terminated with ^Z (eof) instead.

answered Feb 18 '10 at 10:55

Ralph M. Rickenbach
10.9k 5 24 46

Thanks for the answer! Any examples of popular tools that might exhibit this behavior? – Nick Merrill Aug 1 '16 at 4:44

6 @NickM Almost all POSIX/Unix command-line tools that take text input or read a text file assume a line ending (`\n`) at end of file. Several text editors, like Vim, and several compilers (notably C++ and Python) will issue warnings. (In C++'s case, the standard explicitly requires this.) – greyfade Nov 26 '16 at 8:08

2 So what you're saying is ... it's a cargo cult – Jaykul Jun 14 at 18:33

42

If you try to concatenate two text files together, you will be much happier if the first one ends with a newline character.

answered Apr 21 '14 at 17:44

user1809090
986 1 8 6

34

Apart from the fact that it is a nicer cursor position when you move to the end of a file in a text editor.

Having a newline at the end of the file provides a simple check that the file has not been truncated.

answered Feb 18 '10 at 10:56

rsp
20.8k 4 47 61

191 The file might be truncated and you would never even kn – Simon Nickerson Feb 18 '10 at 10:57

6 First part is correct, second part is not. – Aniket Suryavanshi Jun 29 '17 at 10:52

16

An argument can also be made for cleaner diffs if you append to the file following the same reasoning as Why are trailing commas allowed in a list?

The following is copied (and trimmed a bit) from the linked resource:

Changing:

```
s = [
  'manny',
  'jack',
]
```

to:

```
s = [
  'manny',
  'jack',
  'roger',
]
```

involves only a one-line change in the diff:

```
  s = [
    'manny',
    'jack',
+   'roger',
  ]
```

This beats the more confusing multi-line diff when the trailing comma was omitted:

```
  s = [
    'manny',
-   'jack'
+   'jack',
+   'roger'
  ]
```

edited Aug 2 at 12:20

answered Oct 21 '15 at 8:31

Mathias Bak

2,347 3 22 33

Link-only answers aren't considered valuable on SO. Please copy the relevant information here while maintaining attribution. – isherwood Jul 25 at 15:38

15

The empty line in the end of file appears so that standard reading from the input stream will know when to terminate the read, usually returns EOF to indicate that you have reached the end. The majority of languages can handle the EOF marker. It is there for that reason from the old days, under DOS, the EOF marker was F6 key or Ctrl-Z, for *nix systems, it was Ctrl-D.

Most, if not all, will actually read right up to the EOF marker so that the runtime library's function of reading from input will know when to stop reading any further. When you open the stream for Append mode, it will wipe the EOF marker and write past it, until a close is explicitly called in which it will insert the EOF marker at that point.

Older tools were expecting a empty line followed by EOF marker. Nowadays, tools can handle the empty line and ignore it.

edited Aug 1 '13 at 15:06

answered Feb 18 '10 at 11:07

t0mm13b
30.1k 6 63 98

5 ^D was not "the EOF marker". Pressing ^D caused the shell to close the write side of the pipe that the foreground process group was reading from, so that a read from that pipe returned EOF. There is no "EOF marker". – William Pursell Feb 18 '10 at 11:13

@William Pursell You mistakenly conflated *NIX and Windows. Legacy Windows/DOS absolutely used an EOF marker (26, 0x1a) embedded usually at the end of most files as a holdover for compatibility with ancient CP/M (Who the heck used CP/M after 1983?). Other "fun": `\r\n` instead of `\n`, DOS calls using a mix of ASCIIZ and ASCII$. Even worse, later on Windows usually insert an Unicode byte order mark (BOM) at the beginning of most text files. Lovely "uniqueness." – user246672 Oct 18 '17 at 5:19

8

Also when you modify file and appends some code at the end of file - diff (at least git diff in standard coniguration) will show that you changed the last line, while the only thing you've actually done - added a newline symbol. So cvs reports become less convenient.

answered May 14 '13 at 14:23

prijutme4ty
91 1 1

4

Some languages define their input file in terms of input lines, where each input line is a series of characters terminated by a carriage return. If their grammar is so defined, then the last valid line of the file must be terminated by a carriage return too.

answered Feb 18 '10 at 11:07

Damien_The_Unbeliever
202k 18 265 356

1

It's because of the definition of what a text file is. When you create a new text file in any unix environment, the contents of that file is the new line character '\n'

Without this, the file isn't really identified as a text file. Now once we add code to this text file, its about not removing this initial new line that defines a text file itself.

answered Oct 11 '18 at 13:35

Victor Fernandes
75 10