**C Escape Sequences[[1]](#footnote-1)**

Escape sequences are used to represent certain special characters within C string and character literals in order to print new line characters, special characters (such as the backspace), or allow for the printing of the double- or single-quote.

The following escape sequences are available on virtually all C compilers. There are some extra escape sequences that are vendor-specific, and thus, are not included in this reference.

|  |  |  |
| --- | --- | --- |
| **Escape sequence** | **Description** | **Representation** |
| **\'** | single quote | byte 0x27 in ASCII encoding (39 decimal) |
| **\"** | double quote | byte 0x22 in ASCII encoding (34 decimal) |
| **\?** | question mark | byte 0x3f in ASCII encoding (63 decimal) |
| **\\** | backslash | byte 0x5c in ASCII encoding (92 decimal) |
| **\a** | audible bell | byte 0x07 in ASCII encoding (7 in decimal) |
| **\b** | backspace | byte 0x08 in ASCII encoding (8 in decimal) |
| **\f** | form feed - new page | byte 0x0c in ASCII encoding (12 in decimal) |
| **\n** | line feed - new line | byte 0x0a in ASCII encoding (10 in decimal) |
| **\r** | carriage return | byte 0x0d in ASCII encoding (13 in decimal) |
| **\t** | horizontal tab | byte 0x09 in ASCII encoding (9 in decimal) |
| **\v** | vertical tab | byte 0x0b in ASCII encoding (11 in decimal) |
| **\nnn** | arbitrary octal value | byte nnn (Must be a valid octal value!) |
| **\xnn** | arbitrary hexadecimal value | byte nn (Must be a valid octal value!) |

**Notes**

1. Of the octal escape sequences, **\0** is the most useful because it represents the terminating null character for C strings.
2. The new-line character **\n** has special meaning when used in writing to a user console or to a text file: it is converted to the OS-specific newline representation, usually a byte or byte sequence. For example, Windows uses the <CR><LF> sequence to when the output should be on a different line. Conversely, Linux uses just the <LF> to move to the beginning of the next line on output.
3. Octal escape sequences have a limit of three octal digits, but terminate at the first character that is not a valid octal digit if encountered sooner.
4. Hexadecimal escape sequences have no length limit and terminate at the first character that is not a valid hexadecimal digit. If the value represented by a single hexadecimal escape sequence does not fit the range of values represented by the character type used in this string literal (char, char16\_t, char32\_t, or wchar\_t), the result is unspecified.
5. The question mark escape sequence \? is used to prevent trigraphs (see next page for explanation) from being interpreted inside string literals: a string such as "??/" is compiled as "**\"**, but if the second question mark is escaped, as in "?**\?**/", it becomes "??/”

**Trigraphs[[2]](#footnote-2)**

The following three-character groups (trigraphs) are parsed before comments and string literals are recognized, and each appearance of a trigraph is replaced by the corresponding primary character:

|  |  |
| --- | --- |
| **Primary** | **Trigraph** |
| { | ??< |
| } | ??> |
| [ | ??( |
| ] | ??) |
| # | ??= |
| \ | ??/ |
| ^ | ??' |
| | | ??! |
| ~ | ??- |

Because trigraphs are processed early, a comment such as // Will the next line be executed?????/ will effectively comment out the following line, and the string literal such as "Enter date ??/??/??" is parsed as "Enter date **\\**??".

1. Taken from <https://en.cppreference.com/w/cpp/language/escape> and edited [↑](#footnote-ref-1)
2. Removed in C17. Taken from cppreference.com and edited [↑](#footnote-ref-2)