Remove undefined behavior for non-basic source characters in source files

Document: n3807

Author: Ryan Karl

Date: 2026-02-20

Changes: Replace the undefined behavior for non-basic source characters outside the listed exceptions with extensible behavior (based on working draft N3685, using the technique proposed in N3710).

Undefined Behavior: A character not in the basic source character set is encountered in a source file, except in an identifier, a character constant, a string literal, a header name, a comment, or a preprocessing token that is never converted to a token (J.2 (6), N3685). The editor should remove this from Annex J.2 if this behavior is reclassified as extensible behavior.

Analysis:
The existing 3rd paragraph in section 5.3.1 ends with: “If any other characters are encountered in a source file (except in an identifier, a character constant, a string literal, a header name, a comment, or a preprocessing token that is never converted to a token), the behavior is undefined.” This paper originally explored resolving this UB as either a constraint violation or implementation-defined behavior. After reviewing N3710, I now believe neither is the best fit for this case. This UB preserves extension space for implementations (and potentially future revisions of the standard) to recognize additional source characters outside the current basic source character set in contexts that are not otherwise covered. Reclassifying it as a constraint violation would foreclose that extension space, while reclassifying it as implementation-defined behavior would require the standard to bound and characterize an open-ended set of possible implementation responses.

Recommendation:
The standard should not reword this case as either a constraint violation or implementation-defined behavior. Instead, this paper should adopt the “extensible behavior” technique proposed in N3710 for UBs that primarily preserve implementation/future-standard extension space. Under that approach, an implementation either defines the behavior or issues a fatal diagnostic. This preserves extension latitude without leaving the case as undefined behavior. See the appendix for examples.

Suggested Rewording (relative to N3685; contingent on adoption of N3710 or equivalent wording defining extensible behavior)**:**

Revise Section 5.3.1, paragraph 3 to change the final sentence to:

“If any other characters are encountered in a source file (except in an identifier, a character constant, a string literal, a header name, a comment, or a preprocessing token that is never converted to a token), the behavior is ~~undefined~~ extensible.”

Dependency note (non-normative for this paper): N3710 proposes “extensible behavior” to cover cases intended to remain available for implementation extensions or future standardization, with the rule that an implementation either defines the behavior or issues a fatal diagnostic.

Acknowledgments: Thanks to the UB study group, David Svoboda, Dave Banham, and Joseph S. Meyers.

Appendix:

Consider the program below:

```c
/* stray.c */

#include <stdio.h>

int main(void) {
   int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
   return x;
}
```

Compiling this code with older releases of popular compilers demonstrates longstanding practices. For example, compiling on clang 3.5 we observe the following output:

```
<source>:6:15: error: non-ASCII characters are not allowed outside of
literals and identifiers
    int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
              ^~
<source>:6:14: error: expected ';' at end of declaration
    int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
             ^
             ;
2 errors generated.
Compiler returned: 1
```

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

```
<source>: In function 'main':
<source>:6:15: error: stray '\302' in program
    6 |     int x = 1 <U+00A9> 2;   /* U+00A9 COPYRIGHT SIGN in operator
position */
      |               ^~~~~~~~
<source>:6:17: error: expected ',' or ';' before numeric constant
    6 |     int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
      |                 ^
Compiler returned: 1
```

Compiling on icc 16.0.3 we observe the following output:

```
<source>(6): error: unrecognized token
      int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
                ^
```

```
<source>(6): error: expected a ";"
      int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
                ^

<source>(6): error: unrecognized token
      int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
                 ^

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

Compiling on msvc 16.1 we observe the following output:

```
example.c
<source>(6): error C3873: '0xa9': this character is not allowed as a first
character of an identifier
<source>(6): error C2146: syntax error: missing ';' before identifier 'c'
<source>(6): error C2065: 'c': undeclared identifier
<source>(6): error C2143: syntax error: missing ';' before 'constant'
Compiler returned: 2
```

Compiling on gcc 6.1 we observe the following output:

```
<source>: In function 'main':
<source>:6:15: error: stray '\302' in program
     int x = 1 ?? 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
               ^
<source>:6:16: error: stray '\251' in program
     int x = 1 ?? 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
                ^
<source>:6:18: error: expected ',' or ';' before numeric constant
     int x = 1 © 2;   /* U+00A9 COPYRIGHT SIGN in operator position */
                  ^
Compiler returned: 1
```