-- --

IEEE P1003.2 Draft 11.2 – September 1991

B.1 C Language Definitions

B.1.1 POSIX Symbols

Certain symbols in this annex are defined in headers. Some of those headers could also define symbols other than those defined by this standard, potentially conflicting with symbols used by the application. Also, this standard defines symbols that other standards do not permit to appear in those headers without some control on the visibility of those symbols.

Symbols called *feature test macros* are used to control the visibility of symbols that might be included in a header. Implementations, future versions of this standard, and other standards may define additional feature test macros. The `#define`s for feature test macros shall appear in the application source code before any `#include` of a header where a symbol should be visible to some, but not all, applications. If the definition of the macro does not precede the `#include`, the result is undefined.

Feature test macros shall begin with the underscore character (_) and an uppercase letter, or with two underscore characters.

Implementations may add symbols to the headers shown in Table B-1, provided the identifiers for those symbols begin with the corresponding reserved prefixes in Table B-1. Similarly, implementations may add symbols to the headers in Table B-1 that end in the string indicated as a reserved suffix as long as the reserved suffix is in that part of the name considered significant by the implementation. This shall be in addition to any reservations made in the C Standard {7}.

After the last inclusion of a given header, an application may use any of the symbol classes reserved in Table B-1 for its own purposes, as long as the requirements in the note to Table B-1 are satisfied, noting that the symbol declared in the header may become inaccessible.

Future revisions of this standard, and other POSIX standards, are likely to use symbols in these same reserved spaces.

In addition, implementations may add members to a structure or union without controlling the visibility of those members with a feature test macro, as long as a user-defined macro with the same name cannot interfere with the correct interpretation of the program.

A conforming POSIX.2 application shall define the feature test macro in Table B-2. When an application includes a header and the _POSIX_C_SOURCE feature test macro is defined to be the value 1 or 2, the effect shall be the same as if _POSIX_SOURCE was defined as described in POSIX.1 {8}.

In addition, when the application includes any of the headers defined in this standard, and _POSIX_C_SOURCE is defined to be the value 2:

(1) All symbols defined in POSIX.2 to appear when the header is included shall be made visible.

Table B-1 – POSIX.2 Reserved Header Symbols

Header	Key	Reserved Prefix	Reserved Suffix
<fnmatch.h>	2	FNM_	
<glob.h>	1	gl_	
	2	GLOB_	
<limits.h>	1		_MAX
<regex.h>	1	re_	
	1	rm_	
	2	REG_	
<wordexp.h>	1	we_	
	2	WRDE_	

NOTE: The Key values are:

(1) Prefixes and suffixes of symbols that shall not be declared or `#define`d by the application.

(2) Prefixes and suffixes of symbols that shall be preceded in the application with a `#undef` of that symbol before any other use.

Table B-2 – _POSIX_C_SOURCE

Name	Description
_POSIX_C_SOURCE	Enable POSIX.1 {8} and POSIX.2 symbols; see text.

(2) Symbols that are explicitly permitted, but not required, by POSIX.2 to appear in the header (including those in reserved name spaces) may be made visible.

(3) Additional symbols shall not be made visible, unless controlled by another feature test macro.

The effect of defining the _POSIX_C_SOURCE macro to any other value is unspecified.

If there are no feature test macros present in a program, only the set of symbols defined by the C Standard {7} shall be present. For each feature test macro present, only the symbols specified by that feature test macro plus those of the C Standard {7} shall be defined when the header is included.

B.1.1.1 POSIX Symbols Rationale. *(This subclause is not a part of P1003.2)*

When the application defines the _POSIX_C_SOURCE feature test macro with value 2, it must be aware that all of the name space from POSIX.1 {8} and POSIX.2 has been reserved. This does not imply that a POSIX.2 implementation must support POSIX.1 {8}, just that the application must not conflict with an implementation that does. The application can check _POSIX_VERSION and _POSIX2_C_VERSION at compile time to see which standards are supported, if that is necessary. This is primarily an issue for the headers `<stdio.h>`, `<limits.h>`, `<locale.h>`, and `<unistd.h>`, since other POSIX.1 {8} names appear in other headers not mentioned in POSIX.2.

It is expected that C bindings to future POSIX standards and revisions will define new values for _POSIX_C_SOURCE, with each new value reserving the name space for that new standard or revision, plus all earlier POSIX standards. Using a single feature test macro for all standards rather than a separate macro for each standard furthers the goal of eventually combining all of the C bindings into one standard, which will be included in an international standard that refers to a language-independent ISO/IEC 9945-1 {8}.

B.1.2 Headers and Function Prototypes

Implementations shall declare function prototypes for all functions. Each function prototype shall appear in the header included in the synopsis of the function.

B.1.3 Error Numbers

Some of the functions in this annex use the variable *errno* to report errors. Such usage is documented in Errors in each specification. The usage of *errno* and the meanings of the symbolic names shall be as defined in POSIX.1 {8} B.1.3.

B.1.4 C Language Definitions Rationale. *(This subclause is not a part of P1003.2)*

This clause clarifies the interface to the C Standard {7}. The description was taken from POSIX.1, with one important modification. Since POSIX.1 {8} and the C Standard {7} were being developed and approved at about the same time, POSIX.1 {8} allowed ‘‘Common Usage C’’ implementations to give system vendors time to develop Standard C interfaces. Since Standard C compilers are now commonly available, POSIX.2 does not explicitly describe the binding to Common Usage C. However, such a binding would be straightforward, as long as the rules for Common Usage C in POSIX.1 are followed.