The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones
derek@knosof.co.uk

6.10.4 Line control

Constraints

#line

The string literal of a **#line** directive, if present, shall be a character string literal. 1985

Commentary

This requirement that character string literals be used (i.e., no wide string literals) is more restrictive than that for the **#include** directive (which specifies implementation-defined handling of the sequence of characters).

#include q-char-sequence

Semantics

line number

The *line number* of the current source line is one greater than the number of new-line characters read or introduced in translation phase 1 (5.1.1.2) while processing the source file to the current token. 1986

Commentary

translation phase 1

This defines the term *line number*. Counting new-line characters read or introduced in translation phase 1 means that line splicing does not affect the line number. The first line of the source file has a line number of 1.

#line digit-sequence

A preprocessing directive of the form 1987

```
# line digit-sequence new-line
```

causes the implementation to behave as if the following sequence of source lines begins with a source line that has a line number as specified by the digit sequence (interpreted as a decimal integer).

Commentary

Developers point of reference for diagnostic messages is invariably the contents of the untranslated source file. The **#line** directive can simplify the implementations of tools that modify this source during translation, such as a preprocessor. Using this directive removes the need for them (in those cases where a mapping back to the original line number is desirable) to ensure that modifications to the source maintain line number information.

Common Implementations

Some implementations supported the following as an equivalent form:

```
# digit-sequence new-line
```

In some cases this token sequence is generated by translation phase 4 to provide line number information to subsequent phases of translation (to enable them to provide accurate line number information in any diagnostics issued).

The digit sequence shall not specify zero, nor a number greater than 2147483647. 1988

Commentary

preprocessor directives syntax

The syntax of the directive does not permit a minus sign to appear before the digits.

C90

The limit specified in the C90 Standard was 32767.

C++

Like C90, the limit specified in the C++ Standard is 32767.

Common Implementations

Many existing C90, and C++, implementations supported the C99 value.

1989 A preprocessing directive of the form

#line
digit-sequence
s-char-sequence

```
# line digit-sequence "s-char-sequence_opt" new-line
```

sets the presumed line number similarly and changes the presumed name of the source file to be the contents of the character string literal.

Commentary

Hiding the implementation details about the name of the file containing generated, or preprocessed, source can have a variety of advantages (e.g., the contents of a **#include**d file may refer to the name of the source file from which it was generated). This option provides such functionality.

Common Implementations

Some early implementations supported the following as an equivalent form:

```
# digit-sequence "s-char-sequence_opt" new-line
```

1990 A preprocessing directive of the form

```
# line pp-tokens new-line
```

(that does not match one of the two previous forms) is permitted.

Commentary

This form provides some flexibility in allowing the line number and source filename to be specified via macro replacement.

1991 The preprocessing tokens after **line** on the directive are processed just as in normal text (each identifier currently defined as a macro name is replaced by its replacement list of preprocessing tokens).

#line
macros expanded

Commentary

To be exact, the preprocessing tokens after **line** on the directive up to the first new-line character are processed just as in normal text.

1992 The directive resulting after all replacements shall match one of the two previous forms and is then processed as appropriate.

Commentary

However, preprocessing tokens having the form of a constant expression are not evaluated to create an integer constant. For instance, the following will not cause the current line number to be incremented by 10.

```
1  #line __LINE__+10
```

C++

The C++ Standard uses different wording that has the same meaning.

16.4p5 *If the directive resulting after all replacements does not match one of the two previous forms, the behavior is undefined; otherwise, the result is processed as appropriate.*

References