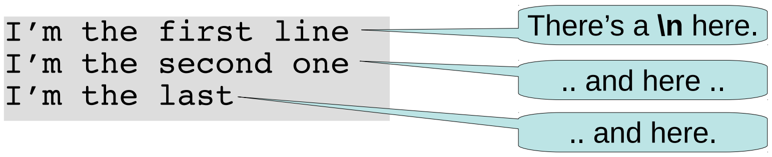
Line Termination

For text files, we normally expect every line to end with line termination. On a Unix system, this means we'll expect the newline character (**\n**, often called a linefeed) at the end of every line, including the last line. This sometimes causes confusion if you think of the newline character as starting a new line (admittedly, that's what its name makes it sound like). Instead, I'd suggest thinking of the newline character as marking the end of a line. So, in a text file like the following example, you'd expect a newline at the end of every line, including the last one:



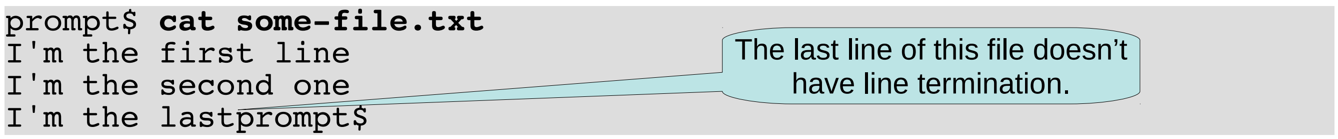
```
I’m the first line
I’m the second one
I’m the last
```

Depending on what text editor you're using, it may be difficult to tell whether the last line of your file ends with a newline. Some editors will automatically put a newline at the end of the last line, even if you don't enter one yourself. It looks like vim, emacs, gedit all do this, but notepad++, sublime text and TextWrangler don't.

If you have a text file (either a source file you wrote or maybe an output file from one of your programs), and you want to see if its last line has line termination, there are a few ways to do this. The easiest is probably just to use the **cat** command to show the contents of the file in the terminal. In the following example, you can see the next shell prompt gets printed on the next line, after the last line of the file. This shows that the last line must have ended with a newline character.

```
prompt$ cat some-file.txt
I'm the first line
I'm the second one
I'm the last
prompt$
```

The following example shows what would happen if the last line didn't have a line terminator. Here, the next shell prompt doesn't appear at the start of a line; it shows up over to the right, at the end of the last line in the file.



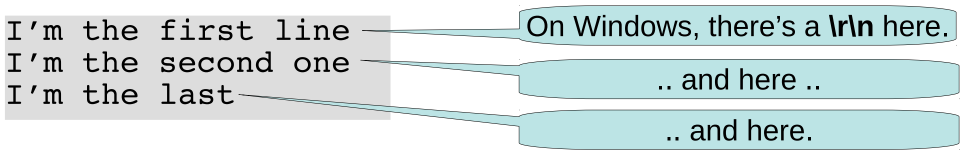
```
prompt$ cat some-file.txt
I'm the first line
I'm the second one
I'm the lastprompt$
```

You can also use the hexdump program to see exactly what characters are in a given file. When you run it with the -C option, it will show you all the ASCII codes for each character in your file over on the left. Over on the right, it will show the corresponding character symbols for each of these (or a dot for special characters or ones that aren't printable). In this example, you can see that the last line ends with the code, 0a, a newline.

```
hexdump -C some-file.txt
00000000  49 27 6d 20 74 68 65 20  66 69 72 73 74 20 6c 69  |I'm the first li|
00000010  6e 65 0a 49 27 6d 20 74  68 65 20 73 65 63 6f 6e  |ne.I'm the secon|
00000020  64 20 6f 6e 65 0a 49 27  6d 20 74 68 65 20 6c 61  |d one.I'm the la|
00000030  73 74 0a                                          |st.|
```

Windows vs Unix Line Termination

Unfortunately, different operating systems have different conventions for marking the end of a line in a text file. Unix-line systems (e.g., Linux or MacOS) use the newline character to mark the end of a line, but Windows systems use a two-character sequence, carriage return (ASCII code 13, **\r**) followed by newline (ASCII code 10, **\n**). So, on a Windows system, the same text might be saved differently in a file:



```
I’m the first line
I’m the second one
I’m the last
```

If you create text file on a Windows system or copy-and-paste text into a text editor on Windows, you can expect it to use the two-character Windows line termination sequence when you save your file.

Most text editors can handle either type of line termination, so looking at a file in your editor (on either type of system) may not tell you what type of line termination the file uses. The Unix **cat** command can help here. If you run it with the **-vT** options it will write out your file to the terminal window, using special sequences like **^M** to indicate a carriage return. In the example below, the **^M** at the end of every line shows that this file has Windows-style line termination.

```
cat -vT some-file.txt
I'm the first line^M
I'm the second one^M
I'm the last^M
```

Changing Line Termination

If you have a file with Windows-style line termination, there’s a simple Unix program that will convert all carriage return, linefeed sequences into just linefeeds. You can run it as follows, giving it just the name of the file you want to convert:

```
dos2unix some-file.txt
dos2unix: converting file some-file.txt to Unix format...
```