Difference between CR LF, LF and CR line break types?

Asked 9 years, 11 months ago Active 1 year, 5 months ago Viewed 981k times

649

244

I'd like to know the difference (with examples if possible) between CR LF (Windows), LF (Unix) and CR (Macintosh) line break types.

line-breaks

edited Aug 14 '17 at 6:14

Gallifreyan
212 5 10

asked Oct 12 '09 at 4:47

3zzy
28.2k 87 233 374

8 Very similar, but not an *exact* duplicate. `\n` is typically represented by a linefeed, but it's not necessarily a linefeed. – Adrian McCarthy Mar 2 '12 at 0:00

80 CR and LF are ASCII and Unicode control characters while `\r` and `\n` are abstractions used in certain programming languages. Closing this question glosses over fundamental differences between the questions and perpetuates misinformation. – Adrian McCarthy Oct 8 '12 at 20:07

5 @AdrianMcCarthy It's a problem with the way close votes act as answers in a way; an answer claiming the two were the same could be downvoted and then greyed out as very, very wrong, but it only takes 4 agreeing votes (comparable to upvotes) to have a very wrong close happen, with no way to counter the vote until after it's happened. – Jon Hanna Mar 13 '14 at 17:03

This formulation of the question is admittedly better, but it is still for all practical purposes the same question. – Jukka K. Korpela Mar 13 '14 at 17:22

5 @JukkaK.Korpela: No, it really isn't. `\n` doesn't mean the same thing in all programming languages. – Adrian McCarthy Mar 13 '14 at 17:34

9 Answers

It's really just about which bytes are stored in a file. `CR` is a bytecode for carriage return (from the days of typewriters) and `LF` similarly,

answered Oct 12 '09 at 4:52

Peter

88.4k 41 163 202

30 I think it's also useful to mention that `CR` is the escape character `\r` and `LF` is the escape character `\n` . In addition, Wikipedia:Newline. – Robert Vunabandi Sep 3 '18 at 15:20

In Simple words `CR and LF` is just end of line and new line according to this link , is this correct ? – stom Sep 18 at 11:02

671

CR and LF are control characters, respectively coded `0x0D` (13 decimal) and `0x0A` (10 decimal).

They are used to mark a line break in a text file. As you indicated, Windows uses two characters the CR LF sequence; Unix only uses LF and the old MacOS (pre-OSX MacIntosh) used CR.

An apocryphal historical perspective:

As indicated by Peter, CR = **Carriage Return** and LF = **Line Feed**, two expressions have their roots in the old typewriters / TTY. LF moved the paper up (but kept the horizontal position identical) and CR brought back the "carriage" so that the next character typed would be at the leftmost position on the paper (but on the same line). CR+LF was doing both, i.e. preparing to type a new line. As time went by the physical semantics of the codes were not applicable, and as memory and floppy disk space were at a premium, some OS designers decided to only use one of the characters, they just didn't communicate very well with one another ;-)

Most modern text editors and text-oriented applications offer options/settings etc. that allow the automatic detection of the file's end-of-line convention and to display it accordingly.

edited Apr 20 '18 at 15:40

Steven M. Vascellaro

7,591 12 55 118

answered Oct 12 '09 at 4:52

mjv

61.6k 12 92 144

3 so actually Windows is the only OS that uses these characters properly, Carriage Return, followed by a Line Feed. – Rolf Sep 3 '18 at 10:38

4 Would it be accurate, then, to say that a text file created on Windows is the most compatible of the three i.e. the most likely to display on all three OS subsets? – Hashim Sep 19 '18 at 19:12

2 @Hashim it might display properly but trying to run a textual shell script with carriage returns will usually result in an error – Omer Apr 24 at 14:06

425

This is a good summary I found:

The Carriage Return (CR) character (`0x0D` , `\r`) moves the cursor to the beginning of the line without advancing to the next line. This character is used as a new line character in Commodore and Early Macintosh operating systems (OS-9 and earlier).

The Line Feed (LF) character (`0x0A` , `\n`) moves the cursor down to the next line without returning to the beginning of the line. This character is used as a new line character in UNIX based systems (Linux, Mac OSX, etc)

The End of Line (EOL) sequence (`0x0D 0x0A` , `\r\n`) is actually two ASCII characters, a combination of the CR and LF characters. It moves the cursor both down to the next line and to the beginning of that line. This character is used as a new line character in most other non-Unix operating systems including Microsoft Windows, Symbian OS and others.

Source

edited Mar 17 '18 at 13:05

Yves M.
20.4k 14 76 107

answered Oct 12 '09 at 4:54

Taylor Leese
37.5k 24 100 138

1 The "vertical tab"-character moves the cursor down and keep the position in the line, not the LF-character. The LF is EOL. – 12431234123412341234123 Nov 17 '16 at 12:07

1 @TaylorLeese Are /r/n and /n/r same? – Vicrobot Sep 22 '18 at 15:54

153

Since there's no answer stating just this, summarized succinctly:

Carriage Return (MAC pre-OSX)

- CR
- \r
- ASCII code 13

Line Feed (Linux, MAC OSX)

- LF

Carriage Return and Line Feed (Windows)

- CRLF
- \r\n
- ASCII code 13 and then ASCII code 10

If you see ASCII code in a strange format, they are merely the number 13 and 10 in a different radix/base, usually base 8 (octal) or base 16 (hexadecimal).

http://www.bluesock.org/~willg/dev/ascii.html

edited Feb 23 '18 at 19:49

answered Aug 31 '16 at 22:07

ahnbizcad

7,607 9 46 72

44

Jeff Atwood has a recent blog post about this: The Great Newline Schism

Here is the essence from Wikipedia:

> The sequence CR+LF was in common use on many early computer systems that had adopted teletype machines, typically an ASR33, as a console device, because this sequence was required to position those printers at the start of a new line. On these systems, text was often routinely composed to be compatible with these printers, since the concept of device drivers hiding such hardware details from the application was not yet well developed; applications had to talk directly to the teletype machine and follow its conventions. **The separation of the two functions concealed the fact that the print head could not return from the far right to the beginning of the next line in one-character time. That is why the sequence was always sent with the CR first. In fact, it was often necessary to send extra characters (extraneous CRs or NULs, which are ignored) to give the print head time to move to the left margin.** Even after teletypes were replaced by computer terminals with higher baud rates, many operating systems still supported automatic sending of these fill characters, for compatibility with cheaper terminals that required multiple character times to scroll the display.

edited Mar 13 '14 at 16:58

bergren2

44 2 6

answered Jan 20 '10 at 19:53

Manu

16.4k 26 69 82

any inktjet-printer (I love to understand since I hate to learn). My other memory-tricks are: "mac? Return to sender" and "NewLineFeed" (to remember that NL===LF and to remember the \n , since CR already has the R in it's abbreviation) – GitaarLAB Feb 1 '13 at 19:52

3 "I'm dubious ... two control codes was necessary for timing". That's not what it says. It says that the extra CRs and NULs are here for giving time for it to come back, not the original CR LF. – Julien Rousseau Sep 24 '14 at 5:58

10 @Adrian Will you take persona experience? 1) In my old teletype days, the printer we used required `<CR><CR><LF>` - so of course I experimented with just one `<CR>` . I sent `<CR><LF>A` after a long line, and you could *hear* the `A` being printed before the carriage fully returned. – John Burger Jun 11 '16 at 17:02

11 @Adrian 2) Don't forget, this was in the electro-mechanical era, where each character did exactly one function. We often emphasised a word by printing the line, then sending `<CR><CR>` and typing the correct number of spaces, then re-printing the same word: a primitive form of bolding. – John Burger Jun 11 '16 at 17:02

3 @Adrian 3) And finally, this was using Baudot (or Murray code), not ASCII. Five data bits, between one start bit and one-and-a-half stop bits. How can you have half a bit? By waiting half a bit time before starting to send the next character, to give the print head time to return to center. – John Burger Jun 11 '16 at 17:06

16

CR - ASCII code 13

LF - ASCII code 10.

Theoretically CR returns cursor to the first position (on the left). LF feeds one line moving cursor one line down. This is how in old days you controled printers and text-mode monitors. These characters are usually used to mark end of lines in text files. Different operating systems used different conventions. As you pointed out Windows uses CR/LF combination while pre-OSX Macs use just CR and so on.

edited Mar 13 '14 at 17:32

Adrian McCarthy
33.6k 10 87 140

answered Oct 12 '09 at 4:55

DmitryK
4,971 1 15 31

7

Systems based on ASCII or a compatible character set use either LF (Line feed, 0x0A, 10 in decimal) or CR (Carriage return, 0x0D, 13 in decimal) individually, or CR followed by LF (CR+LF, 0x0D 0x0A); These characters are based on printer commands: The line feed indicated that one line of paper should feed out of the printer, and a carriage return indicated that the printer carriage should return to the beginning of the current line.

pierrotlefou
20.9k 28 123 162

5

The sad state of "record separators" or "line terminators" is a legacy of the dark ages of computing.

Now, we take it for granted that anything we want to represent is in some way structured data and conforms to various abstractions that define lines, files, protocols, messages, markup, whatever.

But once upon a time this wasn't exactly true. Applications built-in control characters and device-specific processing. The brain-dead systems that required both CR and LF simply had no abstraction for record separators or line terminators. The CR was necessary in order to get the teletype or video display to return to column one and the LF (today, NL, same code) was necessary to get it to advance to the next line. I guess the idea of doing something other than dumping the raw data to the device was too complex.

Unix and Mac actually specified an *abstraction* for the line end, imagine that. Sadly, they specified different ones. (Unix, ahem, came first.) And naturally, they used a control code that was already "close" to S.O.P.

Since almost all of our operating software today is a descendent of Unix, Mac, or MS operating SW, we are stuck with the line ending confusion.

answered Oct 12 '09 at 5:10

DigitalRoss
127k 21 214 302

1

NL derived from EBCDIC NL = x'15' which would logically compare to CRLF x'odoa ascii... this becomes evident when physcally moving data from mainframes to midrange. Coloquially (as only arcane folks use ebcdic) NL has been equated with either CR or LF or CRLF

answered Jun 16 '17 at 0:45

david
11 1

protected by Josh Crozier Dec 27 '18 at 5:25

Would you like to answer one of these unanswered questions instead?