Slay Some Earthly Demons XVIII: Remove undefined behavior if there are nonmatching single or double quotes.

Document: n3570

Author: Ryan Karl

Date: 2025-04-20

Changes: Remove undefined behavior if there are nonmatching single or double quotes.

Undefined Behavior: An unmatched ‘ or “ character is encountered on a logical source line during tokenization (J.2 (26), N3301). The editor should remove this from the Annex J.2 table.

Analysis: The existing sentence in section 6.4 “If a ‘ or a “ character matches the last category, the behavior is undefined” offers no guidance to implementors and can lead to inconsistent implementations. Unmatched quotes might slip through in some toolchains (e.g. pre-ANSI or legacy toolchains), or emit confusing error messages in others.

Recommendation: A clear rule should be added to the standard that requires every quote to form a valid character-constant or string-literal. This will formally align the C standard with virtually every compiler in widespread use. See the appendix for examples.

Suggested Rewording:

…
6.4 Lexical elements
Syntax
token:
keyword
identifier
constant
string-literal
punctuator
preprocessing-token:
header-name
identifier
pp-number
character-constant
string-literal
punctuator
each universal character name that cannot be one of the above
each non-white-space character that cannot be one of the above

Constraints
Each preprocessing token that is converted to a token shall have the lexical form of a keyword, an identifier, a constant, a string literal, or a punctuator. A single universal character name shall match one of the other preprocessing token categories. Each occurrence of the ‘ or “ character shall begin and end with a valid character-constant or string-literal, respectively, as defined in 6.4.4 and 6.4.5.

Semantics
A token is the minimal lexical element of the language in translation phases 7 and 8 (5.1.1.2). The categories of tokens are: keywords, identifiers, constants, string literals, and punctuators. A preprocessing token is the minimal lexical element of the language in translation phases 3 through 6. The categories of preprocessing tokens are: header names, identifiers, preprocessing numbers, character constants, string literals, punctuators, and both single universal character names as well as single non-white-space characters that do not lexically match the other preprocessing token categories.) ~~If a ‘ or a “ character matches the last category, the behavior is undefined.~~ Preprocessing tokens can be separated by white space; this consists of comments (described later), or white-space characters (space, horizontal tab, new-line, vertical tab, and form-feed), or both. As described in 6.10, in certain circumstances during translation phase 4, white space (or the absence thereof) serves as more than preprocessing token separation. White space may appear within a preprocessing token only as part of a header name or between the quotation characters in a character constant or string literal.
…

Acknowledgments: I thank the UB Study group for their helpful comments.

Appendix:
Consider the program below:

```c
/* example.c */

#include <stdio.h>

int main(void) {
    char c = 'A;            /* unmatched single-quote */
    char *s = "Hello, world;/* unmatched double-quote */
    (void)c;
    (void)s;
    return 0;
}
```

Compiling on gcc 6.1 we observe the following output:

```
<source>: In function 'main':
<source>:5:14: error: missing terminating ' character
     char c = 'A;            /* unmatched single-quote */
              ^
<source>:5:14: error: missing terminating ' character
     char c = 'A;            /* unmatched single-quote */
              ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
<source>:6:5: error: expected expression before 'char'
     char *s = "Hello, world;/* unmatched double-quote */
     ^~~~
<source>:6:15: error: missing terminating " character
     char *s = "Hello, world;/* unmatched double-quote */
```

```
              ^
<source>:6:15: error: missing terminating " character
     char *s = "Hello, world;/* unmatched double-quote */
               ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
<source>:8:11: error: 's' undeclared (first use in this function)
     (void)s;
           ^
<source>:8:11: note: each undeclared identifier is reported only once for
each function it appears in
Compiler returned: 1
```

Compiling on clang 3.5 we observe the following output:

```
<source>:5:14: error: missing terminating ' character [-Werror,-Winvalid-pp-
token]
    char c = 'A;            /* unmatched single-quote */
             ^
<source>:5:14: error: expected expression
<source>:6:15: error: missing terminating '"' character [-Werror,-Winvalid-
pp-token]
    char *s = "Hello, world;/* unmatched double-quote */
              ^
<source>:8:11: error: use of undeclared identifier 's'
    (void)s;
          ^
4 errors generated.
Compiler returned: 1
```

Compiling on TI C6x gcc 12.4.0 we observe the following output:

```
<source>: In function 'main':
<source>:5:14: error: missing terminating ' character
    5 |     char c = 'A;            /* unmatched single-quote */
      |              ^
<source>:5:14: error: missing terminating ' character
    5 |     char c = 'A;            /* unmatched single-quote */
      |              ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
<source>:6:5: error: expected expression before 'char'
    6 |     char *s = "Hello, world;/* unmatched double-quote */
      |     ^~~~
<source>:6:15: error: missing terminating " character
    6 |     char *s = "Hello, world;/* unmatched double-quote */
      |               ^
<source>:6:15: error: missing terminating " character
    6 |     char *s = "Hello, world;/* unmatched double-quote */
      |               ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
<source>:8:11: error: 's' undeclared (first use in this function)
    8 |     (void)s;
      |           ^
```

```
<source>:8:11: note: each undeclared identifier is reported only once for
each function it appears in
Compiler returned: 1
```

Compiling on icc 16.0.3 we observe the following output:

```
<source>(5): error: missing closing quote
      char c = 'A;            /* unmatched single-quote */
               ^

<source>(6): error: expected a ";"
      char *s = "Hello, world;/* unmatched double-quote */
      ^

<source>(6): error: missing closing quote
      char *s = "Hello, world;/* unmatched double-quote */
                ^

<source>(8): error: identifier "s" is undefined
      (void)s;
            ^

compilation aborted for <source> (code 2)
Compiler returned: 2
```

Compiling on msvc 16.1 we observe the following output:

```
example.c
<source>(5): error C2001: newline in constant
<source>(5): error C2015: too many characters in constant
<source>(6): error C2143: syntax error: missing ';' before 'type'
<source>(6): error C2001: newline in constant
<source>(7): error C2064: term does not evaluate to a function taking 259
arguments
<source>(7): error C2143: syntax error: missing ')' before 'type'
<source>(7): error C2059: syntax error: ')'
Compiler returned: 2
```