

The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones

derek@knosof.co.uk

5.1.1.3 Diagnostics

diagnostic
shall produce

A conforming implementation shall produce at least one diagnostic message (identified in an implementation-defined manner) if a preprocessing translation unit or translation unit contains a violation of any syntax rule or constraint, even if the behavior is also explicitly specified as undefined or implementation-defined.

146

Commentary

This is a requirement on the implementation. The first violation may put the translator into a state where there are further, cascading violations. The extent to which a translator can recover from a violation and continue to process the source in a meaningful way is a quality-of-implementation issue.

diagnostic
message

The standard says nothing about what constitutes a diagnostic message (usually simply called a *diagnostic*). Although each implementation is required to document how they identify such messages, playing a different tune to represent each constraint or syntax violation is one possibility. Such an implementation decision might be considered to have a low quality-of-implementation.

The Rationale uses the term *erroneous program* in the context of a program containing a syntax error or constraint violation. Developers discussing C often use the term *error* and *erroneous*, but the standard does not define these terms.

C++

1.4p2

- *If a program contains a violation of any diagnosable rule, a conforming implementation shall issue at least one diagnostic message, except that*
- *If a program contains a violation of a rule for which no diagnostic is required, this International Standard places no requirement on implementations with respect to that program.*

A program that contains “a violation of a rule for which no diagnostic is required”, for instance on line 1, followed by “a violation of any diagnosable rule”, for instance on line 2; a C++ translator is not required to issue a diagnostic message.

Other Languages

Java contains no requirement that any violations of its compile-time requirements be diagnosed.

Common Implementations

Traditionally C compilers have operated in a single pass over the source (or at least one complete pass for preprocessing and another complete pass for syntax and semantics, combined), with fairly localized error recovery.

Constraint violations during preprocessing can be difficult to localize because of the unstructured nature of what needs to be done. If there is a separate program for preprocessing, it will usually be necessary to remove all constraint violations detected during preprocessing. Once accepted by the preprocessor the resulting token stream can be syntactically and semantically analyzed. Constraint violations that occur because of semantic requirements tend not to result in further, cascading, violations.

Syntax violations usually causing a message of the form “Unexpected token”, or “One of the following . . . was expected” to be output. Recovering from syntax violations without causing some additional, cascading violations can be difficult.

implemen-
tation
single pass

translated
invalid program

Diagnostic messages need not be produced in other circumstances.⁸⁾

147

Commentary

The production of diagnostics in other circumstances is a quality-of-implementation issue. Implementations are free to produce any number of diagnostics for any reason, but they are not required to do so.

diagnostic
message
produced other
circumstances

Common Implementations

Over time the number of diagnostics produced by implementations has tended to increase. Minimalist issuance of diagnostics is not considered good translator practice.

Coding Guidelines

Some translators contain options to generate diagnostics for constructs that are they consider suspicious, or a possible mistake on the developer's part. Making use of such options is to be recommended as a way of helping to find potential problems in source code.

A guideline recommendation of the form "The translator shall be run in a configuration that maximizes the likelihood of it generating useful diagnostic messages." is outside the scope of these coding guidelines. A guideline recommendation of the form "The source code shall not cause the translator to generate any diagnostics, or there shall be rationale (in a source code comment) for why the code has not been modified to stop them occurring." is also outside the scope of these coding guidelines.

148 EXAMPLE An implementation shall issue a diagnostic for the translation unit:

```
char i;  
int i;
```

because in those cases where wording in this International Standard describes the behavior for a construct as being both a constraint error and resulting in undefined behavior, the constraint error shall be diagnosed.

Commentary

An implementation is not allowed to define undefined behavior to be: No diagnostic is issued.

EXAMPLE
constraint vio-
lation and unde-
fined behavior

References