The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones
derek@knosof.co.uk

Copyright ©2002-2008 Derek M. Jones. All rights reserved.

3.4.3

undefined behavior

undefined behavior

behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this International Standard imposes no requirements

Commentary

conforming programs may depend on

The term *nonportable* is discussed elsewhere.

EXAMPLE constraint violation and undefined behavior

Although a sequence of source code may be an erroneous program construct, a translator is only required to issue a diagnostic message for a syntax violation or a constraint violation. The erroneous data can occur during translation, or during execution. For instance, division by zero within a constant expression or a division operator whose right operand had been assigned a zero value during the execution of a previous statement.

The C Standard does not contain any requirements for issuing diagnostics at execution time.

C90

Behavior, upon use of a nonportable or erroneous program construct or of erroneous data, or of indeterminately valued objects, for which this International Standard imposes no requirements.

indeterminate value trap representation reading it; undefined behavior unsigned char pure binary

Use of an indeterminate value need not result in undefined behavior. If the value is read from an object that has **`unsigned char`** type, the behavior is unspecified. This is because objects of type **`unsigned char`** are required to represent values using a notation that does not support a trap representation.

Common Implementations

While the C Standard may specify that use of a construct causes undefined behavior, developers may have expectations of behavior or be unaware of what the C Standard has to say. Implementation vendors face customer pressure to successfully translate existing code. For this reason diagnostic messages are not usually generated at translation time when a construct causing undefined behavior is encountered.

The following are some of the results commonly seen when executing constructs exhibiting undefined behavior:

- *A signal is raised.* For instance, `SIGFPE` on divide by zero, or `SIGSEGV` when dereferencing a pointer that does not refer to an object.
- *The defined behavior of the processor occurs.* For instance, two's complement modulo rules for signed integer overflow.
- *The machine code generated as a result of a translation time decision is executed.* For instance, `i = i++;` may have been translated to the machine code `LOAD i; STORE i; INC i;` (instead of `INC i; LOAD i; STORE i;`, or some other combination of instructions).

Coding Guidelines

Developers are often surprised to learn that some construct, which they believed to have well-defined behavior, actually has undefined behavior. They often have clear ideas in their own heads of what the implementation behavior is in these cases and consider that to be the behavior mandated by the standard. This is an issue of developers' education and is outside the scope of these coding guidelines (although vendors of static analysis tools may consider it worthwhile issuing a diagnostic for uses of such constructs).

Usage

Annex J.2 lists 190 undefined behaviors.

47 NOTE Possible undefined behavior ranges from ignoring the situation completely with unpredictable results, to behaving during translation or program execution in a documented manner characteristic of the environment (with or without the issuance of a diagnostic message), to terminating a translation or execution (with the issuance of a diagnostic message).

undefined behavior possible

Commentary

This is only a list of possible behaviors; it is not intended to be a complete list. Behaving in a documented manner, plus issuing a diagnostic, is often considered to be the ideal case.

Common Implementations

The two most common ways of handling a construct, whose behavior is undefined, are to issue a diagnostic and to ignore it completely (the translator continuing to translate, or the execution environment delivering whatever result happens to occur). Some translators provide options that allow the developer to select the extent to which a translator will attempt to diagnose these constructs (e.g., the *-Wall* option of `gcc`). Very few implementations document any of their handling of undefined behaviors.

Some undefined behaviors often give consistent results, e.g., signed integer overflow, while in other cases the behavior is understood but the results are completely unpredictable. For instance, when accessing an object after its lifetime has ended, the common behavior is to access the storage previously assigned to that object, but the value held in that location is unpredictable.

Most diagnostics issued during program execution (most implementations issue a few during translation) are as a result of the program violating some host requirement in some way (for instance, a misaligned access to storage) and the host issuing a diagnostic prior to terminating program execution.

Coding Guidelines

There are a some undefined behaviors that give consistent results on many processors, for instance, the result of a signed integer overflow. Making use of such behavior is equivalent to making use of representation information, which is covered by a guideline recommendation.

?? representation information using

48 EXAMPLE An example of undefined behavior is the behavior on integer overflow.

EXAMPLE undefined behavior

References