

# **The New C Standard** (Excerpted material)

---

**An Economic and Cultural Commentary**

**Derek M. Jones**

derek@knosof.co.uk

### 5.1.2.2.2 Program execution

In a hosted environment, a program may use all the functions, macros, type definitions, and objects described in the library clause (clause 7). 178

#### Commentary

There are no library subsets; C is a single implementation conformance level standard. However, although functions of the specified name may be present to link against, in translation phase 8, the functionality provided by an implementation's library may vary (including doing nothing and effectively being optional).

Some library functionality was known to be difficult, if not impossible, to implement on certain host environments. In these cases the library functions have minimal required functionality; for instance, signal handling.

The C Standard, unlike POSIX, does not prohibit the use of functions, macros, type definitions, and objects from other standards, but such libraries must not change the behavior of any of the C-defined library functionality.

implemen-  
tation  
extensions

#### Other Languages

Cobol, SQL, and Pascal are multi-conformance level standards. They contain additional language constructs at each level. A program is defined to be at the level of the highest level construct it uses.

---

9) Thus, `int` can be replaced by a typedef name defined as `int`, or the type of `argv` can be written as `char ** argv`, and so on. 179

#### Commentary

The return type might also be specified as `signed int`. A typedef name in C is a synonym.

Some implementations choose to process a function called `main` differently from other function definitions (because of its special status as the function called on program startup). This wording makes it clear that the recognition of `main`, its return type, and arguments cannot be based on a textual match. Semantic analysis is required.

#### C++

The C++ Standard does not make this observation.

#### Coding Guidelines

Replacing one or more of these types by a typedef name suggests that the underlying type may change at some point. Gratuitous use of a typedef name is not worth a guideline recommending against it. Other uses are for a purpose and there is no reason for them to be the subject of a guideline.

footnote  
9

typedef  
is synonym

# References