

Re: EOF location?

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2006-05/msg00690.html>

- *From:* Clever Monkey <clvrnmnky.invalid@xxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 30 May 2006 11:30:03 -0400
-

Richard wrote:

Not such a Clever Monkey wrote:

What happens if we have two lines in a text file on some sort of DOS/Win32 machines:

```
17$ cat testfile.txt
18$ od -hc testfile.txt
19$ wc testfile.txt
20$ ls -l testfile.txt
```

That is hardly "some sort of DOS/Win32", it might be cygwin.

It might, but it is not. I was making the point that I was on a Win32 machine with POSIX utilities that allow us to inspect files. There are several ways to do this. Just trying to be clear, here. (By the way, even if I was running cygwin, I would consider this simply a programming environment and set of libraries running on "some sort of" Win32/DOS.)

I use the exact same POSIX utils (exact, as in from the same vendor) on the 390, as well. This has been a life-saver in the past!

(Win32 is pretty relaxed, in that the last CRLF pair can actually be omitted. In this case we have a "normalized" file).

'Win32' neither knows nor cares about any CR/LF especially whether there is one at the end of the file. There is no such thing as a 'normalized file', though there is a 'normalized file path' which is quite a different thing.

Re: EOF location?

What I said. This was just an aside. I used scare quotes around the word for a reason. The fact is that POSIX utilities *must* right out lines that end in the right EOL char(s). The OS imposes no requirements.

Each
line is 11 characters long; 9 characters and two line-end characters.
Note also that the size in bytes is the same as the character count.

Well duhhh.

I was replying in the same spirit of the OP (or the closest thing to an OP that my node is willing to give me) of "showing your work".

I assume it would depend on the file APIs in question, but it is typical for routines to return lengths that include all the characters, including the CRLF pair.

There are no Operating system disk file APIs in DOS or Windows that care about the CR/LF nor any that give a length that notices whether there are CR/LF or not.

Ok. I never implied otherwise. Some file APIs are able to return lines. They will return lines that include any line ends that are present, even if there are no line ends.

I was speaking to Pete's description of EOL char(s) and EOF on DOS-like systems. This is the only reason I mention line ends at all.

It is up to each language implementation or application to deal with whatever it wishes to do with any particular characters in the file.

Again, I did mention that my current example was anything more than an example, and I did make plenty of caveats. I was speaking to Pete's comment regarding EOF/EOL chars only. The other stuff was just background.

Thank you for expanding upon my comments, however.

The EOF/EOT marker is "present", but is usually used up by API calls that scan for it while retrieving the contents of a file.

No. Wrong. There is no requirement for an EOF to be present, it is indicated entirely by the file size. EOT is not used in files as any

Re: EOF location?

Re: EOF location?

sort of ending of the file.

Not true. Under some circumstances using these APIs to cycle through a text file until you get "EOF" will fail when confronted with a file that contains embedded EOF chars. It's a rare enough but common problem.

Actually, one of the responses to my reply described such a problem.

shrug I've seen it happen enough times, so I have actual experience with the problem.

This implies that those routines that count characters will ignore the EOF. Indeed, those routines will likely stop whatever they are doing immediately once they encounter the EOF and return with whatever they got so far.

That may or may not be true depending entirely on the routine itself. It is not something that is 'likely', some will always do so, some will never do so. Some may even have an option.

Agreed. This is something I took the time to make clear. My use of likely is meant to cover the cases where one routine, function or API works in one way, and another works quite different. That is, I did not intend the meaning to be "a function, routine or API may or may not handling EOF chars reliably depending on how it feels".

Applications that use the usual APIs to get at the contents of text files will normally never count or show the trailing EOF.

DOS/Win32 have no disk file APIs that will ever notice a Ctrl-Z nor care about it. If the file is full of EOF (Ctrl-Z) characters then dir will show how many there are.

I haven't used actual DOS in years, but I'm unclear on how "dir" would do this.

Whether a particular language implementation does this or not depends on the author. For example I just did a small C program that read a text file (fopen, getc) and counted the total characters and the x'1a' characters of a file that had several. It counted the whole file size and 12 x'1a' characters.

I agree this is a convention, and I am pretty POSIX oriented in this regard. I understand you can make a program to actually read in such chars. It is pretty common on Win32/DOS to refer to such files or file access as "binary" (not necessarily binary used in the sense of libc APIs that work with FILEs, but similar).

Re: EOF location?

Re: EOF location?

However, since we were talking about the notion of line ends I kept my discussion pretty much about so-called text files, where both EOL and EOF can matter quite a bit.

But your comments regarding the fact that it is the programmer or implementer that defines what these characters mean is well-taken and appreciated.

.