

## Re: getline and CR, LF, CR/LF, VS/Linux

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.stl/2005-07/msg00033.html>

---

- *From:* r norman <NotMyRealEmail@xxxxxxxxxxxx>
  - *Date:* Mon, 11 Jul 2005 17:44:07 -0400
- 

On Tue, 12 Jul 2005 06:44:37 +1000, Ian Semmel  
<isemmel@xx> wrote:

>  
>  
>r norman wrote:  
>>  
>> The question isn't really whether STL is to blame or not. It is how  
>> the operating system interprets what the C language calls '\n' as an  
>> element in files that are specified to be "text" (whatever that  
>> means). It isn't the STL that translates '\n' into either ASCII  
>> crlf sequences or into the single lf character.  
>>  
>>  
>  
>I think a 'text' file is fairly obvious.  
>  
>I don't care one way or the other. The only reason I used STL was because the  
>program is for Windows and Linux.

The notion of a binary file is, indeed, obvious. What goes into and comes out of a binary file is the byte stream exactly as specified (provided you are aware of packing density, byte order, internal representation of floating point values, alphabet sets, etc.) But the notion of a "text" file is far from obvious. It does NOT mean that the file only contains ASCII printing characters. Some operating systems put a special "end-of-file" marker at the end of a file (ctrl-Z = 0x1A is a common one) and some do not. If it does and, for some weird reason, the "text" file includes that character as data, the operating system will stop reading even though most of the file is still unread according to the byte count. Some systems automatically replace an ascii lf (0x0A) in the output stream with a crlf pair (0x0D-0x0A), reversing that process reading the file. That way, the number of characters contained in the file is not the same as the total number of characters in the strings you write to it. You run into all sorts of difficulties in file seeking to a specified byte count that way.

The interpretation of a raw cr (0x0D) varies by system. On many

devices (and this is the original concept), 0x0D was a simple return, to the first column on the same line. Using 0x0D allowed you to overprint old data. On those system, 0x0A was a simple line feed, to the same column on the next line without returning to the beginning of the line. On other devices, ones that emulate a manual typewriter, the "carriage return" function included both aspects: go to the first column of the next line. For data entry, it is traditional to use the "Enter" or "Return" key (which ordinarily generates the 0x0D character) to mean "this is the end of data entry for this line". That means execute a cr/lf sequence on the display. In C, '\r' is the 0x0d character in binary while '\n' is 0x0a. However, what they "mean" in terms of executing "text" input and output is highly variable.

This is an ancient problem and you can't blame Windows. Before then, MS-DOS and UNIX treated "text" files differently. And you can't even blame Microsoft, the authors of MS-DOS because CP/M also differed from Unix. And CP/M was, I believe, based on even earlier models, but that is getting before my time.

Everybody who has ever moved "text" files between Unix-Linus-xxxIX systems and CP/M - DOS - Windows systems has encountered this problem. It used to be even worse. Not only did you have to worry about what the operating system did, you had to also worry about what the display and printer devices did. They often had internal switches that controlled how they interpreted cr and lf bytes. It was not uncommon to find nicely formatted text on the screen appearing double spaced on the printer or, alternatively, with all the output scrolling out to the far right side of the printer, never returning to the beginning of the line.

---

• **References:**

- ◆ **getline and CR, LF, CR/LF, VS/Linux**  
◇ From: Ian Semmel
- ◆ **Re: getline and CR, LF, CR/LF, VS/Linux**  
◇ From: Gene Bushuyev
- ◆ **Re: getline and CR, LF, CR/LF, VS/Linux**  
◇ From: Ian Semmel
- ◆ **Re: getline and CR, LF, CR/LF, VS/Linux**  
◇ From: Bo Persson
- ◆ **Re: getline and CR, LF, CR/LF, VS/Linux**  
◇ From: Ian Semmel
- ◆ **Re: getline and CR, LF, CR/LF, VS/Linux**  
◇ From: r norman
- ◆ **Re: getline and CR, LF, CR/LF, VS/Linux**

Re: getline and CR, LF, CR/LF, VS/Linux

◇ *From:* Ian Semmel

- Prev by Date: ***Re: getline and CR, LF, CR/LF, VS/Linux***
- Next by Date: ***Re: getline and CR, LF, CR/LF, VS/Linux***
- Previous by thread: ***Re: getline and CR, LF, CR/LF, VS/Linux***
- Next by thread: ***Re: getline and CR, LF, CR/LF, VS/Linux***
- Index(es):
  - ◆ ***Date***
  - ◆ ***Thread***