X3J11 Responses to Issues Raised During Third Public Review

Document Number X3J11/88-148

This document contains the official X3J11 responses for all issues raised in officially registered comments and unregistered letters received during the third public review of the Draft Proposed American National Standard for Information Systems — Programming Language C, X3.159-1989. Although these responses have been reviewed for accuracy and consistency, they do not constitute part of the proposed Standard. Please refer to the most recent editions of the Standard and Rationale documents to accurately ascertain their current contents. (Of necessity, the section, page, and line citations in *this* document always refer to the May 13, 1988 draft Standard and Rationale, not to the current drafts.)

The X3J11 Technical Committee believes that all issues raised by the correspondents have been adequately addressed, and will continue to assume that this is the case unless the author of a registered comment responds to the contrary to the X3 Secretariat within fifteen working days.

The comments received were helpful in many ways, including identifying errors in the draft Standard and Rationale, pointing out the need to clarify or reword certain sections, and leading to Committee discussions resulting in either improvements to the draft Standard and Rationale or reaffirmations of previous decisions. The X3J11 Committee appreciates the comments received, including those that did not lead to changes in the documents.

There were many reasons for not adopting some of the suggestions. Specific individual reasons are provided in this document, but the following general remarks may be helpful in understanding why not all good ideas could be adopted. As indicated in the Rationale, the Committee had to balance many conflicting requirements when determining what should and should not be specified in the Standard. Primary emphasis was placed on correcting technical errors and removing unacceptable restrictions imposed by the specifications. Although many suggestions for new inventions were interesting, X3J11 was in general not in a position to make any substantial change to the language at this point, unless it could be shown to address a severe deficiency.

The Committee has received numerous suggestions for enhancements, additions, and "inventions". In responding to such suggestions, we have been mindful of our obligation to standardize existing practice, rather than creating a new language. Many of these inventive suggestions appear to have real technical value, and it might be appropriate for implementors to experiment with them as locally-provided extensions.

One of the valuable properties of C is that it is a "small" language. When one says that "C is not Ada", or that "C is not PL/I", the comparison of *size* is a major factor. In many cases, an advocate of one single suggestion might justifiably say that adding "just this one feature" would add only marginally to the size of the C language. The problem before the Committee was that a succession of such additions would eventually make C a much larger language than was desired.

These judgements and tradeoffs might be more clearly appreciated by considering a selected list of inventive suggestions which were, in the end, rejected for inclusion in the C Standard. The Committee thanks the authors of the following items, and hopes they understand the difficulty of the tradeoffs involved in these choices.

Herewith, a partial list of inventive suggestions not adopted:

§2.1

- Replace the function `main` with a `main-program` keyword.
- Add a function that returns a pointer to command line.

§2.2

- Add new escapes for DEL, ESC, LF.
- Add `#defines` to specify bit-field ordering.
- Add a header with object size, alignment, *etc.*

§3

- Add parallel-processing capabilities.
- Add a method of specifying desired optimizations.

§3.1

- Allow nested functions.
- Add a `private` keyword.
- Provide semantics for `entry`.
- Allow trailing `$` and `%` in variable names.
- Provide a syntax for arbitrary names.
- Add a `complex` data type.
- Add a `short double` data type.
- Provide a method of specifying sizes of types.
- Add typing by example.
- Add additional types.
- Provide constants for `structs` and `unions`.
- Add binary constants.
- Provide nested comments.

§3.2

- Provide a `boolean` type.

§3.3

- Allow local variables scoped to an expression.
- Provide operator overloading.
- Add expression enhancements: `(n == ('A' || 'B'))`.
- Extend operators to types non-native to C.
- Use the `.` operator in place of `->`.
- Add an absolute offset operator.
- Add an `alignof` operator.
- Allow "address-of" to be applied to constants.
- Add a Pascal-like indirection operator.
- Add an array dimension operator.
- Add *cast-expression* ::= `(`*type-name*`)` *compound-statement*.
- Add *primary-expression* ::= `(` *compound-statement* `)`.

- Add a binary cast operator.
- Allow floating-point operands to `%`.
- Add a `^^` (logical exclusive-or) operator.
- Add *logical-op*= operators.
- Allow `=>` as well as `>=`.
- Get rid of = *vs.* == confusion.
- Add `?=` as a synonym for `==`.
- Allow Pascal's assignment operator as well as =.
- Add a new assignment operator which returns the pre-assignment value.
- Allow assignment of unlike structure types.
- Allow `<=` analogous to `+=`.
- Provide different semantics for the comma operator.

§3.4

- Provide aggregate constants.

§3.5

- Provide a simpler form of declarations.
- Allocate storage at specified addresses.
- Add `public` and `local` storage classes.
- Add distributed initialization.
- Allow `:0` to pad at the beginning of a word.
- Allow unnamed declarators for `struct` padding.
- Add bit-field extensions.
- Add automatic `typedef` for `struct` types.
- Provide more alignment control, *e.g.* `abut`.
- Allow sizes for `enum`s.
- Allow the `enum` increment to be specified.
- Add optional arguments to function prototypes.
- Provide multiple pointer types.
- Add type-generic functions.
- Provide semantics for `register` functions.
- Add a `typeof` operator.
- Add `typedef` *identifier* = *expression*`;`.
- Add anonymous `struct` and `union`.
- Add a keyword "`hidden`" as an alternative to anonymous structures.
- Add an `extract` operator for extracting substructures.
- Provide a way to initialize an arbitrary member of a `union`.

- Impute the type of a variable from the type of its initializer.
- Allow omissions in initializer lists.
- Add a `writeonly` type qualifier.
- Add a `noalias` type qualifier to aid optimization.
- Add `nonconst` and `nonvolatile` type qualifiers.

§3.6

- Add a Pascal-like `with` construct.
- Allow multiple entries to a function.
- Add `case` ranges.
- Allow declarations anywhere in a compound statement.
- Add `asm` blocks.
- Provide `case` relationals (such as `case <n:`).
- Provide `case` continuation (eliminate the "mutually exclusive" constraint).
- Make `while ( )` mean "infinite loop".
- Allow `do` without `while` for infinite loops.
- In a `switch` within a loop, make `break; break;` quit both the `switch` and the loop.

§3.7

- Allow nested function definitions.
- Allow soft (weak) entry points.
- Support "pure" code, *e.g.* `const` function.

§3.8

- Add new directives: `#note`, `#warning`, `#ask`, `#echo`, `#halt`, `#redef`.
- Add `#(` *constant-expr* `)`.
- Make `#define` more like an "editor substitute" power.
- Add `__STDP__` for indicating preprocessing conformance.
- Add `__FUNCTION__`, expanding to the current function name.
- Combine `__DATE__` and `__TIME__` to return a `struct tm`.

§4

- Add byte ordering, alignment, and sizing functions and/or rules.
- Add windowing functions.
- Provide support for generic objects.
- Provide a way to determine object alignment requirements.
- Enlarge the standard library to include a variety of useful functions.

§4.1

- Require `NULL` to be built in.
- Add `TRUE`, `FALSE`, and `EOS` macros.

§4.3

- Add `is_alpha`, `is_alnum`, and `isodigit`.

§4.5

- Add symbols for infinity and not-a-number to `<math.h>`.

§4.7

- Require "reliable signals".
- Define `SIGERR` for algorithmic errors.

§4.8

- Require variable arguments to be accessed through built-in facilities.

§4.9

- Provide ordered disk writes.
- Add I/O to the language.
- Add functions to handle directories.
- Add file name wild-card functions.
- Completely change the `printf` format syntax.
- Require the compiler to check `printf` formats.
- Add check protection.
- Require nonzero precision on `%i` to cause a decimal shift.
- Provide semantics for precision in `%c` conversions.
- Add format repetition.
- Allow `%f` to use "E" notation for very large numbers.
- Add a way to skip to the end of a line.
- Replace `vfprintf` with `%v` in `fprintf`.
- Add `vfscanf` *etc.*
- Add formatted I/O functions that invoke an application-supplied character I/O function.
- Require stream I/O functions to set `errno` upon error.
- Add *curses*-like functions.
- Add special console I/O functions.
- Make `fgets` return a pointer to the end of the buffer, not the beginning.
- Add "locate mode" I/O functions.
- Add a special type for `fseek` pointers.

§4.10

- Add new functions: `frand`, `lrand`, `sfrand`, and `slrand`.
- Add a `malloc`-like function with user-controlled alignment.
- Add a `getflags` function, to process command-line arguments to `main`.
- Provide an analog to the BCPL MULDIV function, which returns the quotient and remainder of $A*B/C$.
- `strtod` should convert "`e25`".
- Add `dtostr` *etc.*
- Provide pointer/text interconversion functions.

§4.11

- Add a function like `strncpy` with null termination guaranteed.
- Add a `strmove` function, which can handle overlapping strings.
- Add move string up/down routines, to handle overlapping moves efficiently.
- Add pattern-matching functions.

§4.12

- Add a new function: `ldifftime`.

Another category of suggestions that were not adopted includes requests for specifications that would conflict with existing practice on many systems, or with the constraints imposed by some important environments. In such cases, the Committee preferred to keep the Standard as widely useful as possible, to forestall the development of system-specific deviations from the Standard.

X3J11 Response to Letter L159

Correspondent's Name and Address:

P. K. Wolfe, Jr.
AT&T Bell Laboratories
Rm. 1A-443
200 Park Plaza
P.O. Box 3050
Naperville, IL 60566

Item 1 (§4.9.6, Doc. No. X3J11/88-089 Letter L159 p. 1)

Summary of Issue: Add `vscanf` *etc.*

Committee Response:
This was considered to be an invention of limited utility.

The `vscanf` family of routines was discussed at previous meetings. The Committee felt that the usefulness of these routines was far less than the usefulness of the `vprintf` family, hence the asymmetry.

X3J11 Response to Letter L160

Correspondent's Name and Address:

Paul L. Sinclair
Austec Inc
609 Deep Valley Drive
Rolling Hills Estates, CA 90274

Item 1 (§1.6, Doc. No. X3J11/88-095 Letter L160 p. 1)

Summary of Issue: Define "indeterminate".

Committee Response:
This proposed editorial change was discussed but not accepted.

"Indeterminate" has its common English meaning. The meaning of "indeterminate value" is clear by the context of its use. Also, in §1.6, the definition of "undefined behavior" prohibits the use of an indeterminate value in a strictly conforming program.

Item 2(a) (§2.1.1.2, Doc. No. X3J11/88-095 Letter L160 p. 1)

Summary of Issue: "Source file" is misused.

Committee Response:
This proposed editorial change was discussed but not accepted.

The term "source file" is defined in §2.1.1.1 of the Standard. It refers to a unit of the program text which may or may not coincide with an operating system's concept of a file. The various mappings performed during phases of translation can be thought of as editing the "source files" even though the files visible to the operating system remain intact.

Item 2(b) (§2.1.1.2, Doc. No. X3J11/88-095 Letter L160 p. 1)

Summary of Issue: When are non-escape source characters converted to the execution character set?

Committee Response:
This was accepted as an editorial change to the Standard.

The description of translation phase 5 has been changed to make it clear that this occurs during that phase.

Item 2(c) (§2.1.1.2, Doc. No. X3J11/88-095 Letter L160 p. 1)

Summary of Issue: Terminology for two types of "string literal" is confusing.

Committee Response:
This proposed editorial change was discussed but not accepted.

String literals in C have historically been called "character string literals"; the suggested rewording was not viewed as an improvement.

Item 3 (§2.1.2.3, Doc. No. X3J11/88-095 Letter L160 p. 1)

Summary of Issue: Required agreement with abstract semantics seems too strict for floating-point data.

Committee Response:
No change to the existing wording was considered necessary.

The descriptions of certain operations such as converting `float` to `double` specify the accuracy required. In these cases, the implementation must ensure that the contents of data files agree with

the abstract semantics' accuracy requirements. The Committee basically decided that it is more important that floating-point arithmetic be accurate than that it be easily optimized. However, the specified abstract semantics of floating-point arithmetic allow the optimizer some amount of latitude.

Item 4 (§3.1.3.4, Doc. No. X3J11/88-095 Letter L160 p. 1)

Summary of Issue: Why do octal and hexadecimal escapes have inconsistent syntax?

Committee Response:
This was considered a request for information, not an issue.

The hexadecimal escape sequences were added to allow specification of character constants in large character sets. Extending octal escape sequences to be like the hexadecimal ones would have broken a large body of existing code.

Item 5 (§3.1.9, Doc. No. X3J11/88-095 Letter L160 p. 2)

Summary of Issue: `/*` exceptions should include header name context.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

§3.1.7 states that the appearance of `/*` between the `<` and `>` or the " delimiters in a header name produces undefined behavior. No change was thought to be necessary in §3.1.9 as you suggest.

Item 6 (§4.3.1.9, Doc. No. X3J11/88-095 Letter L160 p. 2)

Summary of Issue: `isspace` should exclude `ispunct` as well as `isalnum`.

Committee Response:
The Standard must accommodate a variety of environments.

Some locales require alternate printing space characters in addition to `' '`, for which both `isspace` and `ispunct` should return nonzero; thus these categories are not necessarily mutually exclusive. Note that the behavior in the "C" locale is precisely defined.

Item 7 (§4.3.2.1, Doc. No. X3J11/88-095 Letter L160 p. 2)

Summary of Issue: The `tolower` **Returns** section should be reworded.

Committee Response:
This was accepted as an editorial change to the Standard.

§4.3.2.1 **Returns** has been modified to reflect your suggestion.

Item 8 (§4.3.2.2, Doc. No. X3J11/88-095 Letter L160 p. 2)

Summary of Issue: The `toupper` **Returns** section should be reworded.

Committee Response:
This was accepted as an editorial change to the Standard.

§4.3.2.2 **Returns** has been modified to reflect your suggestion.

X3J11 Response to Letter L161

Correspondent's Name and Address:

Eric McQueen
220 N 200 E #7
P.O. Box 159
Logan, UT 84321

Item 1 (§4.10.4.3, Doc. No. X3J11/88-096 Letter L161 p. 1)

Summary of Issue: Require UNIX-style interpretation of `exit` values.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

As the Rationale points out, there has never been a portable way of indicating unsuccessful termination. On UNIX systems, there are some nonzero arguments to `exit` that do not cause an unsuccessful termination indication (due to masking of the input argument to 8 bits). The Standard provides the `EXIT_FAILURE` macro which enables a portable program to indicate unsuccessful termination.

X3J11 Response to Letter L162

Correspondent's Name and Address:

Glenn Herteg
Machine Vision International
1117 W. Huron #103
Ann Arbor, MI 48103

Item 1.4(a) (§1.4, Doc. No. X3J11/88-097 Letter L162 p. 1)

Summary of Issue: Add vertical space before third- and fourth-level headings.

Committee Response:
This proposed editorial change was discussed but not accepted.

The Committee does not consider the current formatting to cause problems with readability.

Item 1.4(b) (§1.4, Doc. No. X3J11/88-097 Letter L162 p. 1)

Summary of Issue: Standard and Rationale should be published together as a unit.

Committee Response:
This concerns matters beyond the scope of X3J11.

Both the Standard and Rationale will be given to our parent committee (X3), and we will suggest this. However, how they eventually choose to publish them is beyond our control.

Item 1.8 (§R1.8, Doc. No. X3J11/88-097 Letter L162 p. 1)

Summary of Issue: Rationale should give hints on quality of implementation.

Committee Response:
It was decided to allow implementors freedom in this regard.

The Committee feels that it should not press for features that are purely matters of quality of implementation.

Item 2.1.1.2 (§2.1.1.2, Doc. No. X3J11/88-097 Letter L162 p. 2)

Summary of Issue: Add forward references.

Committee Response:
This proposed editorial change was discussed but not accepted.

§2 effectively references the entire Standard. See **Forward references** at the beginning of §2.

Item 2.2.4.2 (§2.2.4.2, Doc. No. X3J11/88-097 Letter L162 p. 2)

Summary of Issue: Rename `*SHRT_*` to `*SHORT_*`.

Committee Response:
This proposal conflicts with too much prior art.

These names were adopted from **/usr/group**. We must consider other groups before such changes can be made.

Item 3.1.2.5(a) (§3.1.2.5, Doc. No. X3J11/88-097 Letter L162 p. 3)

Summary of Issue: Use `Courier` font for keywords in syntactic category names.

Committee Response:
This proposed editorial change was discussed but not accepted.

This is not considered a problem.

Item 3.1.2.5(b) (§3.1.2.5, Doc. No. X3J11/88-097 Letter L162 p. 3)

Summary of Issue: Clarify the distinction between an unqualified and a non-qualified version of a type.

Committee Response:
The Committee believes this is clear enough as is.

A type is *unqualified* if its type category (formerly called "top type") has *no* type qualifiers applied. It is *appropriately non-qualified* in respect to a *specific* type qualifier when this qualifier is absent from those applied to its type category.

Note that the wording in this section has been improved for other reasons.

Item 3.1.7 (§R3.1.7, Doc. No. X3J11/88-097 Letter L162 p. 3)

Summary of Issue: Command name was omitted (typo).

Committee Response:
The request is reflected in the current draft.

Item 3.3.1 (§3.3.1, Doc. No. X3J11/88-097 Letter L162 p. 4)

Summary of Issue: Add *primary-expression* ::= (*compound-statement*) as in GNU CC.

Committee Response:
This is too radical a change to adopt at this stage.

There was no support for this proposal. It is not specific enough and is too inventive. The accompanying documentation, while a good description of the feature, was not nearly specific enough for a standard. For example, it does not specify the semantic constraints required to define the type and value of the compound statement, or the effect of a `return` statement within it.

Item 3.3.2.2(a) (§3.3.2.2, Doc. No. X3J11/88-097 Letter L162 p. 4)

Summary of Issue: Require explicit dereference for use of pointer to function.

Committee Response:
The Committee has voted against this idea.

This proposal is deemed contrary to the spirit of C. The feature that the proposal would change has historically been part of the language. It stems from the fact that one actually does call a function *via* a pointer to the function. Thus, the function name silently undergoes a conversion from type *function* to type *pointer to function*. The asterisk (dereference) operator converts this to a function, which is immediately silently reconverted to a pointer to function. The expression flips between these states until all asterisks are consumed, then the pointer to function is called. Any change to this would force massive changes in the Standard.

Item 3.3.2.2(b) (§3.3.2.2, Doc. No. X3J11/88-097 Letter L162 p. 4)

Summary of Issue: Implicit declaration of functions as returning `int` should be declared obsolescent.

Committee Response:
The Committee discussed this proposal but decided against it.

Your proposal was considered but voted against. It was deemed too radical a change to the language for too little benefit. The Committee has tried to refrain from dictating programming style.

Item 3.3.3.2 (§R3.3.3.2, Doc. No. X3J11/88-097 Letter L162 p. 5)

Summary of Issue: Explain how "data abstraction is enhanced" by allowing extra address and dereference operators to apply to objects.

Committee Response:
This proposed editorial change was discussed but not accepted.

This allows macros to be written using the "address of" operator that accept functions as well as objects. The Committee feels that this is sufficiently well explained.

Item 3.3.4 (§3.3.4, Doc. No. X3J11/88-097 Letter L162 p. 5)

Summary of Issue: Reword the "cast ... has no effect on type or value" sentence.

Committee Response:
Changes have been made along the lines you suggested.

Item 3.5.2(a) (§3.5.2, Doc. No. X3J11/88-097 Letter L162 p. 6)

Summary of Issue: Defaulting absence of type specifiers to mean `int` is deplorable.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

If the given example is intended as a global declaration, it is invalid because no declaration specifier follows the semicolon. This change from the K&R definition of the language is the current state of the Standard. If the example is intended as a local declaration, *i.e.* part of a compound statement in a function definition, it is valid only if it is the last declaration and if the identifiers after the semicolon are all previously declared; in this case, it becomes a comma expression which is indeed a valid statement. However, the Committee does not believe that this special case warrants the suggested change.

Item 3.5.2(b) (§R3.5.2, Doc. No. X3J11/88-097 Letter L162 p. 6)

Summary of Issue: `const` and `volatile` are type qualifiers, not type specifiers.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 3.5.2.3 (§R3.5.2.3, Doc. No. X3J11/88-097 Letter L162 p. 6)

Summary of Issue: Standard and Rationale have different section titles.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 3.5.3(a) (§R3.5, Doc. No. X3J11/88-097 Letter L162 p. 6)

Summary of Issue: Subsections were misnumbered.

Committee Response:
The request is reflected in the current draft.

Item 3.5.3(b) (§R3.5.3, Doc. No. X3J11/88-097 Letter L162 p. 7)

Summary of Issue: Fix wording about `const` function parameters.

Committee Response:
The request is reflected in the current draft.

The missing "If" has been added. The Committee feels the rest is correct as it stands.

Item 3.5.4.1 (§R3.5.4.1, Doc. No. X3J11/88-097 Letter L162 p. 7)

Summary of Issue: Missing "have" (typo).

Committee Response:
The request is reflected in the current draft.

Item 3.5.4.2(a) (§R3.5.4.2, Doc. No. X3J11/88-097 Letter L162 p. 7)

Summary of Issue: Remove trailing semicolon (typo).

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 3.5.4.2(b) (§R3.5.4.2, Doc. No. X3J11/88-097 Letter L162 p. 7)

Summary of Issue: Add omitted "`struct`" (typo).

Committee Response:
The request is reflected in the current draft.

Item 3.5.6 (§3.5.6, Doc. No. X3J11/88-097 Letter L162 p. 7)

Summary of Issue: Add `typedef` *identifier* = *expression* `;`, or reconsider adding `typeof`.

Committee Response:
This was considered to be an invention of limited utility.

Considering the rejection of compound statements as expressions (see comments on §3.3.1 above), this proposal is of limited utility and would be too radical a change.

Item 3.8.2 (§3.8.2, Doc. No. X3J11/88-097 Letter L162 p. 8)

Summary of Issue: Justify or fix specification for nested `#includes`.

Committee Response:
The request is reflected in the current draft.

The Rationale already explains this.

The Committee chose to make the search algorithm implementation-defined to accommodate those who wished to search in relation to the original source file being compiled as well as those who wished to search in relation to the file being included. An implementation could also choose to first look for the header in conjunction with the original top-level file, then (if not found) in conjunction with the current file containing the `#include`.

The phrase you cite in the Rationale does not appear in the May 1988 draft.

Item 3.9(a) (§3.9, Doc. No. X3J11/88-097 Letter L162 p. 9)

Summary of Issue: Omission of type specifier in a declaration should be declared obsolescent.

Committee Response:
This proposed editorial change was discussed but not accepted.

See our response to your comment 3.5.2(a) above.

Item 3.9(b) (§3.9, Doc. No. X3J11/88-097 Letter L162 p. 9)

Summary of Issue: Implicit declaration of function as returning `int` should be declared obsolescent.

Committee Response:
This proposed editorial change was discussed but not accepted.

See our response to your comment 3.3.2.2(b) above.

Item 4.1.2(a) (§R4.1.2, Doc. No. X3J11/88-097 Letter L162 p. 10)

Summary of Issue: Add "be" before "includable" (typo).

Committee Response:
The request is reflected in the current draft.

Item 4.1.2(b) (§4.1.2, Doc. No. X3J11/88-097 Letter L162 p. 10)

Summary of Issue: Reserve a portion of name space for library implementors.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

The Committee believes that the proposed solution is inadequate because it does not address the problem of conflicts among libraries provided by separate developers.

Item 4.1.3 (§4.1.3, Doc. No. X3J11/88-097 Letter L162 p. 10)

Summary of Issue: `<errno.h>` should define `ERROR`.

Committee Response:
The Committee discussed this proposal but decided against it.

There was no support for the proposal. The user can provide the requested macro if desired, using `#undef ERROR` or choosing another name (such as `ANERROR`) to avoid name space conflicts.

Item 4.1.5 (§4.1.5, Doc. No. X3J11/88-097 Letter L162 p. 10)

Summary of Issue: `<stddef.h>` should define `TRUE` and `FALSE`.

Committee Response:
The Committee discussed this proposal but decided against it.

This is only a subset of the infinite possibilities; the Committee tried to mandate only the most essential. Also, coding standards are beyond the scope of the Standard.

Item 4.9.4.4 (§4.9.4.4, Doc. No. X3J11/88-097 Letter L162 p. 13)

Summary of Issue: Rename `tmpnam` to `tmpname`.

Committee Response:
This proposal conflicts with too much prior art.

The name was adopted as part of the library base document.

Item 4.10.2 (§4.10.2, Doc. No. X3J11/88-097 Letter L162 p. 15)

Summary of Issue: Add `drand48` *etc.* (improved pseudo-random sequence generators).

Committee Response:
The Committee has voted against this idea.

The Committee has discussed this issue and decided that there are too many implementation-specific details surrounding predictable random number generators.

Item 4.10.4.2(a) (§R4.10.4.2, Doc. No. X3J11/88-097 Letter L162 p. 15)

Summary of Issue: Standard and Rationale disagree about whether an `atexit`-registered function is called upon abnormal termination.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The Standard is the only official specification document for C. In any ambiguity, the Standard provides the definitive answer. In this case, the Rationale only mentions "program exit". Since the Standard specifies *normal termination*, there is no ambiguity.

Item 4.10.4.2(b) (§4.10.4.2, Doc. No. X3J11/88-097 Letter L162 p. 15)

Summary of Issue: There should be a way to "pop the stack" of `atexit`-registered functions.

Committee Response:
The Committee discussed this proposal but decided against it.

The Committee rejected this proposal on the grounds that it was not sufficiently specific and that the semantics of such a function are extremely tricky. A user who requires the requested functionality can register a function with `atexit` and then use that function to maintain its own stack of other functions for execution at normal termination. That would also avoid interfering with use of `atexit` by other modules unknown to the application programmer, for example in a graphics support library.

Item 4.10.4.3 (§R4.10.4.3, Doc. No. X3J11/88-097 Letter L162 p. 15)

Summary of Issue: Change "have" to "has" (typo).

Committee Response:
This was accepted as an editorial change to the Rationale.

Item A.4 (§A.4, Doc. No. X3J11/88-097 Letter L162 p. 20)

Summary of Issue: Rename `*SHRT_*` to `*SHORT_*`.

Committee Response:
This proposal conflicts with too much prior art.

This is essentially the same as your comment concerning §2.2.4.2. See the remarks above.

Item A.6.4 (§A.6.4, Doc. No. X3J11/88-097 Letter L162 p. 21)

Summary of Issue: Uncapitalize word in section title.

Committee Response:
This was accepted as an editorial change to the Standard.

X3J11 Response to Letter L163

Correspondent's Name and Address:

Prescott K. Turner, Jr.
Software Development Technologies, Inc.
375 Dutton Road
Sudbury, MA 01776-2509

Item 1 (§R3.5, Doc. No. X3J11/88-098 Letter L163 p. 1)

Summary of Issue: Subsections were misnumbered.

Committee Response:
The request is reflected in the current draft.

Item 2 (§3.2.1.4, Doc. No. X3J11/88-098 Letter L163 p. 1)

Summary of Issue: Should round when converting `double` to `float`.

Committee Response:
The Standard must accommodate a variety of architectures.

To require rounding when converting from `double` to `float` would be a hardship for some existing implementations and architectures.

Item 3 (§3.2.1.5, Doc. No. X3J11/88-098 Letter L163 p. 1)

Summary of Issue: Casts ought to be able to reduce the precision or range.

Committee Response:
The Committee has voted for this idea.

The Standard already requires the behavior you desire. §3.3.4 requires that preceding an expression with a parenthesized type name converts the value of the expression to the named type. Furthermore, §3.2.1.2 specifies how conversions to smaller integers are performed and §3.2.1.4 specifies how conversions to smaller floating types are performed.

An editorial change has been made in §3.2.1.5 to clarify that the sentence you noticed applies only to floating types.

Item 4 (§R3.2.2.2, Doc. No. X3J11/88-098 Letter L163 p. 1)

Summary of Issue: Change description of `void` from "has no valid states" to "requires no storage"; remove "(nonexistent)".

Committee Response:
Changes have been made along the lines you suggested.

See our response to item 3 in your third public review letter "P14".

Item 5 (§3.3, Doc. No. X3J11/88-098 Letter L163 p. 1)

Summary of Issue: Applauds recent decision to honor grouping by parentheses.

Committee Response:
This was considered a comment rather than an issue.

Thank you.

Item 6 (§3.3.6, Doc. No. X3J11/88-098 Letter L163 p. 2)

Summary of Issue: Attempt to permit pointer to one past the last member of an array was botched.

Committee Response:
Changes have been made along the lines you suggested.

See our response to item 4 in your third public review letter "P14".

Item 7 (§3.3.9, Doc. No. X3J11/88-098 Letter L163 p. 2)

Summary of Issue: Specify explicit properties for equality operators.

Committee Response:
Changes have been made along the lines you suggested.

See our response to item 5 in your third public review letter "P14".

Item 8 (§3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 3)

Summary of Issue: The properties of `noalias` are wrong.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was removed between the second and third public reviews.

Item 9 (§3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 3)

Summary of Issue: `noalias` pointer parameters are wrong.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was removed between the second and third public reviews.

Item 10 (§3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 4)

Summary of Issue: `noalias` implementation license is too intricate.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was removed between the second and third public reviews.

Item 11 (§3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 4)

Summary of Issue: "Object declared by a `noalias` handle" doesn't make sense.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was removed between the second and third public reviews.

Item 12 (§3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 4)

Summary of Issue: Change "function" that reserves storage to "block".

Committee Response:
The Committee has made significant changes in this area.

`noalias` was removed between the second and third public reviews.

Item 13 (§3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 4)

Summary of Issue: Remove `noalias`.

Committee Response:
The request is reflected in the current draft.

`noalias` was removed between the second and third public reviews.

Item 14 (§R3.5.3, Doc. No. X3J11/88-098 Letter L163 p. 4)

Summary of Issue: `const` is *not* taken wholly from C++.

Committee Response:
This is a misinterpretation of correct wording in the document.

A data object may be initialized in an included file.

C++ does *not* require that an initialized `const` external exist only within a header.

Item 15 (§3.8, Doc. No. X3J11/88-098 Letter L163 p. 5)

Summary of Issue: Skipped-directive syntax needs clarification.

Committee Response:
This is a misinterpretation of correct wording in the document.

See our response to item 6 in your third public review letter "P14".

Item 16 (§3.8, Doc. No. X3J11/88-098 Letter L163 p. 5)

Summary of Issue: Preprocessing file syntax should apply to *each* source file.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

See our response to item 7 in your third public review letter "P14".

Item 17 (§3.8.1, Doc. No. X3J11/88-098 Letter L163 p. 6)

Summary of Issue: The definitions for `defined` are unnecessarily complex.

Committee Response:
The Committee chose a different approach to deal with this issue.

See our response to item 8 in your third public review letter "P14".

Item 18 (§3.8.1, Doc. No. X3J11/88-098 Letter L163 p. 6)

Summary of Issue: "Usual arithmetic conversions" does not apply to preprocessing.

Committee Response:
The Committee chose a different approach to deal with this issue.

This request is already reflected in the draft for the third public review. See page 85, lines 4-5 in the May 1988 draft.

Item 19 (§3.8.3.2, Doc. No. X3J11/88-098 Letter L163 p. 6)

Summary of Issue: Explain how \ works in the example.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

§3.8.3.2 states that \ is replaced by \\ in string literals and character constants. Since the argument in question is neither of these, no such replacement is performed.

Item 20 (§3.8.3.4, Doc. No. X3J11/88-098 Letter L163 p. 7)

Summary of Issue: The phrase "nested replacements" is imprecise.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

See our response to item 9 in your third public review letter "P14".

Item 21 (§4.5.2, Doc. No. X3J11/88-098 Letter L163 p. 7)

Summary of Issue: Range of inverse trig functions is overspecified.

Committee Response:
No change to the existing wording was considered necessary.

No change is needed, as the Standard is clear. The Standard only specifies which valid values the function *may* return, not the specific range that *must* be returned. This implies, for example, that the actual value returned for `acos(-1.0)` may be slightly less than the "true" (but unrepresentable) value of π; it is not permitted to exceed π. This strict guarantee is actually useful.

Item 22 (§4.11, Doc. No. X3J11/88-098 Letter L163 p. 10)

Summary of Issue: Clarify modification *via* pointer argument to a const-qualified type.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

Your intent is already covered in §3.5.3 in the second paragraph of **Semantics**: If an attempt is made to modify an object with a const-qualified type through use of an lvalue with non-const-qualified type, the behavior is undefined.

X3J11 Response to Letter L164

Correspondent's Name and Address:

Ted Toal
Software Science
20278 Scotts Flat Rd
Nevada City, CA 95959

Item 15 (§2.2.4.2, Doc. No. X3J11/88-099 Letter L164 p. 1)

Summary of Issue: An 8-bit processor need not use 4 bytes for a `long`.

Committee Response:
The Committee has voted against this idea.

The Committee decided that a `long` value must use at least 32 bits.

Item 23 (§3.1.2.4, Doc. No. X3J11/88-099 Letter L164 p. 1)

Summary of Issue: A `volatile` object should not be initialized.

Committee Response:
This is a misinterpretation of correct wording in the document.

Don't "define" such an object in C code. A volatile *object* should have an initial value. An I/O port would normally be accessed *via* a pointer and in such a case only the *pointer* would need initialization, not the location pointed to.

Alternatively, the volatile object could be declared **extern** from within your C code, with the "definition" occurring in assembly code or in the operating system.

Item 30(a) (§3.1.3.4, Doc. No. X3J11/88-099 Letter L164 p. 1)

Summary of Issue: Which is the "high-order" bit?

Committee Response:
Changes have been made along the lines you suggested.

This section has been reworded to make the intent clearer.

Item 30(b) (§3.1.3.4, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Is `'\0'` somehow preferred?

Committee Response:
This was considered a request for information, not an issue.

This is a matter of stylistic choice. The Standard makes it clear that `0` and `'\0'` are identical values. Many members of the Committee prefer the latter in character contexts.

Item 37 (§3.2.2.1, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Array usage should be consistent with function indirection behavior.

Committee Response:
The Committee has voted against this idea.

The situation with arrays is different.

```
char (*a)[4];
```

is a pointer to an array of 4 `chars`. It is *not* the same as a pointer to the first element (a `char`) of an array of 4 `chars`. `a++` increments a by 4 rather than 1.

Item 45(a) (§3.3.3.3, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Clarify the result of applying unary minus to an unsigned operand.

Committee Response:
This was considered a request for information, not an issue.

§3.1.2.5 makes this clear. Unary minus *can* be applied to unsigned items; the result has the promoted size and type.

Item 45(b) (§3.3.3.2, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: The footnote is too generous with E.

Committee Response:
The request is reflected in the current draft.

The footnote was changed between the second and third public reviews to require that the expression be a valid operand of the address-of operator.

Item 46 (§3.3.3.4, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Wants an operator like `sizeof` that yields the declared size of an array parameter rather than the pointer's size.

Committee Response:
The Committee has voted against this idea.

This is another instance of arrays not being first-class objects. You cannot pass arrays by value as parameters in C, and the size information is not available to the function in the general case.

Item 47 (§3.3.4, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Is the alignment of an unsigned integral type necessarily the same as the corresponding signed type?

Committee Response:
This was considered a request for information, not an issue.

Yes, the alignment requirements are the same. See §3.1.2.5, page 22, lines 40-42 of the May 1988 draft.

Item 49 (§3.3.8, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Clarify comparison of a signed integer with an unsigned one.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

This situation is covered by the usual arithmetic conversions.

Item 51(a) (§3.3.10, Doc. No. X3J11/88-099 Letter L164 p. 2)

Summary of Issue: Wants to apply bitwise-and to pointer as well as integral types.

Committee Response:
This could not be efficiently implemented on many architectures.

This operation may not have any meaning on some machines where pointers are not represented in a similar manner to integers. That is why the conversion of a pointer to an integer is implementation-defined.

You can, of course, apply such operators to the result of casting a pointer to an integral type, then convert back to a pointer; whether the operations are meaningful is implementation dependent.

Item 51(b) (§3.3, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Add logical exclusive-or.

Committee Response:
This was considered to be an invention of limited utility.

Note that both operands would have to be evaluated in order to determine the result of such an exclusive-or operator, so there could not be any "short-circuit" evaluation such as occurs for `&&` and `||` (which is what makes them especially useful). You can, if you wish, define your own logical exclusive-or macro as follows:

```
#define XOR(a,b) (!(a) != !(b))
```

Item 51(c) (§3.3.16, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Add *logical-op*= operators.

Committee Response:
This was considered to be an invention of limited utility.

This is pure invention, which the Committee tries to avoid unless it addresses a critical need.

Item 56(a) (§3.4, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Permit more general use of floating point in integral constant expressions.

Committee Response:
The Committee has voted against this idea.

The Committee decided that it did not want to require compile-time floating-point arithmetic in conforming compilers. That could be extremely difficult for cross-compilers.

Item 56(b) (§3.4, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Address constants can also be created by casting integers.

Committee Response:
In some cases, this proposal would be difficult to implement.

This is not portable since it would be meaningless on many machines where pointers are different from integers. See Rationale §3.3.4. We have, however, added words to permit implementations to accept such forms in environments where this is useful.

Item 59 (§3.5.1, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Allow `static` specifier on function declaration within a block.

Committee Response:
The Committee has voted against this idea.

The Committee chose a block scope model for identifiers. While some implementations in the past have allowed `extern` and `static` name scopes to extend beyond the block in which they were declared, we felt that the block scope model is cleaner and more understandable.

Item 61 (§3.5.2.1, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Allow `char` and `long` bit fields.

Committee Response:
This was considered to be an invention of limited utility.

The Committee discussed this and decided it was not needed since it provides no additional capabilities.

The storage layout for bit fields is extremely machine-dependent and difficult to specify in a portable fashion. What should happen when bit fields of different types are specified? The Committee decided against adding complexity in this area, especially since bit fields are not very portable anyway.

An implementation can provide other types of bit fields as an extension.

Item 62(a) (§3.5.2.2, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: Allow a trailing comma in an enumerator list.

Committee Response:
The Committee has voted against this idea.

Trailing comma in initializers is permitted because it is part of the base document (prior art used by existing programs), unlike **enums** where it was felt that this grammatical irregularity should not be propagated. It is also felt that machine-generated **enums** are less likely than machine-generated initializers.

Item 62(b) (§3.5.2.2, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: What value does one get for an enumeration constant when the maximum integer is exceeded?

Committee Response:
This was considered a request for information, not an issue.

This is a violation of the constraint that the value of an enumeration constant must be representable as an `int`. A conforming implementation is required to diagnose this.

Item 64 (§3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 3)

Summary of Issue: The description of type qualifiers is inadequate.

Committee Response:
The request is reflected in the current draft.

This had been completely reworded in the May 1988 draft.

Item 66 (§3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 4)

Summary of Issue: Specify compatibility when one type is qualified and the other is not.

Committee Response:
The Committee believes this is clear enough as is.

These types are not compatible. When used as the value of an expression, the qualified type is converted to the corresponding unqualified type; this allows them to be intermixed in expressions.

Item 67(a) (§3.5.4, Doc. No. X3J11/88-099 Letter L164 p. 4)

Summary of Issue: Allow pointer syntactic category after direct declarator, as in `int f()*`.

Committee Response:
This was considered to be an invention of limited utility.

This is pure invention, which the Committee tries to avoid unless it addresses a critical need.

Item 67(b) (§3.5.4, Doc. No. X3J11/88-099 Letter L164 p. 4)

Summary of Issue: Give meaning to some "invalid" but syntactically-allowed forms such as `int f()[]`.

Committee Response:
This was considered to be an invention of limited utility.

This is pure invention, which the Committee tries to avoid unless it addresses a critical need.

Item 69 (§3.5.4.3, Doc. No. X3J11/88-099 Letter L164 p. 4)

Summary of Issue: Rule stated for `T D1` is wrong.

Committee Response:
This is a misinterpretation of correct wording in the document.

The reductions must be applied recursively as specified by the declarator grammar. Consider the parse tree for this declaration and you will see that for the identifier `f` recursively applying the rules appearing in §3.5.4 produces:

`f` is an `int`,
by §3.5.4 (`D1` is `f`)
`f` is a function returning `int`,
by §3.5.4.3 (`D1` is `f( )`)
`f` is a function returning pointer to `int`,
by §3.5.4.1 (`D1` is `*f( )`)

Item 72 (§3.5.7, Doc. No. X3J11/88-099 Letter L164 p. 4)

Summary of Issue: Outlaw multiple initialization of internal objects.

Committee Response:
This is a misinterpretation of correct wording in the document.

This is already covered by the scope rules which prevent multiple declarations of local objects with the same name.

Item 78 (§3.6.4.2, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: What about code preceding any label within a switch body?

Committee Response:
This was considered a request for information, not an issue.

The code must meet all constraints, but is never executed since the semantics of `switch` force only code following the matching label, if any, to be executed.

Item 82 (§3.7, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: Can *internal* identifiers have multiple definitions?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

This is already covered by §3.1.2.2 on linkage (items with no linkage) and in the section on declarations, §3.5, page 55, lines 17-19 in the May 1988 draft.

Item 83 (§3.7.1, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: Convert array argument to pointer to array, not pointer to first member.

Committee Response:
The Standard reflects the base document in this regard.

See our comment on item 37 above.

Item 91(a) (§3.8.3, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: Allow an empty string for a macro argument.

Committee Response:
The Committee has voted against this idea.

The Committee choose to disallow this because it is not allowed for functions and it confuses the difference between a macro with no arguments and one (that has one parameter) with an empty argument list.

Item 91(b) (§3.8.3.2, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: Give a usage example for the # operator.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

Such an example already exists on page 90 of the May 1988 draft.

Item 95 (§3.8.8, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: Does `__FILE__` contain escapes when needed?

Committee Response:
This was considered a request for information, not an issue.

The precise contents of `__FILE__` are unspecified, but they must form a valid string literal token. The rest is a quality of implementation issue.

Item 98(a) (§4.1.5, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: Should suggest use of `size_t` for array indices.

Committee Response:
This concerns matters beyond the scope of X3J11.

Coding standards are beyond the scope of the Standard.

Item 98(b) (§4.1.5, Doc. No. X3J11/88-099 Letter L164 p. 5)

Summary of Issue: `<stddef.h>` should define more macros.

Committee Response:
This was considered to be an invention of limited utility.

This is only a subset of the infinite possibilities; the Committee tried to mandate only the most essential. Also, coding standards are beyond the scope of the Standard.

Item 106 (§4.4.1.1, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Don't want to link in code and data for unneeded locales.

Committee Response:
Quality of implementation is beyond the scope of the Standard.

Yes, you may indeed carry extra baggage under some implementations. However, this can be nicely handled by a clever implementation.

Item 124(a) (§4.9.1, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: `FOPEN_MAX` might depend on configuration.

Committee Response:
This was considered a comment rather than an issue.

Yes, it might. However, this is intended to indicate a limit for the library, not the operating system. The minimum system configuration needed to run conforming programs is outside the scope of this Standard and is left up to the implementor to specify.

Item 124(b) (§4.9.1, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Is `FILENAME_MAX` supposed to include room for a path prefix also, or just the name within a directory?

Committee Response:
This was considered a request for information, not an issue.

The form of file or path names is outside the scope of the Standard. Naturally (where this is an issue) an implementation should accommodate reasonable path prefixes, but this is a quality of implementation issue.

Item 125(a) (§4.9.2, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Help programmer create his own compatible streams.

Committee Response:
This was considered to be an invention of limited utility.

We would require a specific proposal before we could consider this.

Assuming that the intent is to allow standard I/O functions to be used with these streams, on POSIX implementations this can often be accomplished by obtaining a "file descriptor" handle then using `fdopen` to associate it with a standard I/O stream.

Item 125(b) (§4.9.2, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Extra null bytes being appended to a binary stream is not a good idea.

Committee Response:
This is a misinterpretation of correct wording in the document.

This was not for CP/M only; any file system that uses fixed-length records may need to do this. For example, DEC has a number of systems that use this kind of file.

Item 126 (§4.9.3, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Clarify stream buffering.

Committee Response:
The request is reflected in the current draft.

The wording in question was revised between the second and third public reviews.

Item 129 (§4.9.5.2, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Require semantics for `fflush` of input streams too.

Committee Response:
The Committee has voted against this idea.

The Committee decided against providing semantics for this case. Implementations are, however, free to add such semantics.

Item 130(a) (§4.9.5.3, Doc. No. X3J11/88-099 Letter L164 p. 6)

Summary of Issue: Allow a read operation to immediately follow a write operation on an update append-mode stream.

Committee Response:
The Committee has voted against this idea.

The Committee decided that an explicit file-positioning function must be called because we did not want to mandate an implementation such that the current file mode (read/write) must be checked only after determining the place to do the write. Some do it the other way (check mode, then determine where to write). Requiring this would also complicate some macros such as `putc`.

Item 130(b) (§4.9.5.3, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: What is the initial file position when opened for update?

Committee Response:
This was considered a request for information, not an issue.

The Standard in §4.9.3 states that it is implementation-defined when a file is opened with append mode whether the file position indicator is positioned at the beginning or the end of the file.

Item 131 (§4.9.5.6, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: The effect of `_IOLBF` on an *input* stream is not specified.

Committee Response:
This was accepted as an editorial change to the Standard.

The wording in the Standard was changed (§4.9.5.6, page 128 of the May 1988 draft) to state that `_IOLBF` causes input/output to be line buffered.

Item 132 (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: Dislikes undefined behavior in `fprintf` *etc.*

Committee Response:
Quality of implementation is beyond the scope of the Standard.

Leaving this behavior undefined allows an implementation to choose semantics appropriate for its environment.

Item 134 (§4.9.6.2, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: Allow * to indicate maximum field width in `fscanf` *etc.*

Committee Response:
This proposal would conflict with other portions of the Standard.

The addition of this feature would cause an ambiguity with the assignment-suppressing use of *.

Item 135 (§4.9.6.2, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: Make the hidden "peek" function available for application use.

Committee Response:
The Committee has voted against this idea.

For some implementations `ungetc` could be used by `scanf`, and for others `scanf` need not call an actual function to perform the look-ahead. Thus the function you request be made available might not exist.

Item 140(a) (§4.9.6, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: Add formatted I/O functions that invoke an application-supplied character I/O function.

Committee Response:
This was considered to be an invention of limited utility.

Item 140(b) (§4.9.6, Doc. No. X3J11/88-099 Letter L164 p. 7)

Summary of Issue: `vfprintf` *etc.* don't support multi-level use of variable argument lists.

Committee Response:
This is a misinterpretation of correct wording in the document.

The current specification already provides the functionality of passing a variable argument list through multiple levels of calls by passing a `va_list` argument, as specified in §4.8.

Item 141 (§4.9, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: Stream I/O functions should be required to set `errno` upon error.

Committee Response:
The Committee discussed this proposal but decided against it.

The Standard reflects widespread existing practice in this regard.

Item 142 (§4.9, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: Wants a function to test for availability of input.

Committee Response:
In some cases, this proposal would be difficult to implement.

This would indeed be useful in certain situations, but may not be practical in some environments.

Item 144 (§4.9.7.11, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: How can one discard unwanted pushed-back characters on an unseekable stream?

Committee Response:
This was considered a request for information, not an issue.

There is no portable method of discarding of pushed-back characters for a file without a file position indicator. Such discarding was not intended to provide functionality to be exploited by applications, but rather to streamline implementation of the seek functions. There was no support for changing this.

Item 145 (§4.9.8.2, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: What does `fwrite` do when `nmemb` is 0?

Committee Response:
The Committee believes this is clear enough as is.

The Standard clearly states that the `fwrite` function returns the number of members successfully written.

Item 150(a) (§4.10.1.4, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: The `l` should be optional in `le25` for `strtod`.

Committee Response:
This was considered to be an invention of limited utility.

The Committee felt that there was no prior art existing in this area to support such a change.

Item 150(b) (§4.10.1, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: Make `dtostr` *etc.* available.

Committee Response:
This was considered to be an invention of limited utility.

As you note, `sprintf` can be used for this purpose, even though it may indeed be "overkill".

Item 150(c) (§4.10.1, Doc. No. X3J11/88-099 Letter L164 p. 8)

Summary of Issue: Provide functions to convert a pointer to a textual representation and *vice versa.*

Committee Response:
This was considered to be an invention of limited utility.

As you note, the `%p` format can be used for this purpose.

Item 154 (§4.10.3, Doc. No. X3J11/88-099 Letter L164 p. 9)

Summary of Issue: An available-memory function would be nice.

Committee Response:
The Standard must accommodate a variety of environments.

Such a function would not be reliable in all environments (for example in a multi-tasking system).

Item 163 (§4.11.2.4, Doc. No. X3J11/88-099 Letter L164 p. 9)

Summary of Issue: Need a function like `strncpy` with null termination guaranteed.

Committee Response:
This was considered to be an invention of limited utility.

The user can simply test the last character to see if it is null (since we guarantee that the rest of the string will be null-padded if less than n characters are copied).

Item 170(a) (§4.12.1, Doc. No. X3J11/88-099 Letter L164 p. 9)

Summary of Issue: Is `CLK_TCK` an integer constant?

Committee Response:
This was considered a request for information, not an issue.

`CLK_TCK` is an implementation-defined value and is not guaranteed to be an integer constant.

Item 170(b) (§4, Doc. No. X3J11/88-099 Letter L164 p. 9)

Summary of Issue: Provide support for generic objects.

Committee Response:
This was considered to be an invention of limited utility.

Item 170(c) (§4, Doc. No. X3J11/88-099 Letter L164 p. 10)

Summary of Issue: There is no way to ascertain object alignment requirements.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

Types may be aligned differently depending on where they are used. It was decided to allow implementors freedom in this regard.

Item 170(d) (§4, Doc. No. X3J11/88-099 Letter L164 p. 10)

Summary of Issue: (Optionally) enlarge the standard library.

Committee Response:
Adding too many facilities would unduly enlarge the language.

The Committee has chosen to standardize those functions it feels are most essential. To require all those that you suggest would be unduly burdensome, while making them optional would introduce the notion of "levels" of conformance, a concept the Committee has long opposed.

Item 30 (§3.1.4, Doc. No. X3J11/88-099 Letter L164 p. 12)

Summary of Issue: String literals should be arrays of `const char`.

Committee Response:
This proposal would invalidate too much existing source code.

Although this is an appealing suggestion, the Committee was forced to leave string literals as arrays of characters, not arrays of constant characters, since there exist too many instances of code that initializes a non-`const` pointer variable with a string literal, as in

```
char *p = "xxx";
```

Item 32 (§R3.1.9, Doc. No. X3J11/88-099 Letter L164 p. 12)

Summary of Issue: Insufficient reason is given for disallowing nested comments.

Committee Response:
The Standard reflects the base document in this regard.

Existing code may well contain a comment of the form:

```
/* ...
/* ...
*/
```

Allowing for nested comments would break such code.

Item 36 (§3.2.2.3, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: The prohibition against interconversion of function and object pointers is not found in the Standard.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

§3.3.2.2 says that if the types of the arguments after promotion are not compatible with those of parameters after promotion, the behavior is undefined. `NULL` is not defined to be compatible with pointer types (see §3.5.4.1). §3.2.2.3 allows `NULL` to be explicitly assigned to or compared for equality to a pointer; it does not allow `NULL` to be passed to a function without its prototype being visible.

Item 52 (§3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: How about a `writeonly` type qualifier?

Committee Response:
This was considered to be an invention of limited utility.

Item 56 (§R3.5.7, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: The example isn't "troubling".

Committee Response:
The Standard reflects the result of previous discussion of this issue.

The Committee feels that the implementation freedom gained by the rules justifies the restrictions imposed on the users.

Item 57 (§3.5.7, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: Initialization of non-first member of a union would be useful.

Committee Response:
This was considered to be an invention of limited utility.

You can usually declare the union members in the necessary order.

Item 58 (§3.6.6.3, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: A way to break out of a loop from within a switch body would be useful.

Committee Response:
This was considered to be an invention of limited utility.

You can use `goto` for this.

Item 61 (§3.8.1, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: Yet another suggestion for a way to test system type, *etc.*

Committee Response:
This was considered to be an invention of limited utility.

The variety of, and rapid change of, systems prevents any standardization of this kind of function.

Item 65 (§R3.8.3.3, Doc. No. X3J11/88-099 Letter L164 p. 13)

Summary of Issue: Macro expansion *before* pasting would seem more useful.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

When the operand of `##` is a formal parameter, the corresponding actual argument replaces it, but is not further (recursively) expanded. Thus

```
#define     a(n)   aaa##n
#define     b      2
a(b)  /* expands to aaab, not aaa2 or aaan. */
```

We hope this clarifies the meaning of "expand" in this context.

Item 66 (§3.8.5, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: Add `#warning` as a non-fatal version of `#error`.

Committee Response:
The Committee has voted against this idea.

There is not a large enough body of prior art for this.

Item 70 (§R4.1.2, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: Recommend a method for wrappers to attain header idempotency.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 77(a) (§R4.4.1.1, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: Would appreciate implementation hints for locale.

Committee Response:
The Committee discussed this proposal but decided against it.

Various implementations of this feature currently exist. The implementors have not complained about undue inefficiency.

Item 77(b) (§4.5, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: If `dtostr` *etc.* aren't added, then retain `ecvt` *etc.*

Committee Response:
The Standard reflects the result of previous discussion of this issue.

The Committee feels that the `sprintf` function provides equivalent functionality.

Item 82 (§4.7, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: Define `SIGERR`, to be raised for algorithmic errors.

Committee Response:
In some cases, this proposal would be difficult to implement.

Also, this is pure invention, which the Committee tries to avoid unless it addresses a critical need.

Item 88 (§4.9.5.1, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: The implementation could write a zero byte when closing an unwritten stream, rather than placing this burden on the application.

Committee Response:
In some cases, this proposal would be difficult to implement.

Also, this might be surprising behavior in an environment that just drops empty files. There could be (nonportable) code on such implementations that depends on an unwritten file not being created at `fclose` time.

Item 94(a) (§4.9.7.11, Doc. No. X3J11/88-099 Letter L164 p. 14)

Summary of Issue: Yet another suggestion about how `ungetc` should work.

Committee Response:
This proposal would invalidate too much existing source code.

The change you proposed would break existing code that uses `ungetc` after a file-positioning function.

Item 94(b) (§R4.9.9, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Should recommend preferred use of `fgetpos` over `ftell`.

Committee Response:
No change to the existing wording was considered necessary.

The functions `fgetpos` and `fsetpos` are intended to supplement `fseek` and `ftell`, not replace them. The old functions provide greater flexibility when the programmer knows that the file is small enough.

Item 6 (§2.1.1.1, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Change "files" to "streams".

Committee Response:
No change to the existing wording was considered necessary.

The Committee feels that "files" is the appropriate term here. This should not be confused with the C concept of "streams".

Item 8 (§2.1.2.2, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Prohibit attempts to extend the length of `argv` strings.

Committee Response:
No change to the existing wording was considered necessary.

This is unlikely to be misunderstood.

Item 9 (§2.1.2.3, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Change "files" to "streams".

Committee Response:
No change to the existing wording was considered necessary.

The Committee feels that "files" is the appropriate term here. This should not be confused with the C concept of "streams".

Item 14 (§2.2.4.1, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Clarify that 32 *levels* of expression nesting must be supported.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 16 (§2.2.4.2, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Values are not given for some `FLT_*` *etc.* macros.

Committee Response:
No change to the existing wording was considered necessary.

These macros do not have required minimum or maximum values; they merely provide implementation-dependent information.

Item 19 (§3.1, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Change to "each non-white-space character that cannot be *part of* one of the above".

Committee Response:
No change to the existing wording was considered necessary.

The current wording is technically correct.

Item 21(a) (§3.1.2, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: §3.1.2.* should not be subsections of the **Keywords** section.

Committee Response:
The request is reflected in the current draft.

Item 21(b) (§3.1.2.1, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Can argument names differ between a function prototype and the corresponding function definition?

Committee Response:
This was considered a request for information, not an issue.

Yes, the names can differ.

Item 21(c) (§3.1.2.1, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Outer declaration is not hidden if the declarations have linkage.

Committee Response:
No change to the existing wording was considered necessary.

The inner declaration still goes out of scope at the block end. However, there are extra constraints on linkage if a file-scope declaration of the same identifier is visible at the beginning of the nested declaration.

It is felt that *scope* and *linkage* are best described separately.

Item 21(d) (§3.1.2.4, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Doesn't understand the footnote.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

A volatile object may have a value stored at any time independent of the program.

Item 24(a) (§3.1.2.5, Doc. No. X3J11/88-099 Letter L164 p. 15)

Summary of Issue: Aren't `signed char` and `unsigned char` also basic types?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The signed and unsigned integer types include the signed and unsigned `char` types.

Item 24(b) (§3.1.2.5, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Say that a structure contains "non-overlapping", not "sequentially allocated", members.

Committee Response:
The Committee believes this is clear enough as is.

Item 24(c) (§3.1.2.5, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: These definitions are hard to find.

Committee Response:
The Committee believes this is clear enough as is.

Item 26 (§3.1.2.6, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Clarify that the types being referred to are function types.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 34 (§3.1.9, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: `/*` in header name does not start a comment.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

In fact this behavior is explicitly undefined — see §3.1.7, page 32, lines 4-6 in the May 1988 draft.

Item 35 (§3.2, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Conversion to a compatible type causes no change to *what*?

Committee Response:
This was accepted as an editorial change to the Standard.

Conversion of an operand value to a compatible type causes no change to the value or representation.

Item 36 (§3.2.1, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Subsection titles seem to be in differing fonts.

Committee Response:
This was not considered an issue requiring action.

The master document uses the same font for these. This was probably just a result of the reproduction process.

Item 37(a) (§3.2.2.1, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Type is "that of", not "specified by", the lvalue.

Committee Response:
The Committee believes this is clear enough as is.

Item 37(b) (§3.2.2.1, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Add reference to use of function designator as pointer.

Committee Response:
The Committee believes this is clear enough as is.

See §3.2.2.1 of the May 1988 draft. Also, there is an example in §3.7.1.

Item 41(a) (§3.3.2.2, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Change "argument" to "actual argument".

Committee Response:
The Committee believes this is clear enough as is.

This term is defined in §1.6.

Item 41(b) (§3.3.2.2, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Change "parameter" to "formal argument".

Committee Response:
The Committee believes this is clear enough as is.

This term is defined in §1.6.

Item 42(a) (§3.3.2.2, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Fix wording about type not including a prototype.

Committee Response:
The request is reflected in the current draft.

However, we took a different approach.

Item 42(b) (§3.3.2.2, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Add "However,".

Committee Response:
The Committee believes this is clear enough as is.

Item 42(c) (§3.3.2.2, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Add a footnote to explain why there needs to be a sequence point before the actual call to a function.

Committee Response:
The Committee believes this is clear enough as is.

This is inappropriate for the Standard, which is not intended to double as a tutorial. The reason of course is that some guarantee is needed, since the function might refer to an external object that was supposed to be modified as a side-effect of argument evaluation, and there is no other guarantee of this in the Standard.

Item 43(a) (§3.3.2.3, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Change "an arrow `->`" to "the `->` operator".

Committee Response:
The Committee believes this is clear enough as is.

Item 43(b) (§3.3.3.1, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Add "`(E-=1)`".

Committee Response:
The Committee believes this is clear enough as is.

Item 45(a) (§3.3.3.4, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: `sizeof(void)` should be invalid.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

It *is* invalid; `sizeof` can't be applied to incomplete types. See §3.3.3.4 of the May 1988 draft.

Item 45(b) (§3.3.3.4, Doc. No. X3J11/88-099 Letter L164 p. 16)

Summary of Issue: Result of `sizeof` is an *unsigned* integer constant.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The type of the result is explained later in the **Semantics** section. It is an unsigned integral type.

Item 48 (§3.3.6, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Delete "incrementing is equivalent to adding 1".

Committee Response:
No change to the existing wording was considered necessary.

Note that there is a similar statement about decrementing a few lines farther on.

Item 50(a) (§3.3.7, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Change "undefined" shift to "implementation defined".

Committee Response:
No change to the existing wording was considered necessary.

The Committee believes that "undefined" is the correct category for this. See §1.6 of the May 1988 draft. Besides, how could a portable program use it?

Item 50(b) (§3.3.8, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Need a defined term for "qualified or unqualified versions of compatible types".

Committee Response:
This proposed editorial change was discussed but not accepted.

The terms are defined in §3.1.2.5 and §3.1.2.6.

Item 54 (§3.3.16.1, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Reference prohibition on type conversion by assignment between two members of the same union.

Committee Response:
This is a misinterpretation of correct wording in the document.

This *is* the paragraph in question.

The Committee believes this is clear enough as is.

Item 59 (§3.5.1, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Change "an identifier for a function" to "a function identifier".

Committee Response:
The Committee believes this is clear enough as is.

Since it is clearly stated that *identifiers*, not functions, have scope, we don't think this can be misread.

Item 62(a) (§3.5.2.1, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Add forward reference to **Tags** section.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 62(b) (§3.5.2.2, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Clarify footnote.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 63 (§3.5.2.2, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Add forward reference to **Tags** section.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 65 (§3.5.4, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: "Contains" is poor wording.

Committee Response:
The request is reflected in the current draft.

Item 68(a) (§3.5.4.1, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Reword for clarity.

Committee Response:
The Committee believes this is clear enough as is.

Item 68(b) (§3.5.4.2, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Say which expression is meant.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 70 (§3.5.4.3, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: "External linkage" is wrong.

Committee Response:
This editorial change has been made.

Item 71(a) (§3.5.5, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Add more space in syntax.

Committee Response:
This editorial change has been made.

Item 71(b) (§3.5.6, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Reword "specifies the type specified".

Committee Response:
The Committee believes this is clear enough as is.

Item 72 (§3.5.7, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Structure or union can be initialized with non-constants.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

There is no conflict. The **Constraint** refers only to members of an *initializer list* (enclosed in braces). The license in **Semantics** for non-constant expressions in a `struct` or `union` initializer applies only to an initializer that is a single *assignment expression* (no braces).

Item 73(a) (§3.5.7, Doc. No. X3J11/88-099 Letter L164 p. 17)

Summary of Issue: Reword to "no initializer at the declaration if ...".

Committee Response:
Changes have been made along the lines you suggested.

Item 73(b) (§3.5.7, Doc. No. X3J11/88-099 Letter L164 p. 18)

Summary of Issue: Should say "in a *brace-enclosed* list".

Committee Response:
This was accepted as an editorial change to the Standard.

Item 80 (§3.6.6.1, Doc. No. X3J11/88-099 Letter L164 p. 18)

Summary of Issue: Change "current function" to "function containing the `goto`".

Committee Response:
Changes have been made along the lines you suggested.

Item 84(a) (§3.7.2, Doc. No. X3J11/88-099 Letter L164 p. 18)

Summary of Issue: There appears to be confusion about internal *vs.* external definitions.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

There are no internal definitions. You are confusing external linkage with external definition.

Item 84(b) (§3.7.2, Doc. No. X3J11/88-099 Letter L164 p. 18)

Summary of Issue: Reword sentence about tentative definitions.

Committee Response:
This proposed editorial change was discussed but not accepted.

The Committee believes this is clear enough as is.

Item 84(c) (§3.7.2, Doc. No. X3J11/88-099 Letter L164 p. 18)

Summary of Issue: Add statement about linkage disagreement.

Committee Response:
This proposed editorial change was discussed but not accepted.

The Committee believes this is clear enough as is.

Item 84(d) (§3.7.2, Doc. No. X3J11/88-099 Letter L164 p. 18)

Summary of Issue: Reword whole section.

Committee Response:
This is too radical a change to adopt at this stage.

The Committee believes the Standard is clear enough as is.

Item 90 (§3.8.3, Doc. No. X3J11/88-099 Letter L164 p. 20)

Summary of Issue: Embedded (in text is awkward.

Committee Response:
No change to the existing wording was considered necessary.

The Committee decided that the suggested changes would be inconsistent with the rest of the document.

Item 92 (§3.8.3.5, Doc. No. X3J11/88-099 Letter L164 p. 20)

Summary of Issue: Another disadvantage of a macro is being unable to take its address.

Committee Response:
This editorial change has been made.

However, the Committee did not intend the list to be complete.

Item 93 (§3.8.3.5, Doc. No. X3J11/88-099 Letter L164 p. 20)

Summary of Issue: Example page is formidable.

Committee Response:
This was considered a comment rather than an issue.

Item 94 (§3.8.4, Doc. No. X3J11/88-099 Letter L164 p. 20)

Summary of Issue: String literal use precedes its definition.

Committee Response:
The request is reflected in the current draft.

Item 125 (§4.9.3, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Reword "creating an existing file".

Committee Response:
The Committee believes this is clear enough as is.

We decided to retain the existing wording; your proposed words would introduce an ambiguity regarding the antecedent for "its contents", since two distinct files would seem to be involved.

Item 126 (§4.9.3, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Buffered streams transmit data at times other than just when a buffer is filled.

Committee Response:
No change to the existing wording was considered necessary.

Your comments are correct. This is covered elsewhere in the Standard.

Item 127 (§4.9.4.2, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: `rename` may also fail if the named target exists.

Committee Response:
This proposed editorial change was discussed but not accepted.

Actually, this is explicitly implementation-defined.

Item 131 (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Padding may be with zeros instead of spaces.

Committee Response:
Changes have been made along the lines you suggested.

Item 132(a) (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: If the precision is omitted, what is it taken as?

Committee Response:
This was considered a request for information, not an issue.

If the precision is omitted, the default is defined in the description for each conversion specifier.

Item 132(b) (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: * for precision requires the leading .

Committee Response:
Changes have been made along the lines you suggested.

Item 132(c) (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: State that flag characters may appear in any order.

Committee Response:
Changes have been made along the lines you suggested.

Item 132(d) (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: When - is omitted, result will be right-justified.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 132(e) (§4.9.6.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: When + is omitted, a plus sign shall not appear in the result.

Committee Response:
This was accepted as an editorial change to the Standard.

The Committee has added a description of the form for decimal numbers to the Standard.

Item 136 (§4.9.6.2, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: It's not the input pointer that's undefined, but its use.

Committee Response:
This proposed editorial change was discussed but not accepted.

This is unlikely to be misunderstood.

Item 137 (§4.9.6.2, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Change "early conflict" to "matching failure".

Committee Response:
The request is reflected in the current draft.

Item 144 (§4.9.7.11, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Delete redundant sentence.

Committee Response:
This proposed editorial change was discussed but not accepted.

The Committee believes that the sentence in question adds to the clarity of the document.

Item 165 (§4.11.4.5, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Add "if `sl` is not null".

Committee Response:
The Committee believes this is clear enough as is.

Item 170 (§4.12.2.1, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Processor time used by *what*?

Committee Response:
No change to the existing wording was considered necessary.

This is discussed under **Returns** for the `clock` function.

Item 197(a) (§A.5, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Is pointer conversion an implicit narrowing?

Committee Response:
The Committee believes this is clear enough as is.

Yes, in some environments pointer conversion is an implicit narrowing.

Item 197(b) (§A.7, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Add "union initialization" to the **Index**.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 197(c) (§A.7, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Embolden primary description section numbers in the **Index**.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 13(a) (§R2.1.2.3, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Comment about the term "common subexpression elimination".

Committee Response:
This was considered a comment rather than an issue.

Item 13(b) (§R2.1.2.3, Doc. No. X3J11/88-099 Letter L164 p. 21)

Summary of Issue: Fix spelling: "netwerk" (typo).

Committee Response:
The request is reflected in the current draft.

Item 15 (§R2.2.1.1, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Multiple ?s are probably more common.

Committee Response:
No change to the existing wording was considered necessary.

Multiple ?s are unchanged unless followed by one of the special characters.

Item 16 (§R2.2.2, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Omit "alphabet".

Committee Response:
Changes have been made along the lines you suggested.

Item 22 (§R3.1.2.2, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Suggests a better reason for on-the-fly name generation.

Committee Response:
No change to the existing wording was considered necessary.

This was considered to be just another example and unnecessary.

Item 32 (§R3.1.7, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Command name was omitted (typo).

Committee Response:
The request is reflected in the current draft.

Item 35 (§3.2.2.1, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Move statement about when array is converted to pointer to the Standard.

Committee Response:
The Committee believes this is clear enough as is.

The Standard is already clear enough; the Rationale was misleading and this has been fixed. Thank you for calling our attention to this out-of-date wording.

Item 36 (§3.2.2.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Shouldn't the Standard note that function pointers may be incommensurate with object pointers or integers?

Committee Response:
The Committee believes this is clear enough as is.

The Standard explicitly states that this is the case.

Item 37 (§R3.2.2.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Change "mapping the" to "mapping to" (typo).

Committee Response:
The request is reflected in the current draft.

Item 39 (§R3.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Change "optimize to" to "optimize the" (typo).

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 48 (§R3.4, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: "Integer constants" seems too restrictive.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 51 (§R3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Improve wording about `noalias`.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was dropped altogether between the second and third public reviews.

Item 52(a) (§R3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Start sentence with "If".

Committee Response:
The request is reflected in the current draft.

Item 52(b) (§R3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Comment about `const noalias`.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was dropped altogether between the second and third public reviews.

Item 52(c) (§R3.5.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Comment about `volatile noalias`.

Committee Response:
The Committee has made significant changes in this area.

`noalias` was dropped altogether between the second and third public reviews.

Item 53 (§R3.5.4, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Some implementations *now* have such a limit.

Committee Response:
The request is reflected in the current draft.

Item 55 (§R3.5.4.3, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Not only "more efficient" calling sequence, but also follow system standard inter-language conventions.

Committee Response:
No change to the existing wording was considered necessary.

Item 59(a) (§R3.7.1, Doc. No. X3J11/88-099 Letter L164 p. 22)

Summary of Issue: Should be "as *names of* formal parameters".

Committee Response:
This proposed editorial change was discussed but not accepted.

One cannot use a type name as a name of an object in the same scope. If it is redefined within another scope, that redefines the name.

Item 59(b) (§R3.7.1, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Can't this also start a function definition?

Committee Response:
Changes have been made along the lines you suggested.

Item 59(c) (§R3.7.1, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: There is an apparent contradiction about type rewriting.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 63 (§R3.8.3, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Macros *cannot* be used wherever functions can be.

Committee Response:
This proposed editorial change was discussed but not accepted.

The Committee decided that the Rationale is clear as is.

Item 66 (§R3.8.4, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Omit "to" after "giving" (typo).

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 70 (§R4.1.2, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Add "be" before "includable" (typo).

Committee Response:
The request is reflected in the current draft.

Item 71 (§R4.1.4, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Whose inventions?

Committee Response:
No change to the existing wording was considered necessary.

The "inventions" in question have been created in multiple circumstances and thus the suggested change is not appropriate.

Item 75 (§R4.4, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: The use of period as decimal point *was* built into the library functions.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 76 (§R4.4, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Delete redundant "are" (typo).

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 78 (§R4.5, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Bit patterns are not "hexadecimal"!

Committee Response:
This was accepted as an editorial change to the Rationale.

The word "hexadecimal" has been removed.

Item 81 (§R4.6, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Change "to" to "the" (typo).

Committee Response:
The request is reflected in the current draft.

Item 84 (§R4.8.1.1, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Why "(nested)"?

Committee Response:
This was accepted as an editorial change to the Rationale.

Originally we thought a { might be contained in `va_start`. It does not have to be nested. The Committee has made a change to the document.

Item 89 (§R4.9.5.3, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Put Table 4.1 into the Standard.

Committee Response:
No change to the existing wording was considered necessary.

The tables are in the Rationale for information purposes only.

Item 100 (§R4.10.6.2, Doc. No. X3J11/88-099 Letter L164 p. 23)

Summary of Issue: Put Table 4.2 into the Standard.

Committee Response:
No change to the existing wording was considered necessary.

The tables are in the Rationale for information purposes only.

X3J11 Response to Letter L165

Correspondent's Name and Address:

Rainer Gerhards
Petronellastr. 6
D-5112 Baesweiler
West Germany

Item 1 (§3.8.6, Doc. No. X3J11/88-100 Letter L165 p. 1)

Summary of Issue: `#pragma` is incompatible with existing preprocessors.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

We have rehashed the pragma question many times. If we were to change the `#pragma` directive, there would be a number of subtle changes required throughout the Standard, requiring additional public review. We could find no existing practice to evaluate your alternative proposal.

The Committee feels that any program with `#pragma` probably includes other ANSI-specific constructs. Such constructs are best protected by an `#if` preprocessor directive. Because it may break pre-ANSI C implementations even when "`#if`ed-out", `#pragma` is best hidden from the preprocessor by being contained in a conditionally-included header file. This is admittedly awkward, but it does work, and since use of `#pragma` is expected to be relatively rare a simpler approach does not seem necessary.

Since the original K&R specification for directive lines required that the # be the first character on the line, some older compilers may accept " `#pragma` ..." (note the initial space).

X3J11 Response to Letter L166

Correspondent's Name and Address:

Paul D. Watson
Microdynamics Incorporated
10461 Brockwood Road
Dallas, TX 75238

Item 1 (§2.2.1, Doc. No. X3J11/88-101 Letter L166 p. 1)

Summary of Issue: Has there been any attempt to use an international standard for character sets?

Committee Response:
This concerns matters beyond the scope of X3J11.

The establishment of a standardized international character set is outside the scope of this Committee or any programming language committee.

Furthermore, the Japanese and European spokesmen have insisted on a 1-byte `char` for efficiency reasons. Proposals for "cleaner" alternatives to multibyte characters were discussed at previous meetings and rejected in favor of the method embodied in the current draft, which has the approval of many people concerned with internationalization issues.

Nothing in the Standard prevents an implementation as you have described.

X3J11 Response to Letter L167

Correspondent's Name and Address:

Paul D. Watson
Microdynamics Incorporated
10461 Brockwood Road
Dallas, TX 75238

Item 1 (§2.2.1, Doc. No. X3J11/88-102 Letter L167 p. 1)

Summary of Issue: Should specify a single ISO-defined repertoire for characters.

Committee Response:
This concerns matters beyond the scope of X3J11.

Your believable observations about cheap high-performance machines being common in the future notwithstanding, compilers for today's machines compete on the basis of performance, and any requirement of more than 1-byte `chars` would put ANSI C compilers at a severe disadvantage.

The Japanese community is satisfied with our inclusion of `wchar_t` and multibyte characters in the proposed Standard. We feel it is beyond the scope of the Committee to define a specific ISO character set. Certainly users of other character sets such as EBCDIC would be displeased with any such specification.

X3J11 Response to Letter L168

Correspondent's Name and Address:

Richard H. Gumpertz
PARS Service
P.O. Box 20007
Kansas City, MO 64195

Item 3.5.7 (§3.5.7, Doc. No. X3J11/88-103 Letter L168 p. 1)

Summary of Issue: Allow empty initializer lists.

Committee Response:
This was considered to be an invention of limited utility.

The initializer `{0}` accomplishes the same thing.

Item min (§3.1.5, Doc. No. X3J11/88-103 Letter L168 p. 1)

Summary of Issue: Wants min and max operators.

Committee Response:
This is too radical a change to adopt at this stage.

As you mentioned, they are available as generic (but unsafe) macros. They also can be implemented as safe (but not generic) function calls. Complicating the language just to obtain safe, generic operations did not seem worth the burden upon all implementations.

Item 3.3.8 (§3.3.8, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Are relational comparisons with a null pointer well-defined?

Committee Response:
In some cases, this proposal would be difficult to implement.

Equality comparison against a null pointer is well-defined, but not relational (order) comparison. The pointers must point to parts of the same object.

Item 3.3.9 (§3.3.9, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Equality operation is not well-defined when an operand is a null pointer expression.

Committee Response:
Changes have been made along the lines you suggested.

Item 4.12.3.2 (§4.12.3, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Time functions should return structures, not pointers to internal static storage.

Committee Response:
The Standard reflects widespread existing practice in this regard.

"Hysterical raisins", as you said in a later context, which we take to mean "historical reasons".

Item ctime (§4.12.3, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Time function parameters should be `time_t`, not pointer to `time_t`.

Committee Response:
The Standard reflects widespread existing practice in this regard.

More "hysterical raisins".

(There are several problems with library routine interfaces which we would fix if we were designing them from scratch rather than canonicalizing existing practice.)

Item 4.6.1 (§4.6.1, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Should allow `setjmp` to be used as operand of `&&` or `||`.

Committee Response:
The Standard must accommodate a variety of architectures.

After a `longjmp`, control flow resumes in the function that called `setjmp`, just as though its call had returned. If this flow can resume in the middle of an expression evaluation, some architectures will experience problems dealing with balancing the stack, *etc.*

Item 3.5.2.2 (§3.5.2.2, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Allow size and signedness modifiers on `enum`s.

Committee Response:
The Committee has voted against this idea.

There is no existing practice, and insufficient utility.

Item 3.5.2.1(a) (§3.5.2.1, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Allow size modifiers on bit fields.

Committee Response:
The Committee has voted against this idea.

A **Common extension** already describes this idea.

Also see our response to letter L164 page 3, item 61.

Item 3.5.2.1(b) (§3.5.2.1, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Allow `enum` for bit field.

Committee Response:
The Committee has voted against this idea.

Again, this is described as a **Common extension**.

Item 4.8(a) (§4.8, Doc. No. X3J11/88-103 Letter L168 p. 2)

Summary of Issue: Dislikes `va_list` becoming indeterminate after a function call using it.

Committee Response:
The Standard must accommodate a variety of environments.

There already exist different implementations, some using an array type for a `va_list`, others using a scalar or structure type. So the called function would update the called-by-reference `va_list` in some environments, but not in others.

Item 4.8(b) (§4.8, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: May one use variables of type `va_list *`?

Committee Response:
This was accepted as an editorial change to the Rationale.

The answer is "yes". Additional wording has been added to the Rationale to make this usage clear.

Item 4.8(c) (§4.8, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: May one assign variables of type `va_list`?

Committee Response:
This was accepted as an editorial change to the Rationale.

The answer is "no". The implementation may choose to implement these variables in a way that is incompatible with the assignment operator. Wording was added to the Rationale to clarify this issue.

Item 4.8.1.2 (§4.8.1.2, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: Dislikes variable arguments having promoted type.

Committee Response:
This was considered to be an invention of limited utility.

The benefits of not widening variable arguments are not great enough to warrant this change. Wording was added to the Rationale to clarify this issue.

Item pure (§3.7.1, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: Wants support for the notion of a "pure" function.

Committee Response:
The Committee has voted against this idea.

The Committee has discussed this at previous meetings.

Note that a sufficiently clever implementation can perform many of the kind of optimizations that this would provide.

Item 3.5.3 (§3.5.3, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: Add type qualifiers with complementary meanings.

Committee Response:
This was considered to be an invention of limited utility.

Not enough benefit was seen to justify this large a change to the Standard.

Item 4.12.1 (§4.12.1, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: Offset of `tm_year` by −1900 is silly.

Committee Response:
The Standard reflects widespread existing practice in this regard.

You guessed it — "hysterical raisins".

Item 4.12.2.3(a) (§4.12.2.3, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: Specify in more detail the meaning of negative `tm_isdst`.

Committee Response:
The Standard must accommodate a variety of environments.

Implementations differ in the availability of all this information. The offset of "one hour" is locale-dependent, by the way.

Item 4.12.2.3(b) (§4.12.2.3, Doc. No. X3J11/88-103 Letter L168 p. 3)

Summary of Issue: How should `tm_isdst` be set within a DST overlap?

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 4.12.2.3(c) (§4.12.2.3, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: Is `mktime` allowed to leave `tm_isdst` negative on return?

Committee Response:
This was considered a request for information, not an issue.

The answer is "yes".

Item 4.12.3.4(a) (§4.12.3.5, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: AM/PM strings conflict with normal usage.

Committee Response:
Changes have been made along the lines you suggested.

What `strftime` uses for its `%p` format is already specified as being locale-dependent. We have changed the wording to make it clear that a binary choice is not required, so that for example `N` could be used for noon. (It was decided to allow implementors freedom in this regard.)

It did not seem necessary to add these members to `struct lconv`.

Item 4.4.1.1 (§4.4.1.1, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: May `setlocale` truncate over-long string parameters?

Committee Response:
Quality of implementation is beyond the scope of the Standard.

The implementation can choose the number of significant characters for the locale name.

Item 4.12.3.4(b) (§4.12.3.5, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: Noon is not 12:00 p.m.

Committee Response:
No change to the existing wording was considered necessary.

The `strftime` model is such that there is no 2400; midnight would be expressed as 0000 (for military usage of a 24-hour clock) or 12:00 a.m. (for one common usage of a 12-hour clock), and noon would be expressed as 1200, or 12:00 p.m. If an application requires more carefully controlled behavior, it should perform its own time display formatting. Implementations may also support additional conversion specifiers as an extension.

Also see our response to item 4.12.3.4(a).

Item anonymous (§3.5.2.1, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: Wants anonymous structures and unions.

Committee Response:
The Committee has voted against this idea.

This suggestion may be entertained by those who consider future versions of the C Standard.

Item constructor (§3.5.7, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: Allow aggregate constructors in contexts other than initializers.

Committee Response:
This is too radical a change to adopt at this stage.

Perhaps this can be done in a future version of the C Standard.

Item 3.8.3.2 (§3.8.3.2, Doc. No. X3J11/88-103 Letter L168 p. 4)

Summary of Issue: Wants `#(`*constant-expression*`)` to expand to minimal decimal representation of the constant expression.

Committee Response:
This is too radical a change to adopt at this stage.

This does not appear to be based on prior art.

X3J11 Response to Letter L169

Correspondent's Name and Address:

Michael Brannigan
Information & Graphics Systems
15 Normandy Court
Atlanta, GA 30324

Item 1 (§3.3.8, Doc. No. X3J11/88-104 Letter L169 p. 1)

Summary of Issue: Comparison of floating-point data between register and memory may cause problems.

Committee Response:
This was considered a request for information, not an issue.

Summary: `double X,Y; if( X < Y ) ...` — must this be computed as type `double` rather than (partially) in extended precision?

An implementation is required to *narrow the value* whenever an explicit assignment occurs or when an explicit cast is used. Therefore,

```
X = 2.1e-3;
```

requires that a value with the precision dictated by type `double` be stored in `X`. Similarly,

```
(float) Y
```

requires that the value of `Y` be converted to the precision dictated by type `float`. The expression

```
X < Y
```

can be computed in an extended precision but this requires that the "narrowed" value be converted to extended precision. In

```
A < B/C
```

the value of `A` can be converted to an extended precision. After `B` and `C` are converted to extended precision the division can be performed in extended precision and finally the comparison can be performed in extended precision.

Unfortunately, your example is incomplete but it seems that the compiler in question is not standard conforming. We hope that this information will help you make that determination.

Item 2 (§3.5.4.2, Doc. No. X3J11/88-104 Letter L169 p. 1)

Summary of Issue: Have (unspecified) ideas about implementation of multidimensional arrays.

Committee Response:
A specific proposal is needed before action can be taken.

With regard to your concern with multidimensional arrays, we recognize that numerical software has needs that the present language level does not fully meet. We hope you and other reviewers with similar interests will participate in developing appropriate extensions. Thank you for your comments.

X3J11 Response to Letter L170

Correspondent's Name and Address:

Glen Seeds
XIOS Systems
1600 Carling Avenue
Suite 150
Ottawa, Ontario
Canada K1Z 8R8

Item 0(a) (General, Doc. No. X3J11/88-126 Letter L170 p. 1)

Summary of Issue: Inadequate information was given concerning how to respond.

Committee Response:
Changes have been made along the lines you suggested.

Sorry! We fully expected CBEMA/X3 to attach a cover letter with that information.

The response document now has the 15-day notice moved earlier to make it more visible.

Item 0(b) (§3.3.8, Doc. No. X3J11/88-126 Letter L170 p. 1)

Summary of Issue: Use a balloting/response method similar to POSIX.

Committee Response:
This concerns matters beyond the scope of X3J11.

Now that the final Standard has been approved, this is a moot point.

Item 1(a) (§4.6, Doc. No. X3J11/88-126 Letter L170 p. 2)

Summary of Issue: Requests point-by-point rebuttal of the arguments he had made in second-round comments.

Committee Response:
This was considered a comment rather than an issue.

The Committee is trying to provide a standard that can be efficiently supported on a variety of architectures. We believe that the Standard reflects current practice, as the base document provides no guidance in this area.

1. The words you object to are necessary because on some architectures it is difficult to restore registers to the values they hold as of the call to `longjmp`. More importantly, a number of optimizing compilers can allocate more than one (non-register) `auto` variable to a single machine register. It would be unwise for the draft to effectively prohibit this popular optimization of register allocation.

2. Yes, a strict, minimal implementation of `setjmp`/`longjmp` may break existing programs. However, it is the feeling of the Committee that programs that relied on the restoration of automatic variables could not be considered portable, even though such programs may have worked on more than one non-optimizing implementation.

3. The Committee provided the "`volatile`" loophole to allow portable programs to be written, in spite of the problem described in the previous item.

   Automatic `volatile` declarations can be construed to mean "don't optimize" or "no registers". Therefore, a `volatile` qualifier on a automatic variable essentially forbids the compiler to put such an object in a register across function calls.

4. We agree that some current applications of `setjmp`/`longjmp` are not maximally portable. This reflects the realities of working with C on various architectures. To make this facility

work as you wish on all architectures would be prohibitively expensive for some implementations.

Item 1(b) (§4.9.6, Doc. No. X3J11/88-126 Letter L170 p. 2)

Summary of Issue: Requests point-by-point rebuttal of the arguments he had made in second-round comments.

Committee Response:
This was considered a comment rather than an issue.

Your proposal to add the library functions `snprintf` and `vsnprintf` has merit. The proposal was brought before the Committee in a previous meeting and had support from some of the Committee members. However, the required 2/3 support in the Committee could not be mustered due to lack of prior art. That in itself is considered sufficient grounds for rejecting a proposal.

Of course, an implementation is free to provide such functions as an extension.

Item 2 (§R4.6, Doc. No. X3J11/88-126 Letter L170 p. 2)

Summary of Issue: The Rationale should justify all QUIET CHANGES.

Committee Response:
No change to the existing wording was considered necessary.

A full discussion of all such changes would make the Rationale much larger than desired, for little benefit.

The Committee believes that the Standard reflects existing practice in the `setjmp/longjmp` area. We do not believe that a QUIET CHANGE has been made.

Item 3 (§4.9.5.2, Doc. No. X3J11/88-126 Letter L170 p. 2)

Summary of Issue: The C dpANS and IEEE 1003.1 conflict with respect to `fflush` effect on input streams.

Committee Response:
This is a misinterpretation of correct wording in the document.

This appears to be a reference to the following (from P1003.1/DRAFT 13, §8.2.3.4):

> If the stream is open for reading, any unread data buffered in the stream shall be invalidated.
>
> For a stream open for reading, if the file is not already at EOF, and the file is one capable of seeking, the file offset of the underlying open file description shall be adjusted so that the next operation on the open file description deals with the byte after the last one read from or written to the stream being closed [*sic*].

In the proposed C Standard, the behavior of `fflush` on an input stream is ***undefined***, for which the Standard ***imposes no requirements***, which permits "add-on" standards such as IEEE 1003.1-1988 to define additional semantics. Thus, there is no conflict in this case.

X3J11 Response to Letter L171

Correspondent's Name and Address:

D. Hugh Redelmeier
29 Donino Avenue
Toronto, Ontario
M4N 2W6
Canada

Item 131/25 (§4.9.6.1, Doc. No. X3J11/88-127 Letter L171 p. 1)

Summary of Issue: There is no portable way to print an unsigned integer.

Committee Response:
Changes have been made along the lines you suggested.

Item 33/36 (§3.1.8, Doc. No. X3J11/88-127 Letter L171 p. 2)

Summary of Issue: Preprocessor number definition breaks existing correct programs.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

At the September 1988 meeting the Committee discussed two proposals, including yours, but decided to leave the definition the way it is. Trying to refine the notion of preprocessing number tokens at this stage is likely to introduce as many surprises as are fixed. The few cases of existing code that would be broken were not considered to present a significant practical problem.

Some Committee members insisted that they *wanted* preprocessor numbers to be "greedier" than one might think appropriate, so that (for example) binary constants such as `0b1001` could be supported as an extension, and that any change other than fixing *just* the "broken code" case would not be acceptable.

Item 119/15 (§4.6.2.1, Doc. No. X3J11/88-127 Letter L171 p. 3)

Summary of Issue: Make only *local* automatic variables vulnerable on `longjmp`.

Committee Response:
This was accepted as an editorial change to the Standard.

The paragraph beginning on page 116, line 11 of the May 1988 draft has been changed.

Item 25/15b (§3.1.2.5, Doc. No. X3J11/88-127 Letter L171 p. 4)

Summary of Issue: "Top type" is still described poorly.

Committee Response:
Changes have been made along the lines you suggested.

Item 49/40 (§3.3.8, Doc. No. X3J11/88-127 Letter L171 p. 4)

Summary of Issue: The reworded constraint list is incomplete.

Committee Response:
No change to the existing wording was considered necessary.

There simply are no valid uses that combine a pointer to a complete type and a pointer to an incomplete type for these operators.

Item 50/33 (§3.3.9, Doc. No. X3J11/88-127 Letter L171 p. 4)

Summary of Issue: Pointer comparison wording is wrong.

Committee Response:
Changes have been made along the lines you suggested.

Item 87/26 (§3.8, Doc. No. X3J11/88-127 Letter L171 p. 4)

Summary of Issue: Problems exist with `defined` and `sizeof`.

Committee Response:
Changes have been made along the lines you suggested.

X3J11 Response to Letter L172

Correspondent's Name and Address:

Robert Paul Corbett
ELXSI
2334 Lundy Place
San Jose, CA 95131

Item 1 (§2.2.4.2, Doc. No. X3J11/88-128 Letter L172 p. 1)

Summary of Issue: The floating-point model should be changed to use powers of b from 0 to $p-1$.

Committee Response:
The Committee believes this is clear enough as is.

Item 2 (§2.2.4.2, Doc. No. X3J11/88-128 Letter L172 p. 1)

Summary of Issue: Text and formula for `*_EPSILON` disagree.

Committee Response:
This was accepted as an editorial change.

We agree that this was misleading. We have changed the description of `*_EPSILON` to correspond to the formula b^{1-p}.

Item 3 (§2.2.4.2, Doc. No. X3J11/88-128 Letter L172 p. 2)

Summary of Issue: Example `DBL_MIN` is inaccurate.

Committee Response:
This editorial change has been made.

Item 4 (§3.8.4, Doc. No. X3J11/88-128 Letter L172 p. 2)

Summary of Issue: Line numbers should be defined for characters, not tokens.

Committee Response:
Quality of implementation is beyond the scope of the Standard.

The Standard leaves room for alternative interpretation on this point. If one thinks of the token as not being complete until its last character is processed, then §2.1.1.2, §3.8.4, and §3.8.8 make it clear that the value of `__LINE__` would be the line on which the last character of the token `__LINE__` occurs. Thus, by this interpretation, in your example the value printed would be 17. However, if one thinks of the token as being located at its *first* character, the value printed would be 10. It doesn't seem to much matter which choice is made by the implementation for such a degenerate case.

Item 5 (§4.2.1.1, Doc. No. X3J11/88-128 Letter L172 p. 3)

Summary of Issue: A strictly conforming program must declare `abort` before use of `assert`.

Committee Response:
This is a misinterpretation of correct wording in the document.

A strictly conforming program need not concern itself with the declaration of `abort` in this context; that is an issue for the implementor.

We are sorry that our first reply was in error; however, we still feel the burden is on a compiler implementor to ensure that `assert` calls `abort` properly (using an `int`-returning `__abort` library module that calls a properly declared `void abort();` if need be) rather than forcing a user to include `<stdlib.h>` in addition to `<assert.h>`.

Item 6 (§4.4.2.1, Doc. No. X3J11/88-128 Letter L172 p. 3)

Summary of Issue: Monetary examples for other countries are given, why not US?

Committee Response:
Changes have been made along the lines you suggested.

This is only an example, not a requirement for those locales, and the Standard now says that these rules "may well" be used, to make this clearer. The countries listed were chosen for the richness of their variation in locale parameter values. The U.S. is relatively uninteresting in this regard.

Item 7 (§4.5.1, Doc. No. X3J11/88-128 Letter L172 p. 3)

Summary of Issue: `tan` overflow sign problem is not just for large arguments.

Committee Response:
This was accepted as an editorial change.

Right you are. Thanks for pointing this out.

X3J11 Response to Letter L173

Correspondent's Name and Address:

Mark Brader
SoftQuad Inc.
720 Spadina Avenue
Toronto, Ontario
M5S 2T9
Canada

Item 10 (§2.1.2.3, Doc. No. X3J11/88-129 Letter L173 p. 1)

Summary of Issue: Add wording about parentheses in "as if" discussion.

Committee Response:
This was accepted as an editorial change to the Standard.

Text has been added to this section pointing out some of the consequences.

Item 24 (§3.1.2.4, Doc. No. X3J11/88-129 Letter L173 p. 2)

Summary of Issue: Delete "that returns".

Committee Response:
This was accepted as an editorial change to the Standard.

Item 25 (§3.1.2.5, Doc. No. X3J11/88-129 Letter L173 p. 2)

Summary of Issue: Would his proposed change conflict with any known existing implementations?

Committee Response:
It was decided to allow implementors freedom in this regard.

Yes, there are such implementations. In fact, this combination seems quite reasonable for an implementation that really cares about characters, for example for the large character sets needed for typography.

Item 28 (§3.1.2.5, Doc. No. X3J11/88-129 Letter L173 p. 2)

Summary of Issue: Require the same calling convention for `char *` and `void *`.

Committee Response:
This was accepted as an editorial change to the Standard.

A footnote was added to the Standard that explains the intent of the Committee with regard to function calling conventions.

Item 29 (§3.1.2.5, Doc. No. X3J11/88-129 Letter L173 p. 2)

Summary of Issue: The "top type" description is now comprehensible, but wrong.

Committee Response:
Changes have been made along the lines you suggested.

Item 35 (§3.3.1, Doc. No. X3J11/88-129 Letter L173 p. 2)

Summary of Issue: Wording about "form" is wrong.

Committee Response:
This was accepted as an editorial change to the Standard.

Yes, the phrase "form and value" is used in §3.1.3.

Item 38(a) (§3.3.6, Doc. No. X3J11/88-129 Letter L173 p. 3)

Summary of Issue: Need guarantee that a pointer to a scalar may be treated the same as a pointer to a 1-element array.

Committee Response:
This was accepted as an editorial change to the Standard.

Wording was added to give this guarantee.

Item 38(b) (§3.3.6, Doc. No. X3J11/88-129 Letter L173 p. 3)

Summary of Issue: Pointer subtraction rules are incomplete.

Committee Response:
This was accepted as an editorial change to the Standard.

The rules have been made complete.

Item 38(c) (§3.3.6, Doc. No. X3J11/88-129 Letter L173 p. 3)

Summary of Issue: Whole pointer arithmetic description is bogus.

Committee Response:
Changes have been made along the lines you suggested.

Item 42 (§3.5.2, Doc. No. X3J11/88-129 Letter L173 p. 3)

Summary of Issue: A promised editorial change was only partially made.

Committee Response:
This was accepted as an editorial change to the Standard.

The promised change has been made.

Item 43 (§3.5.6, Doc. No. X3J11/88-129 Letter L173 p. 3)

Summary of Issue: The new example is syntactically incorrect.

Committee Response:
This was accepted as an editorial change to the Standard.

The example is correct; the `typedef` causes `t` to be a defined type specifier, and the **Constraints** in §3.5.2 prohibit a typedef name from being combined with any other type specifier (such as `unsigned`) in the same specifier-qualifier list. Therefore the only valid parse of page 68, line 30 of the May 1988 draft is as a type specifier (`unsigned`) followed by an identifier used as a declarator (`t`). Words have been added after the example to further explain this.

Wording has been added to §3.5.2.1 to clarify the signedness of bit fields, even through use of `typedefs`.

Item 45 (§3.5.4.3, Doc. No. X3J11/88-129 Letter L173 p. 3)

Summary of Issue: Array and pointer *declarations* have never been equivalent, except in the one anomalous case.

Committee Response:
This was considered a comment rather than an issue.

You are correct; the equivalence was only in expressions.

Item 51 (§3.8.1, Doc. No. X3J11/88-129 Letter L173 p. 4)

Summary of Issue: A cross-compiler already has to know about the target character set.

Committee Response:
It was decided to allow implementors freedom in this regard.

The Standard says nothing about how the translation phases are to be implemented. As a result, several cross-compilers may all depend upon a single preprocessing program. The Committee therefore chose phase 5 as the earliest phase to require this information.

Item 53 (§4.3, Doc. No. X3J11/88-129 Letter L173 p. 4)

Summary of Issue: An argument validation function for the `is*` functions is essential.

Committee Response:
This was considered to be an invention of limited utility.

As you point out, the function may be trivially implemented; therefore adding it to the Standard would be of little benefit.

Item 55 (§4.4.2.1, Doc. No. X3J11/88-129 Letter L173 p. 4)

Summary of Issue: Wants names for numerical constants.

Committee Response:
The Committee discussed this proposal but decided against it.

These names were not not deemed significant enough to further pollute the name space.

X3J11 Response to Letter L174

Correspondent's Name and Address:

Steve Snelgrove
PO Box 284
Provo, UT 84603-284

Item 1 (§3.1.3.1, Doc. No. X3J11/88-130 Letter L174 p. 1)

Summary of Issue: Need a way to specify a floating-point constant as a bit pattern.

Committee Response:
This was considered to be an invention of limited utility.

Since the actual bit encoding of floating-point values is highly system dependent, a portable program cannot do this. A non-portable program can already perform this kind of manipulation by using a union. (Also note that many architectures do not have special representations for infinity, overflow, *etc.*)

X3J11 Response to Letter L175

Correspondent's Name and Address:

Donn Terry (IEEE 1003.1)
Hewlett-Packard Co.
Desktop Computer Div.
3404 E. Harmony Rd.
Ft. Collins, CO 80525

Item 1 (§2.1.2.2, Doc. No. X3J11/88-131 Letter L175 p. 1)

Summary of Issue: It should be implementation defined whether `environ` is a reserved external identifier.

Committee Response:
The Committee discussed this proposal but decided against it.

The ANSI C and IEEE 1003.1-1988 standards are not necessarily in conflict here, although it is true that in order to avoid the name-space conflict a mutually conforming implementation must rely on some mechanism such as "global symbolic equate" or a zero-size global object `environ` in a separate library module immediately preceding the module that defines storage for `__environ` (the name used by the C run-time startup code). Implementor control over the way the linker operates, while inappropriate to require for the more universal C Standard (hence the constraint on uniqueness of external identifiers), is not unrealistic to expect for most POSIX implementations. Several implementors have in fact indicated their intention to provide such a feature.

Another solution, of course, would be to use separate run-time startup modules for strict ANSI-conforming and vendor-extended (possibly POSIX-conforming) implementations, perhaps *via* a compiler flag. This may be useful anyway, for hiding extensions in certain standard headers.

X3J11 Response to Letter L176

Correspondent's Name and Address:

P.J. Plauger (ISO SC22/WG14)
Whitesmiths, Ltd.
59 Power Road
Westford, MA 01886

Item 1 (§2.2.1.1, Doc. No. X3J11/88-132 Letter L176 p. 1)

Summary of Issue: Need more readable trigraphs.

Committee Response:
The Committee discussed this proposal but decided against it.

See our response to X3J11/88-134 (L177), following.

Item 2 (General, Doc. No. X3J11/88-132 Letter L176 p. 1)

Summary of Issue: Make no other substantive changes.

Committee Response:
This proposal was accepted.

We do not believe we have made any substantial changes since the third public review version.

X3J11 Response to Letter L177

Correspondent's Name and Address:

Keld Simonsen and Bjarne Stroustrup
University of Copenhagen, Denmark
and
AT&T Bell Laboratories, USA

Item 1 (§2.2.1.1, Doc. No. X3J11/88-134 Letter L177 p. 1)

Summary of Issue: Proposal for more readable supplement to trigraphs.

Committee Response:
The Committee discussed this proposal but decided against it.

We cannot support this proposal for a number of reasons. Trigraphs were intended to provide a universally portable format for the *transmission* of C programs; they were never intended to be used for day-to-day reading and writing of programs. Should it be necessary to do so, however, the preprocessor can already be used to improve their readability (exact macro names and definitions are not provided as the Committee prefers to avoid stylistic issues). As larger character sets become more and more popular, the chances of having to deal with a "deficient" character set become smaller and smaller.

Conversion between the current trigraph representations and "normal" representations can be done simply in a context-free manner, but this is not possible with the proposed notation. Also, there are a number of difficulties with the infix subscript operator where empty brackets would have been used. Either the operator must be allowed as a postfix unary operator as well as a binary operator, or the grammar must be extended to allow empty parentheses to appear in those contexts where empty brackets can. Although these problems are by no means insurmountable, we feel that the current trigraphs are adequate for their intended use and that no further enhancements are necessary.

Translation phase 1 actually consists of two parts, first the mapping (about which we say very little) from the external host character set to the C source character set, then the replacement of C source trigraph sequences with single C source characters. (Note that the C source characters represented in our documents in `Courier` font need not appear graphically the same in the host environment, although a reasonable implementation will make them as nearly so as possible.) The kind of mapping you propose can in fact be done in the first part of translation phase 1, and several such "convenience" mappings are already common practice. However, attempting to standardize this mapping is outside the scope of the C Standard, since what is appropriate may depend on the capabilities of the specific hardware, availability of fonts, and so forth.

Although the Committee regrets any "no" votes on either the national or international proposed standards, we feel we must represent our best judgement on technical issues. We hope you will reconsider your objection to the current specification.

X3J11 Response to Letter L178

Correspondent's Name and Address:

Mark Dunn
Dept. Mechanical Engineering
Sheffield University
Mappin St.
Sheffield
England

Item 1 (§3.5.6, Doc. No. X3J11/88-135 Letter L178 p. 1)

Summary of Issue: Yet another proposal for `typeof`, along the lines of `typedef`.

Committee Response:
This does not appear to be based on prior art.

We are sympathetic to your plight. However, there is not enough prior art of this construct for the Committee to be sure that an addition to the Standard such as this will be bug-free.

Item 2 (§3.8.8, Doc. No. X3J11/88-135 Letter L178 p. 3)

Summary of Issue: Wants `__FUNCTION__` macro for current function name.

Committee Response:
This does not appear to be based on prior art.

This would be hard for preprocessors to implement, since they are not cognizant of function definitions.

Item 3 (§3.8.2, Doc. No. X3J11/88-135 Letter L178 p. 3)

Summary of Issue: Wants automatic generation of header idempotency lock symbols.

Committee Response:
This does not appear to be based on prior art.

Not all header use requires idempotency. Lock symbols can be explicitly constructed by the programmer using `#define`.

Furthermore, implementations that are memory-constrained would have difficulty implementing this feature when a program contains a large number of `#included` files.

Item 4 (§3.5.2.1, Doc. No. X3J11/88-135 Letter L178 p. 4)

Summary of Issue: Proposes `hidden` keyword as alternative to anonymous structures.

Committee Response:
This does not appear to be based on prior art.

There is simply no precedent for this extension to be included in the Standard.

Item 5 (§3.5.2.1, Doc. No. X3J11/88-135 Letter L178 p. 5)

Summary of Issue: Proposes `extract` function for extracting "substructures".

Committee Response:
This does not appear to be based on prior art.

Again, we have no precedent to include this extension in the Standard.

X3J11 Response to Letter L179

Correspondent's Name and Address:

Craig Franklin
Green Hills Software, Inc.
425 East Colorado Street, Suite 710
Glendale, CA 91205

Item 1 (§4.9.5.1, Doc. No. X3J11/88-136 Letter L179 p. 1)

Summary of Issue: Require, or at least allow, `fclose((FILE*)NULL)` to close all streams.

Committee Response:
Extensions are allowed in this regard, but they are not required.

An implementation that chooses to do as you request is standard conforming in this area. §4.1.6 states that an invalid argument to a library function causes undefined behavior; therefore, an `fclose` function that does accept `NULL` as a argument cannot be considered non-conforming.

X3J11 Response to Letter L180

Correspondent's Name and Address:

Arthur David Olson
National Institutes of Health
Building 37, Room 3B03
Bethesda, MD 20892

Item 1 (§1.7, Doc. No. X3J11/88-137 Letter L180 p. 1)

Summary of Issue: Change "any" to "all".

Committee Response:
The Committee believes this is clear enough as is.

Item 2 (§3.8.8, Doc. No. X3J11/88-137 Letter L180 p. 2)

Summary of Issue: Move the meaning of `__STDC__` from a footnote into the body of the Standard.

Committee Response:
This was accepted as an editorial change to the Standard.

The footnote explained this clearly enough where it was, but we have moved it into the text.

Item 3 (§4.1.2, Doc. No. X3J11/88-137 Letter L180 p. 3)

Summary of Issue: Prohibit reservation of names other than what is specified.

Committee Response:
This was accepted as an editorial change to the Standard.

Wording has been added to clarify this issue.

Item 4 (§4.5.2.5, Doc. No. X3J11/88-137 Letter L180 p. 5)

Summary of Issue: There is no definition of "large magnitude".

Committee Response:
This was accepted as an editorial change to the Standard.

We agreed with you.

Item 5 (§4.12.3.5, Doc. No. X3J11/88-137 Letter L180 p. 6)

Summary of Issue: Explicitly allow abbreviations for time zone names.

Committee Response:
This was accepted as an editorial change to the Standard.

Thank you; we have indicated that abbreviations may be returned.

Item 6 (§R1.1, Doc. No. X3J11/88-137 Letter L180 p. 7)

Summary of Issue: The Committee has ***nearly always*** held fast to precedent.

Committee Response:
This proposed editorial change was discussed but not accepted.

We really believe we ***have*** held fast to precedent wherever it was clear and unambiguous. For your examples, we know of implementations that did choose the other way.

Item 7 (§R1.1, Doc. No. X3J11/88-137 Letter L180 p. 8)

Summary of Issue: Warn readers that there are changes described other than as QUIET CHANGES.

Committee Response:
Changes have been made along the lines you suggested.

Item 8 (§R1.1, Doc. No. X3J11/88-137 Letter L180 p. 9)

Summary of Issue: Reword "not to interfere" sentence for clarity.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 9 (§R2.1.1.3, Doc. No. X3J11/88-137 Letter L180 p. 10)

Summary of Issue: Clarify that "§1.7" refers to the Rationale, not the Standard.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 10 (§R3.1.3.4, Doc. No. X3J11/88-137 Letter L180 p. 11)

Summary of Issue: Note that some implementations deviated from K&R with regard to the meaning of unknown \ escapes.

Committee Response:
This proposed editorial change was discussed but not accepted.

We haven't attempted to point out all the places where implementations failed to follow K&R.

Item 11 (§R3.1.7, Doc. No. X3J11/88-137 Letter L180 p. 12)

Summary of Issue: *Only* header names obey certain tokenization rules.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 12 (§R3.5.2.3, Doc. No. X3J11/88-137 Letter L180 p. 13)

Summary of Issue: Note existing practice for incomplete type declaration.

Committee Response:
No change to the existing wording was considered necessary.

Avoiding name conflicts by using unique names is a technique that is available for *any* kind of name, and may sometimes be a good "readability" suggestion. But block-scope names (for structure tags as well as other identifiers) should be available for macro-generated and machine-generated names, as well as names included from headers.

Item 13 (§R3.5.4.3, Doc. No. X3J11/88-137 Letter L180 p. 14)

Summary of Issue: Note variable-argument behavior as a QUIET CHANGE.

Committee Response:
The Committee chose a different approach to deal with this issue.

This isn't a QUIET CHANGE because the program would be invalid. However, the Rationale has been changed to call attention to this problem.

Item 14 (§R3.8.3, Doc. No. X3J11/88-137 Letter L180 p. 15)

Summary of Issue: Drop the `#define void int` example.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 15 (§R4.9.1, Doc. No. X3J11/88-137 Letter L180 p. 16)

Summary of Issue: `OPEN_MAX` should be `FOPEN_MAX` (typo).

Committee Response:
This was accepted as an editorial change to the Rationale.

This was a typographical error; thank you for drawing our attention to it.

X3J11 Response to Letter L181

Correspondent's Name and Address:

James Gardner
Software Development Group
University of Waterloo
Waterloo, Ontario
N2L 3G1
Canada

Item GN1(a) (§1.7, Doc. No. X3J11/88-138 Letter L181 p. 1)

Summary of Issue: Functions that can never succeed should be flagged at translation time.

Committee Response:
Quality of implementation is beyond the scope of the Standard.

An implementation is free to emit any amount of non-diagnostic or diagnostic messages during translation. The quality of the implementation will determine what information the translator produces.

Item GN1(b) (§4.4, Doc. No. X3J11/88-138 Letter L181 p. 1)

Summary of Issue: Unsupported locales should be flagged at translation time.

Committee Response:
This concerns matters beyond the scope of X3J11.

This is an issue for the implementor, not the Standard.

See also our response to your comment on §4.4.1.1 below. The choice of strings for locale names was made to facilitate flexibility in accepting user-specified locale information. An implementation has the responsibility of doing whatever is deemed worthwhile to assist a programmer in this regard.

Item GN2 (§3.1.2.5, Doc. No. X3J11/88-138 Letter L181 p. 2)

Summary of Issue: "Same representation" is not enough.

Committee Response:
Changes have been made along the lines you suggested.

We have added a footnote to clarify that "same representation" is meant to include compatibility of `void *` and `char *` (and signed *vs.* unsigned types) in function arguments. *Full* compatibility would allow the following undesirable combination of declarations at file scope:

```
void *x; char *x;
```

Item 1.6 (§1.6, Doc. No. X3J11/88-138 Letter L181 p. 2)

Summary of Issue: When are diagnostics prohibited?

Committee Response:
Quality of implementation is beyond the scope of the Standard.

In a reasonable implementation one would expect a correct program to be compiled quietly, but this does not preclude diagnostics where the implementor thinks they are appropriate, even for implementation-defined behavior.

Item 2.1.1.3 (§2.1.1.3, Doc. No. X3J11/88-138 Letter L181 p. 3)

Summary of Issue: Are diagnostics required for use of syntax extensions?

Committee Response:
This was considered a request for information, not an issue.

An implementation must diagnose a program that uses a syntax extension.

Item 2.1.2.3 (§2.1.2.3, Doc. No. X3J11/88-138 Letter L181 p. 3)

Summary of Issue: Allow optimizations to cause overflow exceptions.

Committee Response:
The Committee has voted against this idea.

An optimizer is free to rearrange code as long as it does not introduce overflows. This decision was made in the interest of greater reliability. Programmers do not readily expect code to start generating overflow traps when optimized.

Item 3.1.8 (§3.1.8, Doc. No. X3J11/88-138 Letter L181 p. 4)

Summary of Issue: Preprocessing numbers have unusual properties.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

At the September 1988 meeting the Committee discussed this again but decided against changing it. Trying to refine the notion of preprocessing number tokens at this stage is likely to introduce as many surprises as are fixed. The few cases of existing code that would be broken were not considered to present a significant practical problem.

Some Committee members insisted that they *wanted* preprocessor numbers to be "greedier" than one might think appropriate, so that (for example) binary constants such as `0b1001` could be supported as an extension, and that any change other than fixing *just* the "broken code" case would not be acceptable.

Item 3.2.1.5 (§3.2.1.5, Doc. No. X3J11/88-138 Letter L181 p. 4)

Summary of Issue: Mixed-signedness comparison should be implementation defined.

Committee Response:
The Committee has voted against this idea.

The Standard reflects widespread existing practice in this regard.

The conversion from `signed` to `unsigned int` is widespread existing practice. A knowledgeable programmer can always write the expanded test

```
(i<0 || i<u)
```

if that is the desired behavior.

Item 3.3 (§3.3, Doc. No. X3J11/88-138 Letter L181 p. 5)

Summary of Issue: Supposed problem with order of evaluation.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

This expression was not intended to be well-defined. A sequence point occurs after the evaluation of the left operand of each comma, but the relative order of the assignments and the other operands is unspecified. When operators that create sequence points are mixed with operators that do not, only a partial ordering of the whole expression is implied.

Item 3.3.6 (§3.3.6, Doc. No. X3J11/88-138 Letter L181 p. 5)

Summary of Issue: Allow pointer subtraction for punned `char *`s within an object.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The old model of a flat address space has been modified to allow implementors the freedom to provide more efficient implementations for their architectures. However, within a *single object* the kind of pointer arithmetic you want is allowed, by casting to and from a character pointer. (See §3.3.4.)

Item 3.3.15 (§3.3.15, Doc. No. X3J11/88-138 Letter L181 p. 5)

Summary of Issue: Disallow `NULL` being defined as `(void *)0`.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The May 1988 draft on page 50, lines 28-9 indicates that a null pointer constant, whether `0` or `(void *)0`, will convert the same way. Hence this is not a problem.

Item 3.5.2.1 (§3.5.2.1, Doc. No. X3J11/88-138 Letter L181 p. 6)

Summary of Issue: Require bit fields to be minimum possible size.

Committee Response:
Quality of implementation is beyond the scope of the Standard.

It is up to an implementor to choose how much memory bit fields consume. The Committee chose not to be more specific.

Item 3.5.2.3 (§3.5.2.3, Doc. No. X3J11/88-138 Letter L181 p. 6)

Summary of Issue: Completing a structure tag should complete an associated `typedef`.

Committee Response:
The request is reflected in the current draft.

Our previous response was erroneous; apparently some mistakes were made in drafting the examples. The second example,

```
typedef struct s T;
struct s {int x;};
T z;
```

was correct, and `z` has a complete type. The third example was indeed syntactically incorrect.

Item 3.5.3 (§3.1.2.5, Doc. No. X3J11/88-138 Letter L181 p. 7)

Summary of Issue: Are qualifiers automatically stripped out of function arguments?

Committee Response:
Changes have been made along the lines you suggested.

Qualifiers are stripped from the type category (formerly called "top type") only in converting from an lvalue to an "rvalue" (the value of an expression). We have specified in §3.1.2.5 that pointers to differently qualified versions of compatible types have the same representation and alignment.

`volatile char *` arguments cannot portably be passed to `printf` because it does not use `volatile` pointers to access a string; also it doesn't matter what the type of the pointer, the declared type of the actual object is what matters. In this case, a truly volatile object cannot be safely accessed *via* a pointer within the `printf` family (in actuality, through *none* of the standard

library functions).

Item 3.5.4.2 (§3.5.4.2, Doc. No. X3J11/88-138 Letter L181 p. 7)

Summary of Issue: Can incomplete arrays be completed later?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

`static` declarations must be complete at each declaration (see §3.7.2 **Semantics**). The third example, involving a global object, can be completed later.

Item 3.5.7 (§3.5.7, Doc. No. X3J11/88-138 Letter L181 p. 8)

Summary of Issue: Use of string literals for initializers is inconsistent.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

For unsigned character pointers, the initializer can be cast to the correct type. For arrays, it is simply a shorthand for the initializer `{'h', 'e', 'l', ...}`, and as such, the variation of character does not matter.

Item 4.1.2 (§4.1.2, Doc. No. X3J11/88-138 Letter L181 p. 8)

Summary of Issue: Standard headers should be allowed to also define names in the space reserved for implementations.

Committee Response:
Changes have been made along the lines you suggested.

The section was reworded to clarify both the issue of double-underscore names and other reserved identifiers in headers. Your suggestion was taken into account in drafting these new words.

Item 4.1.6 (§4.1.6, Doc. No. X3J11/88-138 Letter L181 p. 8)

Summary of Issue: Some application declarations of library functions will be rejected.

Committee Response:
This was considered a comment rather than an issue.

Your observations are correct. An implementation is free to diagnose any mismatch between a user-supplied declaration and the standard declaration.

Item 4.3 (§4.3, Doc. No. X3J11/88-138 Letter L181 p. 9)

Summary of Issue: `<ctype.h>` macros should be made safe for `signed char` arguments.

Committee Response:
The Committee has voted against this idea.

`char` arguments can always be cast to `unsigned char` type before calling a `<ctype.h>` function. Indeed, since one cannot distinguish in classic signed `char` implementations between `EOF` (with value −1) and the 8-bit char `'377'`, whenever there there might be a `'377'` value for a `char` the argument *must* be cast to an `unsigned char` for such implementations.

Item 4.4.1.1 (§4.4.1.1, Doc. No. X3J11/88-138 Letter L181 p. 9)

Summary of Issue: `setlocale` errors should be flagged at translation time.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

See also our response to comment GN1(b) above. We decided to give implementors latitude in allowing users to enter locale strings. An implementation may supply a set of macros if it desires.

Item 4.8 (§4.8, Doc. No. X3J11/88-138 Letter L181 p. 10)

Summary of Issue: State that a copy of a `va_list` may be safely used.

Committee Response:
The Committee discussed this proposal but decided against it.

A program cannot rely on being able to scan an argument list with a copy of the `va_list` object. So, while it is possible to make a copy using `memcpy`, there is no portable use for that copy.

Item 4.9.1(a) (§4.9.1, Doc. No. X3J11/88-138 Letter L181 p. 10)

Summary of Issue: `FOPEN_MAX` definition is incorrect.

Committee Response:
The Committee has voted against this idea.

The Standard reflects the result of previous discussion of this issue.

We decided after considerable debate to leave the description of `FOPEN_MAX` unchanged. We felt that the additional environmental constraints were understood.

Item 4.9.1(b) (§4.9.1, Doc. No. X3J11/88-138 Letter L181 p. 10)

Summary of Issue: Is `FILENAME_MAX` somehow related to `tmpnam`?

Committee Response:
The Committee discussed this proposal but decided against it.

`L_tmpnam` is unrelated to `FILENAME_MAX` and usually is much shorter. The `tmpnam` function uses `L_tmpnam` to determine its buffer size.

Item 4.10.7.2 (§4.10.7.2, Doc. No. X3J11/88-138 Letter L181 p. 10)

Summary of Issue: How does one reset an undefined shift state?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

As stated on page 155, line 22 of the May 1988 draft, the shift state is reset when a null string argument is passed to the relevant function.

Item 4.11.4.1 (§4.11.4.1, Doc. No. X3J11/88-138 Letter L181 p. 11)

Summary of Issue: List operations that may stomp on structure holes.

Committee Response:
This was accepted as an editorial change to the Standard.

We removed the misleading words in the footnote.

Potentially any operation can modify the "holes" in a structure. Since there is no portable way of inspecting them, we place no constraints on the way they behave. Think of holes as effectively "volatile".

Item 4.11.5.8 (§4.11.5.8, Doc. No. X3J11/88-138 Letter L181 p. 11)

Summary of Issue: If no token is found on the first call to `strtok`, it appears that `NULL` need not be returned on subsequent calls.

Committee Response:
This was considered a request for information, not an issue.

Yes, if no token is present on the first call, `strtok` may fail to return `NULL` on subsequent calls. Such usage falls in the category of "unspecified behavior".

Item 4.12.2.1 (§4.12.2.1, Doc. No. X3J11/88-138 Letter L181 p. 11)

Summary of Issue: Explicitly state that the only accurate way to use `clock` is to relate values to one obtained at the beginning of a program.

Committee Response:
This was accepted as an editorial change to the Standard.

A footnote was added to clarify that `clock` must be called twice and only the difference of two calls can be meaningfully used.

X3J11 Response to Letter L182

Correspondent's Name and Address:

Jens Kolind
Dansk Standardiseringsrad
Danish Standards Association
Medlem AF The International Organization for Standardization
ISO OG Comite Europeen de Normalisation
CEN

Item 1 (§2.2.1.1, Doc. No. X3J11/88-141 Letter L182 p. 1)

Summary of Issue: Proposed solution to character set (trigraph) issue.

Committee Response:
The Committee discussed this proposal but decided against it.

This proposal was submitted as L177 (X3J11/88-134); please see our response to that letter for a detailed discussion of this issue.

The Committee discussed this solution but decided that it was too radical a change at this stage and further that it did not represent significant prior art. A counterproposal was offered by one Committee member (see document X3J11/88-108) which addresses some of the readability issues.

X3J11 Response to Letter L183

Correspondent's Name and Address:

Clement Victor Clarke
C.C.S.-Jol Pty. Ltd.
31 Queen Street
Melbourne 3000 Victoria
Australia

Item 1 (§4.11, Doc. No. X3J11/88-142 Letter L183 p. 1)

Summary of Issue: Add built-in string operators.

Committee Response:
The Committee discussed this proposal but decided against it.

Over the years this has been brought up and discussed several times. The Committee feels that this would unduly enlarge the language and is not based on prior art.

Good optimizers can often eliminate the function call overhead. Also note that C gives the programmer more responsibility than other languages; it is possible to define descriptor-based strings and functions to operate on them if desired. One common trick you may be interested in is

```
#define EQ(a,b) (*(a) == *(b) && strcmp(a,b) == 0)
```

Item 2(a) (§3.1.4, Doc. No. X3J11/88-142 Letter L183 p. 2)

Summary of Issue: Need type of string literal that "properly" compares strings containing trailing spaces.

Committee Response:
This does not appear to be based on prior art.

Many programmers would not agree that a string containing trailing spaces is the "same" as a string without them; that would be highly application dependent.

Item 2(b) (§3.1.4, Doc. No. X3J11/88-142 Letter L183 p. 2)

Summary of Issue: Extend meaning of single quotes to mean string literal using machine instructions.

Committee Response:
This does not appear to be based on prior art.

Character constants, using single quotes, are very different from string literals (even if they contain several characters).

Item 3 (§3.5.7, Doc. No. X3J11/88-142 Letter L183 p. 2)

Summary of Issue: Should be able to initialize structure members at the point they are declared.

Committee Response:
A specific proposal is needed before action can be taken.

This does not appear to be based on prior art. (The prior art we look at is C, not PL/I.)

Item 4 (§3.3.2.3, Doc. No. X3J11/88-142 Letter L183 p. 3)

Summary of Issue: Need `with`, to set structure for subsequent member accesses.

Committee Response:
A specific proposal is needed before action can be taken.

This does not appear to be based on prior art. (The prior art we look at is C, not Pascal.)

Item 5 (§4.7, Doc. No. X3J11/88-142 Letter L183 p. 3)

Summary of Issue: Wants a facility like PL/I `ON` condition.

Committee Response:
A specific proposal is needed before action can be taken.

Refer to point 3 above.

Item 6 (§3.3.2.3, Doc. No. X3J11/88-142 Letter L183 p. 3)

Summary of Issue: Need `based`, to set pointer for structure access.

Committee Response:
A specific proposal is needed before action can be taken.

Refer to point 3 above.

Which method is "clearer" depends on one's previous experience. We generally prefer the C approach.

Item 7 (§3.1.2.2, Doc. No. X3J11/88-142 Letter L183 p. 3)

Summary of Issue: Wants PL/I `EXTERNAL` variables.

Committee Response:
A specific proposal is needed before action can be taken.

Refer to point 3 above.

Again, which method is "better" depends on one's previous experience. We have no problem with the C approach.

Item 8 (§3.5.4.3, Doc. No. X3J11/88-142 Letter L183 p. 3)

Summary of Issue: Wants PL/I-style function declaration.

Committee Response:
A specific proposal is needed before action can be taken.

Refer to point 3 above.

Note that C compilers will diagnose function definitions that disagree with a corresponding prototype (assumed to be in scope).

Item 9 (§3.3.2.2, Doc. No. X3J11/88-142 Letter L183 p. 3)

Summary of Issue: Function linkage should use fixed parameter blocks.

Committee Response:
A specific proposal is needed before action can be taken.

Refer to point 3 above.

C implementations may choose from a variety of techniques for implementing function linkage.

Note that C uses "call by value", with each activation record receiving its own copy of its value parameters. This is an essential, and to some attractive, feature of C that cannot be changed in the process of standardization.

X3J11 Response to Letter L184

Correspondent's Name and Address:

Mr Tamura Jones
P.O.Box 11258
2301 EG Leiden
The Netherlands

Item LIBRARY (§A.7, Doc. No. X3J11/88-144 Letter L184 p. 1)

Summary of Issue: `<locale.h>` was omitted from the Index.

Committee Response:
The request is reflected in the current draft.

[NOTE — It was difficult to respond adequately to every issue raised in this letter, as the Committee was unable to determine which draft was being reviewed. We made our best guess as to the exact issues.]

Item ESCAPE (§2.2.2, Doc. No. X3J11/88-144 Letter L184 p. 1)

Summary of Issue: Add `\l` for line feed.

Committee Response:
This does not appear to be based on prior art.

Additionally, there would be significant problems for some implementations (those that do not have a line-feed distinct from new-line).

Item LAYOUT (§General, Doc. No. X3J11/88-144 Letter L184 p. 1)

Summary of Issue: Put section numbers in page headers.

Committee Response:
The request is reflected in the current draft.

Item preproc (§3.8.5, Doc. No. X3J11/88-144 Letter L184 p. 1)

Summary of Issue: Add `#message` for non-fatal warning.

Committee Response:
The Committee has voted against this idea.

There is not a large enough body of prior art for this. Besides, the Standard does not distinguish between levels of diagnostic messages.

Item backspace (§2.2.2, Doc. No. X3J11/88-144 Letter L184 p. 1)

Summary of Issue: Is `\b` a deleting backspace?

Committee Response:
This concerns matters beyond the scope of X3J11.

Most systems do not provide separate "deleting" and "non-deleting" backspace characters.

Item entry (§3.1.1, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: Reserve keywords `entry`, `asm`, and `fortran`.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

The benefits of reserving these keywords in the Standard was insufficient for their inclusion. An implementation can provide these keywords as an extension, providing it adheres to the diagnostic and name space requirements of a conforming implementation. Alternatively, it could provide these keywords in a non-conforming version.

Item __STDC__ (§3.8.8, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: `__STDC__`'s absence is not defined as being non-conforming.

Committee Response:
This concerns matters beyond the scope of X3J11.

Behavior of non-conforming implementations is not within the scope of the Standard. The example given is allowed in ANSI C.

Item 11.25 (§2.2.2, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: Shouldn't "at the final position" be "beyond the final position"?

Committee Response:
This was considered a request for information, not an issue.

No, "at the final position" is correct. It may not be defined what "beyond the final position" means.

Item 12 (§2.2.3, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: Self-modifying code is forbidden.

Committee Response:
This was considered a request for information, not an issue.

You are correct.

Item 12.35 (§2.2.4.1, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: What is a "logical source line"?

Committee Response:
This was considered a request for information, not an issue.

The 509 length limit is before macro expansion. A "logical source line" is the result of line splicing in translation phase 2.

Item 12.39 (§2.2.4.1, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: Do the 257 `case` labels include `default`?

Committee Response:
This was considered a request for information, not an issue.

No, they do not.

Item 13.14 (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 2)

Summary of Issue: Mention that 128 is a nonportable value for a `signed char`.

Committee Response:
This was considered a comment rather than an issue.

That is not the point of this section of the Standard.

Item 13.36 (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Minus sign should be plus sign (typo).

Committee Response:
This was considered a comment rather than an issue.

As you suspected, ISO's copier lost some information.

Item 14.24 (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Use a table for ease of reading.

Committee Response:
The request is reflected in the current draft.

Item 14.29 (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Why are no values given?

Committee Response:
This was considered a request for information, not an issue.

These macros do not have required minimum or maximum values, merely implementation-dependent information.

Item 15.1 (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Why are no values given?

Committee Response:
This was considered a request for information, not an issue.

These macros do not have required minimum or maximum values, merely implementation-dependent information.

Item 17.30 (§3.1, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: "Except as they separate tokens" is bad English.

Committee Response:
The request is reflected in the current draft.

Item 17.34 (§3.1, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Should say "of characters *thereafter*".

Committee Response:
The Committee believes this is clear enough as is.

The context makes this clear enough.

Item 17.35 (§3.1, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Line number was missing.

Committee Response:
The request is reflected in the current draft.

Item 18.43 (§3.1.2, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Explain "undefined behavior" for differing identifiers.

Committee Response:
This was considered a request for information, not an issue.

If identifiers differ in a non-significant character, the behavior is undefined. See §3.1.2, which reflects previous clarification of this issue.

Item 19.25 (§3.1.2.1, Doc. No. X3J11/88-144 Letter L184 p. 3)

Summary of Issue: Can a pointer be assigned its own address?

Committee Response:
This was considered a request for information, not an issue.

The answer is "yes". An initializer is not part of the declarator.

Item 20 (§3.1.2.3, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Want to be able to cast a label to a `void` function.

Committee Response:
The Standard must accommodate a variety of architectures.

Many, if not most, architectures have different requirements for call points and branch points.

Item short(a) (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Why is `short` so big?

Committee Response:
The Standard reflects widespread existing practice in this regard.

The overwhelming existing practice assumes that `short` values are at least 16 bits. Also, support for `int` values greater than 16 bits could not be universally obtained.

Item short(b) (§2.2.4.2, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Make C strongly typed.

Committee Response:
This does not appear to be based on prior art.

There is virtually no support for this idea.

Item 22 (§3.1.2.5, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: `void` should not be an enumerated type.

Committee Response:
The request is reflected in the current draft.

Item 22.10 (§3.1.2.5, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Can a function really return a structure?

Committee Response:
This was considered a request for information, not an issue.

The answer is "yes".

Item 25 (§3.1.4, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Are strings preprocessed?

Committee Response:
This was considered a request for information, not an issue.

The answer is "no", as stated in §3.8.3.

Item 27(a) (§3.1.6, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Line numbers were missing.

Committee Response:
The request is reflected in the current draft.

Item 27(b) (§3.1.6, Doc. No. X3J11/88-144 Letter L184 p. 4)

Summary of Issue: Should say "but *(as a punctuator)* does not specify".

Committee Response:
The Committee believes this is clear enough as is.

The next sentence goes on to state that a given symbol is sometimes a punctuator and sometimes a separator.

Item escape (§2.2.2, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: Add "universal" alert `\g`.

Committee Response:
This proposal contains insurmountable technical errors.

Not every character set defines a BEL character, and only for some (such as ASCII) would a correspondence with "control G" make any sense.

Item 20.5 (§3.1.2.2, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: Why is a linkage clash undefined behavior instead of an error?

Committee Response:
It was decided to allow implementors freedom in this regard.

An implementation is free to diagnose this condition, or to interpret it as an extension.

Item 22 (§3.1.2.5, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: Why are constants defined as a separate type?

Committee Response:
This is a misinterpretation of correct wording in the document.

The term "constant" refers to tokens that represent numeric constant values. The `const` type qualifier is used to denote objects that are not to be modified.

Item 27(a) (§3.1.6, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: Line numbers were missing.

Committee Response:
The request is reflected in the current draft.

This is a duplicate of a previous item.

Item 27(b) (§3.1.6, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: # can occur other than in preprocessing directives.

Committee Response:
This is a misinterpretation of correct wording in the document.

The constraint refers to the # *token*. If the # character appears within a string token, it is not interpreted as a # token.

Item 28 (§3.1.9, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: On which pass are comments scanned?

Committee Response:
This was considered a request for information, not an issue.

Comments are interpreted as white space in translation phase 3, after line splicing and before processing of preprocessor directives.

Item 27(c) (§3.2.2.3, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: How can a null pointer be *guaranteed* not to point to a valid object or function?

Committee Response:
This is a misinterpretation of correct wording in the document.

The implementation is required to provide a bit pattern that is guaranteed to be distinguishable from all valid pointers. This bit pattern is not required to be all zeros.

Item 33.40 (§3.3.2.1, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: Rewrite paragraph for improved clarity.

Committee Response:
Changes have been made along the lines you suggested.

Item 33.47 (§3.3.2.1, Doc. No. X3J11/88-144 Letter L184 p. 5)

Summary of Issue: Minus should be plus (typo).

Committee Response:
This was considered a comment rather than an issue.

This is presumed to be a copier problem.

Item 35 (§3.3.2.3, Doc. No. X3J11/88-144 Letter L184 p. 6)

Summary of Issue: Explain "common initial sequence" requirement.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

There are other benefits that accrue from such a constraint, such as structure members being laid out reproducibly.

[NOTE: We were unable to identify the sentence you wanted clarified (see the note concerning this attached to our response to your first issue); we trust that the revised wording in this vicinity in more recent drafts will resolve your intended issue. See also our response to another comment concerning "page 35" below.]

Item 38 (§3.3.3.4, Doc. No. X3J11/88-144 Letter L184 p. 6)

Summary of Issue: `sizeof(function)` should return the size of the function pointer.

Committee Response:
The Standard provides another way to do this.

It is easy enough to apply an `&` operator to the function name.

Item 43 (§3.3.9, Doc. No. X3J11/88-144 Letter L184 p. 6)

Summary of Issue: Unequal pointers should not be allowed to point to the same object.

Committee Response:
Changes have been made along the lines you suggested.

Item 44.24 (§3.3.12, Doc. No. X3J11/88-144 Letter L184 p. 6)

Summary of Issue: "Level" is not defined.

Committee Response:
The Committee chose a different approach to deal with this issue.

The provisions referring to "same level" have been deleted.

Item 47.20 (§3.3.17, Doc. No. X3J11/88-144 Letter L184 p. 6)

Summary of Issue: Is more than one comma allowed?

Committee Response:
This was considered a request for information, not an issue.

The answer is "yes, but not *adjacent* commas".

Item 35 (§3.3.2.2, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: "Default type" is a strange and wrong concept.

Committee Response:
This proposal would invalidate too much existing source code.

Eliminating implicit declaration of functions would break most existing C code.

Item calling (§3.3.2.2, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: Unused function return value should be an error.

Committee Response:
This proposal would invalidate too much existing source code.

Requiring an explicit `void` cast would break almost every existing C program, as it would be required (for example) on every call to the `printf` function.

Item 53 (§3.5.2.1, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: Declaration that specifies only a structure or union tag is a kludge.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

It allows pure block scope for all identifiers in C; it can be used to ensure forward reference in a forward declaration.

Item 54 (§3.5.2.2, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: Explicitly specify operations allowed for enumeration objects.

Committee Response:
The Committee believes this is clear enough as is.

As stated in §3.2.1.1, an enumeration expression can be used in any context where an integer expression can be used.

Item 57 (§3.5.4, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: Why a limit of 12 declarators?

Committee Response:
The Standard reflects the result of previous discussion of this issue.

This is a duplicate of the translation limit specified in §2.2.4.1.

As with all limits, the number of declarators is a compromise between programmer utility and implementation burden. The 12-declarator limit easily covers FORTRAN's array dimensions rule, and also allows for the classic C compiler internal representation for a type within a 32-bit datum: 4 bits for the basic type, and up to 12 sets of 2-bit declarator descriptors ("array of", "function returning", "pointer to", plus a special "terminating" value signifying "no more declarators").

Item 58 (§3.5.4.3, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: Can a function return a structure?

Committee Response:
This was considered a request for information, not an issue.

The answer is "yes".

Item 58.fn (§3.5.4.3, Doc. No. X3J11/88-144 Letter L184 p. 7)

Summary of Issue: Footnote implies that parameters can follow an ellipsis.

Committee Response:
The request is reflected in the current draft.

Item 65 (§3.6.1, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: Wants a label to be callable.

Committee Response:
The Standard must accommodate a variety of architectures.

See our response to your comment concerning page 20.

Item 66 (§3.6.3, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: Is the semicolon necessary after a label on a null statement?

Committee Response:
This was considered a request for information, not an issue.

A label must be followed by a statement, and therefore cannot be followed by a } token.

Item 71.35 (§3.7.1, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: Restore missing "and" (typo).

Committee Response:
The Committee chose a different approach to deal with this issue.

This has previously been reworded for improved clarity.

Item 73 (§3.7.1, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: Objects to automatic conversion of function parameter to pointer.

Committee Response:
The Standard reflects the base document in this regard.

The K&R rule that states "since all you can do with function names is either call the function or takes its address, when a function name is used in a context other than a call, it is automatically turned into a pointer to the function" naturally causes this behavior.

Item 77.8 (§3.8.1, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: In *what* order is each directive's condition checked?

Committee Response:
This was considered a request for information, not an issue.

They are checked in the order that they are encountered.

Item 79.26 (§3.8.3, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: What does it mean to "skip" pairs of parenthesis tokens?

Committee Response:
This was considered a request for information, not an issue.

Macro argument collection continues until either a right parenthesis not matched by an intervening left parenthesis or a comma not within nested parentheses is encountered.

Item 79 (§3.8.3, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: What is a preprocessing token?

Committee Response:
This was considered a request for information, not an issue.

As specified in the grammar in §3.1, a preprocessing token is one of:

- header name
- identifier
- preprocessing number
- character constant
- string literal
- operator
- punctuator
- other non-white-space character.

Item 80 (§3.8.3, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: Need examples.

Committee Response:
This was considered a comment rather than an issue.

Examples of macro expansion, token pasting, and string substitution are given.

Item 81 (§3.8.3.5, Doc. No. X3J11/88-144 Letter L184 p. 8)

Summary of Issue: Examples are too few and too complicated.

Committee Response:
The Standard is not intended to double as a tutorial.

The examples are intended to illustrate some of the subtleties of the rules, not to instruct programmers.

Item 82 (§3.8.3.5, Doc. No. X3J11/88-144 Letter L184 p. 9)

Summary of Issue: Are the "invalid" examples individually or collectively invalid?

Committee Response:
This was considered a request for information, not an issue.

Within the scope of all the valid examples, each of the invalid examples is invalid.

X3J11 Response to Letter L185

Correspondent's Name and Address:

T. Inose
IPSJ/ITSCJ/SC22/C WG Chairman
Information Technology Standards Commission of Japan
Information Processing Society of Japan
Kikai Shinko Building No. 3-5-8 Shiba-Koen Minato-ku
Tokyo 105
JAPAN

Item P.1 (§3.1.3.4, Doc. No. X3J11/88-145 Letter L185 p. 3)

Summary of Issue: The term "character" is misused.

Committee Response:
The Committee chose a different approach to deal with this issue.

We changed two formerly imprecise uses of "character" (on page 28, lines 35 and 39 of the May 1988 draft) to "character or wide character". Note that "wide character" is defined later in §3.1.3.4 under **Semantics**.

We believe that the use of "character" is clear in the document as is; see §1.6, which has been revised to further reduce confusion. "Character" means just one byte, even in the context of a multibyte character sequence.

For a integer character constant, "character" in the context of escape sequences really is intended to refer to an individual byte, not a whole multibyte character sequence; in the case of a wide character constant, an escape sequence does represent a multibyte character.

Item P.2 (§3.1.3.4, Doc. No. X3J11/88-145 Letter L185 p. 4)

Summary of Issue: Escape sequence in a character constant should be constrained to fit within a character.

Committee Response:
This was accepted as an editorial change to the Standard.

We have changed the Standard to clarify this issue.

Item P.3 (§3.1.3.4, Doc. No. X3J11/88-145 Letter L185 p. 5)

Summary of Issue: Need semantics for multibyte character in a character constant or string literal.

Committee Response:
The Committee believes this is clear enough as is.

We believe that the use of multibyte character is clear in the document as is.

We do not wish to specify the numerical significance of the bytes within a multibyte character. The specification for string literal has been reworded.

We appreciate your pointing out the difficulties that existed with the previous definitions of "character" and "byte". See the revised definitions in §1.6, which clarify that these terms are essentially synonymous.

Item P.4 (§4.8, Doc. No. X3J11/88-145 Letter L185 p. 7)

Summary of Issue: `va_list` must also be defined in `<stdio.h>`.

Committee Response:
This is an issue for the implementor, not the Standard.

`va_list` need not, indeed must not, be defined in `<stdio.h>`.

`typedefs` do not create new types, merely synonyms for other types. In `<stdio.h>` function prototypes an implementation has to use the type that it would define for `va_list` in `<stdarg.h>`. For example, if a `void *` is chosen for `va_list`, the prototype for `vprintf`, `vfprintf`, and `vsprintf` would just use `void *`.

Item P.5 (§4.9.3, Doc. No. X3J11/88-145 Letter L185 p. 8)

Summary of Issue: Add a footnote explaining what "character number zero" means.

Committee Response:
This was considered a comment rather than an issue.

We believe that the text is clear enough as it stands.

Item P.6 (§4.10.7, Doc. No. X3J11/88-145 Letter L185 p. 9)

Summary of Issue: Clarify treatment of shift codes.

Committee Response:
This was accepted as an editorial change to the Standard.

We have made changes in this area to explain that shift sequences do not produce a separate wide character, but are grouped with an adjacent multibyte sequence.

We have added a footnote to the introduction to these functions (§4.10.7) that notes the Committee's intent in this regard.

X3J11 Response to Letter P01

Correspondent's Name and Address:

Alan Mycroft
University of Cambridge
Computer Laboratory
New Museums Site
Corn Exchange Street
Cambridge CB2 3QG
UK

Item 1(a) (§3.2.1.1, Doc. No. X3J11/88-111 Letter P01 p. 1)

Summary of Issue: Clarify where promotion of `char` to `int` occurs.

Committee Response:
Changes have been made along the lines you suggested.

The integral promotions of §3.2.1.1 occur only in those places where they are explicitly mentioned, and thus they do not occur in `sizeof` context or in evaluating the comma operator. The Committee reviewed this definition, and reaffirmed it. However, a footnote has been added to §3.2.1.1 to clarify that the integral promotions are done only where explicitly specified.

Item 1(b) (§3.2.2.1, Doc. No. X3J11/88-111 Letter P01 p. 2)

Summary of Issue: Clarify conversion of array to pointer in comma context.

Committee Response:
No change to the existing wording was considered necessary.

When an lvalue is evaluated, lvalue to value-stored-in-the-designated-object conversions are performed. The discussion in §3.2.2.1 specifies what the result type is (pointer to array, pointer to function, or the type of the lvalue). The comma operator is not one of the operators that prevents changing an array lvalue into a pointer, and it is not one of the operators that cause the integral promotions to be applied.

The Committee deliberately decided on this apparently inconsistent behavior; one reason is that not converting an array to a pointer as the right-hand operand for comma would invalidate an expression like `p = (0,a)`.

Since the comma operator removes lvalueness, if an "array rvalue" were to result, it would not be useful since it could only be accessed *via* a pointer, but the pointed-to object's lifetime would have expired. (See our response to letter P06, item 3.)

Item 2 (§3.1.2, Doc. No. X3J11/88-111 Letter P01 p. 2)

Summary of Issue: "Nondigit" is a misleading name for non-digit identifier characters.

Committee Response:
No change to the existing wording was considered necessary.

It is agreed that "nondigit" is perhaps not the best choice of name. However, we are not any more pleased with the suggested replacements.

Item 3 (§2.2.4.2, Doc. No. X3J11/88-111 Letter P01 p. 2)

Summary of Issue: Specify types of constant macros.

Committee Response:
This was accepted as an editorial change to the Standard.

§2.2.4.2 has been changed to specify that, except for `CHAR_BIT` and `MB_LEN_MAX`, the `<limits.h>` macros shall expand to expressions that have the same type as would an object of the corresponding type after application of the integral promotions.

Item 4 (§3.2.1.5, Doc. No. X3J11/88-111 Letter P01 p. 2)

Summary of Issue: Is it really intended that unsigned arithmetic can be done wrong?

Committee Response:
This was accepted as an editorial change to the Standard.

§3.2.1.5 has been changed to indicate that "greater precision and range" is intended to apply only to floating types. Unsigned arithmetic operations must be done according to the abstract machine definition, and that has always been the Committee's intention.

Item 5 (§3.3.6, Doc. No. X3J11/88-111 Letter P01 p. 3)

Summary of Issue: Paragraph on adding an integer to a pointer is imprecise.

Committee Response:
This was accepted as an editorial change to the Standard.

Subscripting and pointer arithmetic are required to work correctly, and therefore an implementation is required to do the widening described if necessary. A sentence has been added to the **Semantics** section of §3.3.6 which requires that valid pointer arithmetic does not produce an overflow.

Item 6 (§3.3.5, Doc. No. X3J11/88-111 Letter P01 p. 3)

Summary of Issue: Result of exact division of negative quantities is not specified.

Committee Response:
This was accepted as an editorial change to the Standard.

The arithmetically correct result is expected. The wording of §3.3.5 has been changed to clarify this.

Item 7 (§3.5.2.2, Doc. No. X3J11/88-111 Letter P01 p. 3)

Summary of Issue: Type choice for **enum** is inadequately specified.

Committee Response:
This was considered a request for information, not an issue.

a. Yes, the range of values in an **enum** can determine the size.

b. The Standard does not include an incomplete-type form of **enum** declaration, and thus there can be no such forward reference.

c. It is possible (though it seems unlikely) for an implementation to remap the values in an enumeration. However, the implementation would be required to make this remapping invisible for all uses of an enumerated type value as an arithmetic quantity.

Item 8 (§3.5.2.3, Doc. No. X3J11/88-111 Letter P01 p. 3)

Summary of Issue: Possible problem with `typedef` of incomplete types.

Committee Response:
This was considered a comment rather than an issue.

The Committee's previous response was incorrect; the syntax does *not* allow a type name to be used everywhere that "`struct` *tag*" is allowed. We apologize for the error.

Item 9 (§3.5.3, Doc. No. X3J11/88-111 Letter P01 p. 4)

Summary of Issue: Remove useless sentence.

Committee Response:
No change to the existing wording was considered necessary.

Yes, the sentence is redundant, but the point is deemed important enough to be stated explicitly.

Item 10 (§3.5, Doc. No. X3J11/88-111 Letter P01 p. 4)

Summary of Issue: Add "at its outermost level".

Committee Response:
No change to the existing wording was considered necessary.

The phase "... at its outermost level" is an unnecessary redundancy. (If we had said "contain" instead of "declare", then the constraint would indeed have been ambiguous.) In your example the items being declared are declared by an inner declaration, not the outer declaration which is still in violation of the constraint in question.

Item 11 (§3.5.4, Doc. No. X3J11/88-111 Letter P01 p. 4)

Summary of Issue: Use of type names in prototype parameters is still unclear.

Committee Response:
This is a misinterpretation of correct wording in the document.

The declaration `typedef const foo;` is by definition the same as `typedef int const foo;` (as the `int` is automatically supplied as the default type specifier). Thus, there is no way one could presume that in `extern f(int *foo);` the identifier `foo` is other than a name for the parameter with type pointer to `int`. In fact, the grammar does not allow a type name to follow a `*` (unless the type name is being redeclared, in which case it is no longer a type name).

Item 12 (§3.7.1, Doc. No. X3J11/88-111 Letter P01 p. 4)

Summary of Issue: Deprecate the omission of all declaration specifiers in function definitions.

Committee Response:
The Committee has voted against this idea.

The proposal was considered mostly an issue of style. The usage is very common and the Committee felt there was little value in deprecating it.

The fundamental difference between making old-style functions obsolescent and making a lack of declaration specifiers obsolescent is that the language specified by the Committee has two ways of declaring and defining functions, and we would rather the language eventually have just one. This is what we mean to say by obsoleting the old-style form.

Item 13 (§3.8.1, Doc. No. X3J11/88-111 Letter P01 p. 4)

Summary of Issue: Why lock out `#if sizeof` ... and similar extensions?

Committee Response:
The Standard reflects the result of previous discussion of this issue.

The Committee feels that it is important to maintain the separation of preprocessing and other phases of translation. Since the Standard allows the macro definition of identifiers that happen to look like keywords, it would be inconsistent not to guarantee that they can be used in `#if` expressions.

Item 14 (§3.8.3.2, Doc. No. X3J11/88-111 Letter P01 p. 5)

Summary of Issue: Allow # to insert white space to separate certain tokens.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

The third example is the only correct interpretation. While the result may not be optimal, this is not sufficient cause to change currently defined behavior.

The Committee recognizes that almost all existing implementations will not meet the Standard criteria. Rather than adopt either a completely character-based or a token-based approach as reflected in current practice, the Committee developed a compromise which it feels combines the best features of both.

Item 15(a) (§4.1.2, Doc. No. X3J11/88-111 Letter P01 p. 5)

Summary of Issue: Change "reference" to "reference or definition".

Committee Response:
The Committee chose a different approach to deal with this issue.

Words have been added to §4.1.2 of the Standard that more clearly specify which identifiers are reserved when a standard header is included. These new words render the given examples invalid.

Item 15(b) (§4.1.2, Doc. No. X3J11/88-111 Letter P01 p. 5)

Summary of Issue: `NULL` may be defined multiple times.

Committee Response:
Changes have been made along the lines you suggested.

We think your point was that the specification could lead one to believe that

```
#include <stddef.h>
... use of NULL (but nothing else from <stdio.h>) ...
#include <stdio.h>
... use of stuff from <stdio.h>
```

is not conforming because `NULL` is defined in `<stdio.h>` and is referenced before `<stdio.h>` is first included and because `<stdio.h>` is included at least once, so that the conforming version of this example would be

```
#include <stddef.h>
#include <stdio.h>
... use of everything ...
```

This wasn't really our intention, so the wording has been improved.

Item 15(c) (§4.1.2, Doc. No. X3J11/88-111 Letter P01 p. 5)

Summary of Issue: Previous definition of *part* of a type in a standard header can also cause problems.

Committee Response:
The Committee chose a different approach to deal with this issue.

Words have been added to §4.1.2 of the Standard that more clearly specify which identifiers are reserved when a standard header is included. These new words render the given examples invalid.

Item 15(d) (§4.1.2, Doc. No. X3J11/88-111 Letter P01 p. 6)

Summary of Issue: Clarify "types defined by a header".

Committee Response:
No change to the existing wording was considered necessary.

Use of `#define` to "define a type" in a standard header would not produce a conforming implementation. `#define` defines only a *macro*, not a *type*. The Standard thus already requires that `typedef` be used to define `size_t`.

Item 15(e) (§4.1.2, Doc. No. X3J11/88-111 Letter P01 p. 6)

Summary of Issue: May extra structure members pollute the name space?

Committee Response:
This was accepted as an editorial change to the Rationale.

In standard headers such as `<time.h>` and `<locale.h>`, the form of extra member names is not specified; hence, an implementation may add only member names that begin with an underscore. This information has been added to the Rationale.

Item 15(f) (§4.1.2, Doc. No. X3J11/88-111 Letter P01 p. 6)

Summary of Issue: Behavior of multiple header inclusion may be undefined under some circumstances.

Committee Response:
The Committee believes this is clear enough as is.

First, there is no guarantee that `getchar` is defined as a macro. Furthermore, a second inclusion of a standard header (other than `assert.h`) is required to have no effect.

Item 16 (§4.1.3, Doc. No. X3J11/88-111 Letter P01 p. 6)

Summary of Issue: May an implementation save and restore `errno`?

Committee Response:
This was considered a request for information, not an issue.

The intent was that routines could very well save and restore `errno` which may well act as if it were set to zero. However, you say that signal handling makes it possible to see the value of `errno` while it has a nonzero value. As signals come in two categories, synchronous and asynchronous, let's consider them separately. With asynchronous signals (which of course could fall into one of these in-between states), you are only allowed to refer to static storage duration objects with type `volatile sig_atomic_t`. As `errno` does not match this, you cannot validly examine `errno`'s value within an asynchronous signal handler. Within a synchronous handler, the signal must have been generated by the library code. This can happen in only a few places (such as in `abort`). Any other signal generation most likely occurs due to an inappropriate value passed to the library routine, and the behavior in such cases is already undefined. Thus, the Committee believes that the "as if" rule does cover the save/restore case.

Item 17 (§4.2, Doc. No. X3J11/88-111 Letter P01 p. 7)

Summary of Issue: Change the specification for NDEBUG behavior of `assert`.

Committee Response:
The Committee has voted against this idea.

The Committee has decided that the `assert` macro should be completely silent when NDEBUG is true. If a variable declared only when NDEBUG is not defined were used within an `assert` macro, your proposal would result in a diagnostic.

Item 18 (§4.4.1.1, Doc. No. X3J11/88-111 Letter P01 p. 7)

Summary of Issue: What results from `setlocale(LC_ALL,0)` when locale is mixed?

Committee Response:
Changes have been made along the lines you suggested.

Item 19 (§4.6.2.1, Doc. No. X3J11/88-111 Letter P01 p. 7)

Summary of Issue: Make only *local* automatic variables vulnerable on `longjmp`.

Committee Response:
Changes have been made along the lines you suggested.

The paragraph beginning on page 116, line 11 of the May 1988 draft has been changed.

Item 20(a) (§4.9.5.2, Doc. No. X3J11/88-111 Letter P01 p. 7)

Summary of Issue: Allow `fflush` when nothing has been output.

Committee Response:
This was accepted as an editorial change to the Standard.

The suggested change has been incorporated in §4.9.5.2.

Item 20(b) (§4.9.5.2, Doc. No. X3J11/88-111 Letter P01 p. 7)

Summary of Issue: `fflush(NULL)` should synchronize all open files.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

This cannot be reasonably implemented in some environments.

Item 20(c) (§4.9.5.1, Doc. No. X3J11/88-111 Letter P01 p. 8)

Summary of Issue: `fclose(NULL)` would be useful.

Committee Response:
This was considered to be an invention of limited utility.

This is not as useful as `fflush(NULL)`, which is necessary to perform an essential function before a UNIX process fork.

Note that closing `stdin` and `stdout` may be inadvisable in some environments, so the proposed feature would have to be used with more care than is at first evident. (It might remove the last access for the controlling terminal, for example.) An implementor is allowed to add this feature as an extension if it is felt to be useful in a particular environment. The Committee did not wish to require it for all implementations.

Item 21 (§4.9.5.3, Doc. No. X3J11/88-111 Letter P01 p. 8)

Summary of Issue: Does append-mode write update the file position as used for reading?

Committee Response:
This was considered a request for information, not an issue.

The Standard does not define the effect of appending on the file position, and this is intentional. There is conflicting existing practice in this area.

Item 22 (§4.11.2.3, Doc. No. X3J11/88-111 Letter P01 p. 8)

Summary of Issue: Object "overlap" is not defined.

Committee Response:
A specific proposal is needed before action can be taken.

Clearly, overlap is considered to occur when at least one referenced storage location is the same, but we preferred not to try to improve the wording (which was deemed "good enough") and thereby risk breaking something.

Item 23(a) (§R4.2.1.1, Doc. No. X3J11/88-111 Letter P01 p. 8)

Summary of Issue: `assert` example is missing the right-hand side.

Committee Response:
This was accepted as an editorial change to the Rationale.

Item 23(b) (§R4.2.1.1, Doc. No. X3J11/88-111 Letter P01 p. 8)

Summary of Issue: `assert` example fails to diagnose constraint violation of non-`int` argument.

Committee Response:
This proposed editorial change was discussed but not accepted.

This is not required by the Standard.

X3J11 Response to Letter P02

Correspondent's Name and Address:

N.M. Maclaren
University of Cambridge
Computer Laboratory
New Museums Site
Corn Exchange Street
Cambridge CB2 3QG
UK

Item 1 (§3.3.3.4, Doc. No. X3J11/88-112 Letter P02 p. 1)

Summary of Issue: Require `sizeof(void)` to be 1.

Committee Response:
The Committee has voted against this idea.

The Committee has many times reaffirmed its decision to introduce the generic pointer type `void *`. Unlike a character pointer, a generic pointer cannot be dereferenced or used in pointer arithmetic. The language in §3.1.2.5 that requires a generic pointer and a character pointer to have the same representation and alignment has been clarified in the Standard; its intent is to assure that historically common usages continue to work as expected. It remains the explicit intent of the Committee, reflected in the Standard, to forbid pointer arithmetic on generic pointers. The Committee is not convinced that the specific change requested would not require changes to other parts of the Standard.

See also our responses to letter P03, items 3.1.2.5(*).

Item 2 (§4.9.8, Doc. No. X3J11/88-112 Letter P02 p. 1)

Summary of Issue: The effect of using format effectors on a text stream should be implementation defined.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

§4.9.2 lists the effects of some format effectors (special characters whose values result from escape sequences). The effects of those not listed are undefined. The Committee has discussed and rejected defining the effects of other format effectors, and since the exact behavior may be rather complex, we did not wish to force the implementation to do so either.

Item 3 (§4.9.2, Doc. No. X3J11/88-112 Letter P02 p. 2)

Summary of Issue: Does writing a new-line byte to a binary line-buffered stream flush the buffer?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The answer is "yes" if the implementation supports line buffering. However, the Standard does not require such support. See §4.9.3 of the Rationale.

Item 4 (§2.1.1.2, Doc. No. X3J11/88-112 Letter P02 p. 2)

Summary of Issue: Clarify effect of vertical format effectors on new-line.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The example provided was unclear. The Committee believes that it was intended to read

```
#if 1 \n #endif \f void fred(void) {;} \n
```

If an implementation elects to recognize the form-feed character as an end-of-line indicator (and thus translate it into the new-line character) at translation phase 1 (see §2.1.1.2), the example as modified above is valid. Otherwise, the example is explicitly invalid. The syntax of §3.8 mandates that `#endif` directives must be followed by *new-line*; the form-feed character in the modified example does not constitute a new-line except in the case noted above. The Committee believes that the Standard is clear on this issue.

Item 5 (§4.7.1.1, Doc. No. X3J11/88-112 Letter P02 p. 2)

Summary of Issue: Some systems have botched implementations of signals.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

Your examples (coupling between signals, signals being lost, and failed resumption after handlers for certain signals) seem to be already permitted by the very weak specification we give for signal behavior. Thank you for calling this to our attention. We don't like such signal behavior, and we certainly do not wish to actively *encourage* it by including your proposal in the Standard.

The Committee, in attempting to establish a standard for portable use of `signal`, recognizes that some existing implementations may find it advisable to change in order to better support the Committee's intent.

To avoid one possible misunderstanding, we should note that it was not the Committee's intention that an implementor interpret the optional "blocking" clause as license to avoid providing useful conformance with the rest of the specification. Although the Committee did not wish to specify detailed constraints for such a signal blocking facility (not even the requirement that the signal be unblocked upon normal return from the handler), the intention was to permit "held" or "masked" signals (commonly known as "reliable signals") along the lines of IEEE 1003.1-1988.

X3J11 Response to Letter P03

Correspondent's Name and Address:

Cornelia Boldyreff
C Panel Convenor
Department of Computer Science
Brunel University
Uxbridge, UB8 3PH
UK

Item 1.7 (§1.7, Doc. No. X3J11/88-113 Letter P03 p. 1)

Summary of Issue: Diagnose all non-strictly conforming features of a program.

Committee Response:
The Committee discussed this proposal but decided against it.

This requirement is more strict than what the Committee feels can be reasonably expected from a conforming implementation. For example, a program's assumption that the execution environment uses twos-complement arithmetic could not be detected by the typical compiler.

Item 3.1.2.5(a) (§3.1.2.5, Doc. No. X3J11/88-113 Letter P03 p. 1)

Summary of Issue: Remove wording requiring `char *` and `void *` be of the same representation.

Committee Response:
The Committee has voted against this idea.

The intent of the Committee is for these pointers to have the same behavior while retaining distinct types. For example, this allows functions such as `strcmp` to behave appropriately when passed to `bsearch`. Also, the return values and arguments of functions like `malloc` and `memcpy` are guaranteed to work, even for old code.

Item 3.1.2.5(b) (§3.3.3.4, Doc. No. X3J11/88-113 Letter P03 p. 1)

Summary of Issue: Require `sizeof(void)` to be 1.

Committee Response:
The Committee has voted against this idea.

The Committee discussed this, but the feeling was strong that `void` should remain an incomplete type, distinct from `char`, but whose pointers behave the same in some circumstances. Furthermore, as a pointer to `void` is guaranteed to be converted to a pointer to character type without change, a function can always convert from the `void *` to some character pointer in order to access values or to apply pointer arithmetic.

See also our response to letter P02 item 1.

Item 3.3.5 (§3.3.5, Doc. No. X3J11/88-113 Letter P03 p. 2)

Summary of Issue: Fix wording for the result of division involving a negative operand.

Committee Response:
Changes have been made along the lines you suggested.

Item 3.2.1.1 (§3.2.1.1, Doc. No. X3J11/88-113 Letter P03 p. 2)

Summary of Issue: "Wherever an `int` or `unsigned int` may be used" is too comprehensive.

Committee Response:
Changes have been made along the lines you suggested.

The integral promotions of §3.2.1.1 occur only in those places where they are explicitly mentioned, and thus they do not occur in `sizeof` context. The Committee reviewed this definition, and reaffirmed it. However, a footnote has been added to §3.2.1.1 to clarify that the integral promotions are done only where explicitly specified.

Item 3.5.4.3 (§3.5.4.3, Doc. No. X3J11/88-113 Letter P03 p. 2)

Summary of Issue: Add examples of function prototype used in conjunction with `typedef` and structures.

Committee Response:
Changes have been made along the lines you suggested.

Item 3.5.6 (§3.5.6, Doc. No. X3J11/88-113 Letter P03 p. 3)

Summary of Issue: Multiple use of name for bit fields is erroneous.

Committee Response:
This is a misinterpretation of correct wording in the document.

The example is correct. The identifier `t` is a defined type from the foregoing `typedef`. This particular bit field is unnamed. We have added further explanation after the example.

Item 3.5.7(a) (§3.5.7, Doc. No. X3J11/88-113 Letter P03 p. 3)

Summary of Issue: Add cross-reference to footnote.

Committee Response:
The Committee believes this is clear enough as is.

The footnote is present to indicate that the Committee's intention is to allow automatic arrays and structures to be initialized, even though K&R said that it was an error. There is no forward reference needed to clarify this.

Item 3.5.7(b) (§3.5.7, Doc. No. X3J11/88-113 Letter P03 p. 3)

Summary of Issue: When won't automatic variables be initialized?

Committee Response:
This is a misinterpretation of correct wording in the document.

The information requested can be found in §3.1.2.4.

Item 3.5.7(c) (§3.5.7, Doc. No. X3J11/88-113 Letter P03 p. 3)

Summary of Issue: Clarify recursion in structure declaration.

Committee Response:
This is a misinterpretation of correct wording in the document.

The constraints allow only constant expressions in initializing aggregates with initializer lists.

Item 3.6.2 (§3.6.2, Doc. No. X3J11/88-113 Letter P03 p. 4)

Summary of Issue: When do objects come into scope, and can initializers use objects known only at run time?

Committee Response:
The Committee believes this is clear enough as is.

There is no problem concerning scope in your example; the identifier `a` is in scope when the `=` is encountered. However, there is a problem with `&a`; since `a` is of automatic duration, `&a` is not an address constant and is therefore invalid as an initializer. §3.1.2.1 discusses scope of identifiers. See also §3.4 (page 53, line 31 of the May 1988 draft) and §3.5.7 (page 69, line 14).

Item 3.8.3 (§3.8.3, Doc. No. X3J11/88-113 Letter P03 p. 4)

Summary of Issue: State that object-like macros and function-like macros share the same name space.

Committee Response:
This was accepted as an editorial change.

Wording has been added to reflect this.

Item 3.8.3.4 (§3.8.3.4, Doc. No. X3J11/88-113 Letter P03 p. 4)

Summary of Issue: Clarify when function-like macros are not subject to further expansion.

Committee Response:
The Committee believes this is clear enough as is.

If `c` is invoked outside a macro, it isn't rescanned, so the result is `a ()`. If invoked inside a macro (as an argument to a macro), it is rescanned, and in this case the answer is `x`.

X3J11 Response to Letter P04

Correspondent's Name and Address:

Larry Breed
IBM Corporation
1501 California Avenue
P.O. Box 10500
Palo Alto, CA 94303

Item 1.6(a) (§1.6, Doc. No. X3J11/88-114 Letter P04 p. 1)

Summary of Issue: Define "indeterminate".

Committee Response:
This proposed editorial change was discussed but not accepted.

"Indeterminate" has its common English meaning. The meaning of "indeterminate value" is clear by the context of its use. Also, in §1.6, the definition of "undefined behavior" prohibits the use of an indeterminate value in a strictly conforming program.

Item 1.6(b) (§3.1.2.2, Doc. No. X3J11/88-114 Letter P04 p. 1)

Summary of Issue: "Object" is sometimes used where "identifier" is needed.

Committee Response:
Changes have been made along the lines you suggested.

Item 2.2.1.1 (§2.2.1.1, Doc. No. X3J11/88-114 Letter P04 p. 2)

Summary of Issue: Reword trigraph effect limit statement.

Committee Response:
The Committee believes this is clear enough as is.

Item 3.1.2.1 (§3.1.2.1, Doc. No. X3J11/88-114 Letter P04 p. 2)

Summary of Issue: "Same scope" is not defined.

Committee Response:
Changes have been made along the lines you suggested.

Not all scopes end at a } ; some end at the end of a translation unit.

Item 3.1.2.4 (§3.1.2.4, Doc. No. X3J11/88-114 Letter P04 p. 2)

Summary of Issue: Clarify that "label" includes those in a switch body.

Committee Response:
Changes have been made along the lines you suggested.

Item 3.1.2.6 (§3.1.2.6, Doc. No. X3J11/88-114 Letter P04 p. 2)

Summary of Issue: Change "has the following additions" to "satisfies the following conditions".

Committee Response:
This was accepted as an editorial change to the Standard.

Item 3.2 (§3.3.4, Doc. No. X3J11/88-114 Letter P04 p. 2)

Summary of Issue: What are "explicit" and "implicit" conversions?

Committee Response:
Changes have been made along the lines you suggested.

Item 3.3.4 (§3.3.4, Doc. No. X3J11/88-114 Letter P04 p. 3)

Summary of Issue: Qualifier in a cast has no effect.

Committee Response:
Changes have been made along the lines you suggested.

Item 3.4 (§3.4, Doc. No. X3J11/88-114 Letter P04 p. 3)

Summary of Issue: Require uniformity for expressions with integral operands.

Committee Response:
The Committee chose a different approach to deal with this issue.

We restricted the "precision and range at least as great" specification to apply to just *floating* expressions, which was our original intent.

Item 3.6.4.2 (§3.6.4.2, Doc. No. X3J11/88-114 Letter P04 p. 3)

Summary of Issue: Is a `long case` value truncated to `int`?

Committee Response:
This was considered a request for information, not an issue.

As specified in the **Semantics**, `case` label values are always converted to the (integral-promoted) type of the controlling expression. Thus, if the controlling expression for your example had type `int`, then the `long case` value would be truncated to `int`.

Item 4.9.6.8 (§4.9.6.8, Doc. No. X3J11/88-114 Letter P04 p. 3)

Summary of Issue: How can the `arg` argument to `vprintf` be indeterminate on return?

Committee Response:
This was considered a request for information, not an issue.

Since `va_list` could be either an array or nonarray type, it may or may not be passed by value. A portable program must not assume either implementation choice.

Item 4.10.6.2 (§4.10.6.2, Doc. No. X3J11/88-114 Letter P04 p. 3)

Summary of Issue: Reword statement about the sign of an inexact quotient.

Committee Response:
Changes have been made along the lines you suggested.

X3J11 Response to Letter P05

Correspondent's Name and Address:

Robert Paul Corbett
ELXSI
2334 Lundy Place
San Jose, CA 95131

Item 1 (§1.6, Doc. No. X3J11/88-115 Letter P05 p. 1)

Summary of Issue: Improve definition of "byte".

Committee Response:
Changes have been made along the lines you suggested.

Item 2 (§2.2.4.2, Doc. No. X3J11/88-115 Letter P05 p. 1)

Summary of Issue: Floating-point model footnote is wrong.

Committee Response:
This was accepted as an editorial change to the Standard.

Footnote 8 has been deleted. You are correct. Thank you for your comment.

Item 3 (§2.2.4.2, Doc. No. X3J11/88-115 Letter P05 p. 1)

Summary of Issue: Text and formula for `*_EPSILON` disagree.

Committee Response:
Changes have been made along the lines you suggested.

We agree that this was misleading. We have changed the description of `*_EPSILON` to correspond to the formula b^{1-p}.

Item 4 (§2.2.4.2, Doc. No. X3J11/88-115 Letter P05 p. 2)

Summary of Issue: Example `DBL_MIN` is inaccurate.

Committee Response:
This was accepted as an editorial change to the Standard.

X3J11 Response to Letter P06

Correspondent's Name and Address:

Alan Beale
SAS Institute Inc.
SAS Circle
Box 8000
Cary, NC 27512-8000

Item 1 (§3.1.8, Doc. No. X3J11/88-116 Letter P06 p. 1)

Summary of Issue: Preprocessing number definition causes surprises.

Committee Response:
The Standard reflects the result of previous discussion of this issue.

At the September 1988 meeting the Committee discussed two proposals, but decided to leave the definition the way it is. Trying to refine the notion of preprocessing number tokens at this stage is likely to introduce as many surprises as are fixed. The few cases of existing code that would be broken were not considered to present a significant practical problem.

Some Committee members insisted that they *wanted* preprocessor numbers to be "greedier" than one might think appropriate, so that (for example) binary constants such as `0b1001` could be supported as an extension, and that any change other than fixing *just* the "broken code" case would not be acceptable.

Item 2 (§4.9.1, Doc. No. X3J11/88-116 Letter P06 p. 3)

Summary of Issue: Fix `FILENAME_MAX` definition.

Committee Response:
Changes have been made along the lines you suggested.

`FILENAME_MAX` is now further described in a footnote as the implementation's recommended size for an array that is to hold a file name.

Item 3 (§3.2.2.1, Doc. No. X3J11/88-116 Letter P06 p. 6)

Summary of Issue: Allow dereferencing/indexing of non-lvalue arrays.

Committee Response:
This is too radical a change to adopt at this stage.

Changing the rules to allow conversion of a non-lvalue array to a pointer in this case was felt to be too drastic a change to the language at this point. Such a conversion would raise, for instance, the issue of the lifetime of the temporary object holding the array value.

Item 4 (§4.7.1.1, Doc. No. X3J11/88-116 Letter P06 p. 8)

Summary of Issue: Prohibition against library use of `signal` imposes an undue burden on implementors.

Committee Response:
This was accepted as an editorial change to the Standard.

The Standard now makes it clear that an application-provided signal handler may portably call `signal` only to reset the handler for its own signal.

X3J11 Response to Letter P07

Correspondent's Name and Address:

Gavin Koch
SAS Institute Inc.
SAS Circle
Box 8000
Cary, NC 27512-8000

Item 1 (§3.1.2.6, Doc. No. X3J11/88-117 Letter P07 p. 1)

Summary of Issue: Compatibility of structures should not depend on where they are defined.

Committee Response:
This proposal conflicts with too much prior art.

Removing the words "in separate translation units" would conflict with prior art. In a single translation unit, C has always treated distinct structure declarations as defining distinct structure types. Also, allowing them to be compatible with different tags would place too great a burden on the implementation to determine compatibility, something that is easy to determine as it now stands.

The rules have to be different for defining compatibility for structures in different translation units. In a single translation unit, a structure type is defined exactly once, then the structure tag is used after that to refer to that structure type. In two translation units, the structure type necessarily has to be defined twice (once for each translation unit). Thus the rule about a structure type only being defined once has to be relaxed. Since the implementation typically cannot check across translation unit boundaries whether or not the tags are identical, it did not seem important to require that they be identical.

This is now explained in the Rationale.

Item 2 (§3.3.16.1, Doc. No. X3J11/88-117 Letter P07 p. 2)

Summary of Issue: Clarify "the value ... is accessed".

Committee Response:
No change to the existing wording was considered necessary.

The text on page 37, lines 17-27 of the May 1988 draft explains what an access requires of an lvalue.

In any case, where the right hand side of the assignment is not an lvalue, the assigned value is not accessed from the object. This means that your example

```
y.c = y.x.b + 1; /* (2) */
```

is valid.

X3J11 Response to Letter P08

Correspondent's Name and Address:

Gary H. Merrill
SAS Institute Inc.
SAS Circle
Box 8000
Cary, NC 27512-8000

Item 1 (§3.1.2.1, Doc. No. X3J11/88-118 Letter P08 p. 1)

Summary of Issue: Prototype scope for tag seems unnatural.

Committee Response:
The Committee discussed this proposal but decided against it.

This problem may be solved by something of the form

```
struct TAG;
```

appearing with file scope prior to the prototype.

X3J11 Response to Letter P09

Correspondent's Name and Address:

Henry Spencer
25 Harbord St.
University of Toronto
Toronto, Ont. M5S 1A1
Canada

Item 1 (§1.7, Doc. No. X3J11/88-119 Letter P09 p. 1)

Summary of Issue: Subtle name-space constraints deserve a footnote.

Committee Response:
Changes have been made along the lines you suggested.

Item 2 (§2.1.1.1, Doc. No. X3J11/88-119 Letter P09 p. 1)

Summary of Issue: Suggested wording improvement.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 3 (§2.1.1.2, Doc. No. X3J11/88-119 Letter P09 p. 1)

Summary of Issue: Comments should participate in space condensation.

Committee Response:
Changes have been made along the lines you suggested.

Item 4 (§2.1.1.2, Doc. No. X3J11/88-119 Letter P09 p. 2)

Summary of Issue: Token conversion should be required to be one-to-one.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 5 (§2.2.1, Doc. No. X3J11/88-119 Letter P09 p. 2)

Summary of Issue: Reorder list of garbage-character containing tokens.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 6 (§3.1.2.5, Doc. No. X3J11/88-119 Letter P09 p. 2)

Summary of Issue: Top type examples are poorly worded.

Committee Response:
This proposed editorial change was discussed but not accepted.

We agree that there were problems in understanding this concept, and have made significant editorial changes to clarify this area.

Item 7 (§3.1.2.6, Doc. No. X3J11/88-119 Letter P09 p. 2)

Summary of Issue: Structure compatibility must include bit-field widths.

Committee Response:
Changes have been made along the lines you suggested.

Item 8 (§3.1.3.4, Doc. No. X3J11/88-119 Letter P09 p. 2)

Summary of Issue: Limits on character constant width are not clearly allowed.

Committee Response:
This proposed editorial change was discussed but not accepted.

The value of an integer character constant containing more than one character is implementation defined (page 29, lines 7-10 of the May 1988 draft). An implementation can choose to limit the logical source line length (if so, the limit must be at least 509). Thus, the size of a character constant can be limited if the implementation desires.

Item 9 (§3.3.2.3, Doc. No. X3J11/88-119 Letter P09 p. 3)

Summary of Issue: Common initial subsequence must consider bit fields.

Committee Response:
Changes have been made along the lines you suggested.

Item 10 (§3.5.2.1, Doc. No. X3J11/88-119 Letter P09 p. 3)

Summary of Issue: Semantics of a leading zero-width bit field are unspecified.

Committee Response:
The Committee discussed this proposal but decided against it.

The Committee believes that the statement is clear enough as written; leading zero-width unnamed bit fields have no effect.

Item 11 (§3.5.2.3, Doc. No. X3J11/88-119 Letter P09 p. 3)

Summary of Issue: What is the meaning of "size not needed"?

Committee Response:
This proposed editorial change was discussed but not accepted.

This is merely a restatement of the definition of "incomplete type", defined in §3.1.2.5.

The use of a *complete* type as opposed to an incomplete type is required by various prohibitions (*e.g.* see **Constraints** for §3.3.2.2, function calls).

Item 12 (§3.7.1, Doc. No. X3J11/88-119 Letter P09 p. 3)

Summary of Issue: Declaration of tags in a declaration list should be permitted.

Committee Response:
Changes have been made along the lines you suggested.

Item 13 (§3.8, Doc. No. X3J11/88-119 Letter P09 p. 3)

Summary of Issue: Horizontal white-space rules are contradictory.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 14 (§3.8.1, Doc. No. X3J11/88-119 Letter P09 p. 3)

Summary of Issue: Keywords can appear in preprocessor expressions.

Committee Response:
Changes have been made along the lines you suggested.

This point is already covered by footnote 74 on page 84 of the May 1988 draft. The wording was changed to clarify the intent of the Committee.

Item 15 (§4.1.2, Doc. No. X3J11/88-119 Letter P09 p. 4)

Summary of Issue: Other reserved identifiers are not forbidden.

Committee Response:
Changes have been made along the lines you suggested.

X3J11 Response to Letter P10

Correspondent's Name and Address:

David G. Hough
PO Box 20370
San Jose, CA 95160

Item 1(a) (§2.2.4.2, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Raise `LDBL_MAX`.

Committee Response:
The Committee discussed this proposal but decided against it.

Many implementations cannot meet this requirement. This is a quality of implementation issue. The Committee also voted against requiring this as a **FUTURE LANGUAGE DIRECTION**.

[NOTE — Although items 1(*) were proposed for inclusion as **FUTURE DIRECTIONS**, they were judged on the basis of current support in the Committee for the proposals. This is proper, because **FUTURE DIRECTIONS** represent the current consensus desires of the Committee for things that cannot feasibly be mandated immediately. Future revisions of the C Standard are not actually constrained by the current Standard's **FUTURE DIRECTIONS**, although implementors and programmers are advised to take heed of these intentions.]

Item 1(b) (§3.9, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Need non-aliased array parameters.

Committee Response:
Changes have been made along the lines you suggested.

Words were added to **FUTURE LANGUAGE DIRECTIONS** to allow for declaration of non-aliased array-of-type parameters.

Item 1(c) (§3.4, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Don't evaluate constant expressions with side effects at compile time.

Committee Response:
No change to the existing wording was considered necessary.

It was not entirely clear what you meant; here are the three possibilities that occurred to us:

1. If you meant signals as described on page 37 of the May 1988 draft, then the behavior is undefined *if* the code would actually be reached.
2. If you meant out-of-band effects other than as described on page 37, then we specify nothing; that is outside the scope of the Standard.
3. If you meant calculations where rounding mode determines the answer, then at translation time the implementation *must* compute the result to *at least* as much precision as would occur at run time.

Item 1(d) (§3.4, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Need method to force compile time evaluation.

Committee Response:
Quality of implementation is beyond the scope of the Standard.

A conforming implementation must abide by the requirements of the Standard. Any additional flavors of implementation are up to the discretion of the implementor to provide.

Item 1(e) (§3.5.7, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Need syntax for initializer counts.

Committee Response:
Extensions are allowed in this regard, but they are not required.

This is a good idea, but we needed a concrete proposal. We have considered proposals for this a number of times, but none have seemed worthy of blessing as a **FUTURE LANGUAGE DIRECTION**.

Item 1(f) (§4.9.6.1, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Display sign of zero.

Committee Response:
Quality of implementation is beyond the scope of the Standard.

Item 1(g) (§4.9.6.1, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Display unique format for true zero.

Committee Response:
This proposal conflicts with too much prior art.

The Committee has rejected this in the past for inclusion in the Standard proper, and it was felt that there would be insufficient support for this as a **FUTURE LIBRARY DIRECTION**.

Item 1(h) (§4.10.1.4, Doc. No. X3J11/88-120 Letter P10 p. 3)

Summary of Issue: Add standard method for printing and reading NaNs and Infinities.

Committee Response:
Extensions are allowed in this regard, but they are not required.

Since there are many architectures that don't support NaNs and Infinities, there was insufficient support for this even as a **FUTURE LIBRARY DIRECTION**.

Item 2(a) (§3.2.1.5, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Discourage implicit type narrowing.

Committee Response:
This proposal conflicts with too much prior art.

This is already listed as a **Common warning**. An implementation is at liberty to issue warnings.

Item 2(b) (§3.5.6, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Discourage implicit initializer.

Committee Response:
This proposal conflicts with too much prior art.

There is too much existing code that would break with this requirement, and the Standard is not the place to expound on hygienic programming practices. An implementation is at liberty to issue warnings.

Item 2(c) (§4.5, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Deprecate `errno`.

Committee Response:
The Committee discussed this proposal but decided against it.

The Committee voted against making `errno` behavior implementation-defined in the math functions.

Item 2(d) (§4.7, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Require sane use of `SIGFPE`.

Committee Response:
The Standard must accommodate a variety of environments.

There are too many existing implementations that treat division by integer zero as `SIGFPE`. Words have been added to the Rationale making it clear that `SIGFPE` does not imply floating-point expressions only. These names are historical.

Item 2(e) (§4.10.1, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Use of `atof` and `atol` is hazardous.

Committee Response:
The Committee discussed this proposal but decided against it.

These routines have been kept since they can be implemented to run faster by not doing the error checking. Also, these routines are used by a vast amount of existing code.

Item 3(a) (§2.2.4.2, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Footnote 8 is surprising.

Committee Response:
The Committee chose a different approach to deal with this issue.

Footnote 8 has been deleted (it was wrong). Thank you for your comment.

Item 3(b) (§3.1.3.1, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Decimal-to-floating bounds are surprisingly restrictive.

Committee Response:
The Committee chose a different approach to deal with this issue.

The Standard inadvertently imposed an impractically close bound on the error in decimal-to-float conversion. We have modified it to permit the result to be the nearest representable value or either adjacent representable value.

Item 3(c) (§3.2.1.4, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Emphasize requirement to convert `double` to `float` correctly.

Committee Response:
The Committee believes this is clear enough as is.

We think the present statement of the requirement is sufficiently clear.

Item 3(d) (§3.3, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Emphasize requirement to honor parentheses.

Committee Response:
The Committee believes this is clear enough as is.

As you note, the Rationale discusses this clearly; also, it has received a lot of "press". It's hard to imagine this not being noticed by any member of the target audience for the Standard.

Also see §2.1.2.3.

Item 3(e) (§3.3.4, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Emphasize requirement to perform conversion on floating-point casts.

Committee Response:
The Committee believes this is clear enough as is.

The Standard and Rationale both discuss this requirement, which is not new. (What is new is that the resulting `float` no longer need be immediately promoted back to `double` in an expression. That too has received a lot of "press".)

Item 3(f) (§4.5.1, Doc. No. X3J11/88-120 Letter P10 p. 4)

Summary of Issue: Several math functions are now overspecified.

Committee Response:
Changes have been made along the lines you suggested.

We have removed the requirement for `tan` to return a correctly-signed `HUGE_VAL`.

Otherwise we believe the specifications are loose enough to not require guard bits for efficient implementation.

Item 3(g) (§4.5.4.3, Doc. No. X3J11/88-120 Letter P10 p. 5)

Summary of Issue: `ldexp` and `frexp` are biased against non-binary floating point implementations.

Committee Response:
This was accepted as an editorial change to the Rationale.

These functions essentially map the actual floating point to and from a simpler binary model. Information may indeed be lost in the translation. We have added words to this effect to the Rationale.

Item 3(h) (§3.5.4.3, Doc. No. X3J11/88-120 Letter P10 p. 5)

Summary of Issue: `printf` needs a prototype in scope.

Committee Response:
This was considered a comment rather than an issue.

This is a requirement on maximally conforming programs, not on conforming implementations. We doubt that existing implementations are likely to change their `printf` interface.

Item 3(i) (§4.5.1, Doc. No. X3J11/88-120 Letter P10 p. 5)

Summary of Issue: Who can set `errno`?

Committee Response:
This is a misinterpretation of correct wording in the document.

`errno` is required to be set by the §4.5 math functions and the §4.10 `ato*` and `strto*` functions, but may be set by others. The Standard forbids any library function's resetting `errno` to 0.

Item 3(j) (§4.9.6.2, Doc. No. X3J11/88-120 Letter P10 p. 5)

Summary of Issue: `scanf` mismatch *vs.* pushback.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

`scanf` is not required to push back more than one character; the specification does not require a more powerful pushback mechanism. The "`-`." in your example are indeed lost.

These functions reflect existing practice. Some Committee members prefer to confine their use to `sscanf` on the result of `fgets`, and to implement their own look-ahead mechanism rather than rely on the quite limited `ungetc` mechanism.

Item 3(k) (§4.10.1.6, Doc. No. X3J11/88-120 Letter P10 p. 5)

Summary of Issue: The name `strtoul` exceeds six characters.

Committee Response:
This was considered a comment rather than an issue.

There are other external identifiers in the standard library like this. However, all such names are unique in the first six characters.

Item 4 (§1.1, Doc. No. X3J11/88-120 Letter P10 p. 5)

Summary of Issue: Anticipate supplemental standards.

Committee Response:
This was considered a comment rather than an issue.

We have changed the Standard to clarify that `<float.h>` is a model with which implementations are not required to conform. We believe that the Standard does not prevent extensions along the lines you describe.

Item 5 (§2.2.4.2, Doc. No. X3J11/88-120 Letter P10 p. 6)

Summary of Issue: Use "significand".

Committee Response:
Changes have been made along the lines you suggested.

"Mantissa" has been changed to "significand" except in the `*_MANT_DIG` macro names.

Item 6 (§2.2.4.2, Doc. No. X3J11/88-120 Letter P10 p. 7)

Summary of Issue: Postpone `<float.h>`.

Committee Response:
This is too radical a change to adopt at this stage.

Eliminating `<float.h>` was considered too radical a change. The Standard now states that the floating-point model is illustrative only, and we reworded the definitions of `*_EPSILON` and `*_DIG` along the lines of your suggestions.

Item 7 (§R3.2.1.4, Doc. No. X3J11/88-120 Letter P10 p. 8)

Summary of Issue: Round conversions between floating-point types.

Committee Response:
The Committee chose a different approach to deal with this issue.

The Rationale sentence was kept but reworded.

Item 8 (§3.5.4.2, Doc. No. X3J11/88-120 Letter P10 p. 9)

Summary of Issue: Allow array parameters to have variable dimensions.

Committee Response:
The Committee discussed this proposal but decided against it.

The Committee voted against moving the constraint into the **Semantics** section (to allow extensions in this regard). This was considered too radical a change.

Item 9 (§R3.7.1, Doc. No. X3J11/88-120 Letter P10 p. 10)

Summary of Issue: Remove incorrect Rationale comment.

Committee Response:
No change to the existing wording was considered necessary.

Your premise (that a signed-magnitude representation is required) was mistaken; the Standard has been made clearer about this.

Item 10 (§4.5.1, Doc. No. X3J11/88-120 Letter P10 p. 11)

Summary of Issue: Make `errno` implementation defined.

Committee Response:
The Committee discussed this proposal but decided against it.

The Committee has voted one more time to leave `errno` as is. There is sentiment that `errno` is an undue burden, but unfortunately it is required. Also see our response to item 2(c).

Item 11(a) (§R4.5.6.4, Doc. No. X3J11/88-120 Letter P10 p. 12)

Summary of Issue: `fmod` discussion is incomplete.

Committee Response:
This was accepted as an editorial change to the Rationale.

Your words have been added to the Rationale.

Item 11(b) (§R4.5.2, Doc. No. X3J11/88-120 Letter P10 p. 13)

Summary of Issue: Trig function argument reduction needs a warning.

Committee Response:
Changes have been made along the lines you suggested.

Item 11(c) (§R4.5.6.1, Doc. No. X3J11/88-120 Letter P10 p. 13)

Summary of Issue: `ceil` rationale needs words.

Committee Response:
This was accepted as an editorial change to the Rationale.

Your words have been added to the Rationale.

Item 12 (§4.5, Doc. No. X3J11/88-120 Letter P10 p. 13)

Summary of Issue: Standardize `hypot`.

Committee Response:
The Committee discussed this proposal but decided against it.

We recognize that `hypot` has value, but are reluctant to add a function that was not in the base document, unless it addresses a critical need.

Item 13 (§4.5.4.6, Doc. No. X3J11/88-120 Letter P10 p. 14)

Summary of Issue: Delete `modf`.

Committee Response:
The Committee discussed this proposal but decided against it.

This function exists in the base document. Because of its history, the Committee has voted to retain it.

Item 14 (§4.7, Doc. No. X3J11/88-120 Letter P10 p. 14)

Summary of Issue: Specify which signals can arise.

Committee Response:
The Committee discussed this proposal but decided against it.

Macros, defined at translation time, cannot be used to describe system behavior that can change at execution time. The definition of `signal` is loose because of the number of implementation-defined characteristics associated with signals.

X3J11 Response to Letter P11

Correspondent's Name and Address:

Shane P. McCarron
NAPS International
117 Mackubin St.
Suite 6
St. Paul, MN 55102

Item 1 (§1, Doc. No. X3J11/88-121 Letter P11 p. 2)

Summary of Issue: Refer to IEEE 1003.1-1988.

Committee Response:
This proposed editorial change was discussed but not accepted.

IEEE 1003.1-1988 requires conformance to ANSI C, but ANSI C does not depend on IEEE 1003.1-1988.

ANSI C is intended to apply to a variety of architectures and operating systems, including many that are not covered by IEEE 1003.1-1988.

The Rationale explains the relationship between the two standards.

Item 2 (§4, Doc. No. X3J11/88-121 Letter P11 p. 2)

Summary of Issue: Name space pollution is existing practice.

Committee Response:
The Committee has reaffirmed this decision on more than one occasion.

Far from restricting applications, this requirement is necessary for portable applications to exist. Implementations may provide additional names under control of compiler switches or better still underscore-macros, as described in the Rationale §4.1.2. In "conforming" mode these extensions must be hidden.

Item 3 (§2.1.1.2, Doc. No. X3J11/88-121 Letter P11 p. 2)

Summary of Issue: White space within tokens should not be replaced.

Committee Response:
The Committee believes this is clear enough as is.

This clearly refers to white-space characters that are not part of a preprocessing token.

Item 4 (§2.1.2.2, Doc. No. X3J11/88-121 Letter P11 p. 2)

Summary of Issue: To what does the business about argument "case" refer?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

"Execution character set" is the correct interpretation.

Item 5 (§2.1.2.2, Doc. No. X3J11/88-121 Letter P11 p. 2)

Summary of Issue: There is a problem with `environ`.

Committee Response:
The Committee discussed this proposal but decided against it.

The ANSI C and IEEE 1003.1-1988 standards are not necessarily in conflict here, although it is true that in order to avoid the name-space conflict a mutually conforming implementation must

rely on some mechanism such as "global symbolic equate" or a zero-size global object `environ` in a separate library module immediately preceding the module that defines storage for `__environ` (the name used by the C run-time startup code). Implementor control over the way the linker operates, while inappropriate to require for the more universal C Standard (hence the constraint on uniqueness of external identifiers), is not unrealistic to expect for most POSIX implementations. Several implementors have indicated their intention to provide such a feature.

Another solution, of course, would be to use separate run-time startup modules for strict ANSI-conforming and vendor-extended (possibly POSIX-conforming) implementations, perhaps *via* a compiler flag. This may be useful anyway, for hiding extensions in certain standard headers.

Item 6 (§2.2.4.1, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: Clarify "32 expressions nested by parentheses".

Committee Response:
This was accepted as an editorial change to the Standard.

This has been reworded.

Item 7 (§2.2.4.1, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: Dislikes external identifier restrictions.

Committee Response:
This was considered a comment rather than an issue.

Sorry — changing this would conflict with too much prior art, and the Standard must accommodate a variety of environments. We don't like this restriction any more than you do, but we feel it is necessary given current circumstances. Implementations are free (and encouraged) to do better than the minimum requirement, although portable programs cannot count on it.

Note that case indistinguishability reduces the number of possible names by a factor of slightly more than 28, not 2 as you stated.

Item 8 (§2.2.4.1, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: IEEE 1003.2 may require C translators to accept source lines up to a length defined by `LINE_MAX`.

Committee Response:
This was considered a comment rather than an issue.

This is okay; there is no conflict.

Item 9 (§2.2.4.2, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: Powers of two would be helpful.

Committee Response:
The Committee believes this is clear enough as is.

The relation of these limits to powers of two can be computed easily. The Standard is not intended to double as a tutorial.

Item 10 (§3.1.2.1, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: File-scope identifier's scope should not extend backward.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

Refer to the last paragraph in §3.1.2.1.

We have also added words to clarify what "same scope" means.

Item 11 (§3.1.2.2, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: Antecedent is unclear; reword.

Committee Response:
Changes have been made along the lines you suggested.

Item 12 (§3.2.1.2, Doc. No. X3J11/88-121 Letter P11 p. 3)

Summary of Issue: Is magnitude of objects really supposed to be analyzed at run time?

Committee Response:
The Committee believes this is clear enough as is.

You seem to be confusing the terms "value" and "size". "Value" is used to refer to the magnitude of a number and is used in the text to specify the result of the conversion or that the result is undefined. "Size" refers to the quantity that results from the `sizeof` operator and is what is used to determine the conversion to be applied.

Item 13 (§3.3.6, Doc. No. X3J11/88-121 Letter P11 p. 4)

Summary of Issue: Clarify pointer subtraction.

Committee Response:
This is a misinterpretation of correct wording in the document.

The type of object pointed to is that derived from the composite type of the two pointers. The constraints for the binary - operator disallow pointers to incomplete types (such as `void *`) since there is no known size for the required pointer arithmetic. If in your example the pointers had been of type `char *`, then the correct result would indeed have been 6.

Item 14 (§3.5.6, Doc. No. X3J11/88-121 Letter P11 p. 4)

Summary of Issue: "Inner scope" is not defined.

Committee Response:
The Committee believes this is clear enough as is.

"Inner" has its common English meaning.

Item 15 (§3.5.7, Doc. No. X3J11/88-121 Letter P11 p. 4)

Summary of Issue: Specify *which* base document.

Committee Response:
The Committee believes this is clear enough as is.

§1.5 clearly indicates that one base document applies to §3 and the other applies to §4.

Item 16(a) (§3.8.2, Doc. No. X3J11/88-121 Letter P11 p. 4)

Summary of Issue: Was embedded > really intended to be valid in header names?

Committee Response:
This was considered a request for information, not an issue.

The answer is "yes, for the implementation-defined search only". The subsequent h-char-sequence is *not* required to be considered valid.

Also note that the <...> form need not correspond to the name of an actual file (it might be built into the compiler).

Item 16(b) (§3.8.2, Doc. No. X3J11/88-121 Letter P11 p. 4)

Summary of Issue: \ is not allowed in a header name.

Committee Response:
This is a misinterpretation of correct wording in the document.

This does indeed prohibit a *portable* program from using a \ in a header name, because not all environments allow it. Undefined behavior is not prohibited; it's just not portable.

Item 17 (§3.9.5, Doc. No. X3J11/88-121 Letter P11 p. 4)

Summary of Issue: Obsolescent function definition is more readily documentable.

Committee Response:
This was considered a comment rather than an issue.

We do not consider the obsolescent function definitions to be more documentable.

Item 18 (§4.5.1, Doc. No. X3J11/88-121 Letter P11 p. 5)

Summary of Issue: Implementation-defined return value should be distinguishable from a non-error value.

Committee Response:
The Standard must accommodate a variety of architectures.

On some architectures, some functions may legitimately return all representable values.

Item 19 (§4.6, Doc. No. X3J11/88-121 Letter P11 p. 5)

Summary of Issue: `setjmp` is not necessarily a macro.

Committee Response:
This proposed editorial change was discussed but not accepted.

The intent is clear, and better wording would complicate the document unnecessarily.

Item 20 (§4.9.2, Doc. No. X3J11/88-121 Letter P11 p. 5)

Summary of Issue: There is no portable way to put VT and FF characters out to a text file.

Committee Response:
The Standard must accommodate a variety of environments.

Some environments do not allow this.

Item 21 (§4.9.2, Doc. No. X3J11/88-121 Letter P11 p. 5)

Summary of Issue: Removal of trailing spaces from a text line is annoying.

Committee Response:
The Standard must accommodate a variety of environments.

It's unavoidable in some environments.

Item 22 (§4.9.2, Doc. No. X3J11/88-121 Letter P11 p. 5)

Summary of Issue: Clarify that appended null characters appear at the end of the stream.

Committee Response:
This was accepted as an editorial change to the Standard.

We thought that "to append" means "to place at the end", but this has been clarified.

Item 23 (§4.9.3, Doc. No. X3J11/88-121 Letter P11 p. 5)

Summary of Issue: Comment about append mode behavior.

Committee Response:
This was considered a comment rather than an issue.

See §4.9.7.3, which states that if a stream is open for appending, written characters do appear at the end.

Your comment led us to make a related improvement; thank you.

Item 24 (§4.9.3, Doc. No. X3J11/88-121 Letter P11 p. 6)

Summary of Issue: Truncating a text file after a write precludes making multiple passes over such a file.

Committee Response:
The Standard must accommodate a variety of environments.

This is unavoidable in some environments.

Item 25 (§4.9.5.3, Doc. No. X3J11/88-121 Letter P11 p. 6)

Summary of Issue: Does append to null-padded binary stream overwrite the nulls?

Committee Response:
This issue can be resolved by a careful reading of the Standard.

Appending follows the null padding. It's unavoidable in some environments.

Item 26 (§4.9.5.3, Doc. No. X3J11/88-121 Letter P11 p. 6)

Summary of Issue: Truncation of binary update streams on open should be somehow determinable by application code.

Committee Response:
This is a misinterpretation of correct wording in the document.

We have clarified the sentence in question.

X3J11 Response to Letter P12

Correspondent's Name and Address:

Mark Brader
SoftQuad Inc.
720 Spadina Avenue
Toronto, Ontario
M5S 2T9
Canada

Item 1 (§1.6, Doc. No. X3J11/88-122 Letter P12 p. 1)

Summary of Issue: Specify that every byte value is considered to be a character.

Committee Response:
Changes have been made along the lines you suggested.

We have clarified that in both source and execution environments, a character is just whatever fits in a byte.

Item 2 (§2.1.2.2, Doc. No. X3J11/88-122 Letter P12 p. 1)

Summary of Issue: Is three-argument `main` nonconforming?

Committee Response:
This is a misinterpretation of correct wording in the document.

A portable program will only use zero or two arguments for `main`. An implementation that allows three arguments may well be conforming, because this extension is not required to be diagnosed.

Item 3(a) (§3.1.2.5, Doc. No. X3J11/88-122 Letter P12 p. 1)

Summary of Issue: Are qualified types distinct from the corresponding unqualified ones?

Committee Response:
The Committee chose a different approach to deal with this issue.

Words were added to clarify the significance of type qualifiers.

Item 3(b) (§3.1.2.5, Doc. No. X3J11/88-122 Letter P12 p. 1)

Summary of Issue: Wording of "top type" is still unclear.

Committee Response:
Changes have been made along the lines you suggested.

The wording was revised to clarify this concept.

Item 4 (§3.1.2.5, Doc. No. X3J11/88-122 Letter P12 p. 2)

Summary of Issue: `signed int` is sometimes a basic type.

Committee Response:
Changes have been made along the lines you suggested.

Your analysis is correct, and wording has been added to clarify `int` bit fields.

Item 5 (§4.1.3, Doc. No. X3J11/88-122 Letter P12 p. 3)

Summary of Issue: Allowing `errno` to be a macro breaks existing code.

Committee Response:
The Standard must accommodate a variety of environments.

Some environments require that the latitude be given. Existing code is not broken, it is merely not portable.

Item 6 (§4.4, Doc. No. X3J11/88-122 Letter P12 p. 3)

Summary of Issue: Why are nonnegative members of `struct lconv` required to be (possibly signed) `char` rather than some unsigned type?

Committee Response:
This was considered a request for information, not an issue.

The values required seemed not to require more than the positive range of a `char` variable.

Item 7 (§4.10.5.1, Doc. No. X3J11/88-122 Letter P12 p. 3)

Summary of Issue: `bsearch` invocation imposes only a partial ordering.

Committee Response:
Changes have been made along the lines you suggested.

X3J11 Response to Letter P13

Correspondent's Name and Address:

Mark Dunn
Dept. Mechanical Engineering
Sheffield University
Mappin St.
Sheffield
England

Item 1 (§3.5.2.1, Doc. No. X3J11/88-123 Letter P13 p. 1)

Summary of Issue: Structure member alignment should not be affected by subsequent members.

Committee Response:
The request is reflected in the current draft.

The Standard already guarantees the restriction you desire, in an indirect way. In §3.3.2.3, the Standard says that if a union contains several structures that share a common initial sequence, and if the union object currently contains one of these structures, it is permitted to inspect the common initial part of any of them.

X3J11 Response to Letter P14

Correspondent's Name and Address:

Prescott K. Turner, Jr.
Software Development Technologies, Inc.
375 Dutton Road
Sudbury, MA 01766-2509

Item 1 (§2.2.4.2, Doc. No. X3J11/88-124 Letter P14 p. 1)

Summary of Issue: Remove footnote about floating-point model.

Committee Response:
This was accepted as an editorial change to the Standard.

Footnote 8 has been deleted. You are correct. Thank you for your comment.

Item 2 (§2.2.4.2, Doc. No. X3J11/88-124 Letter P14 p. 1)

Summary of Issue: Example `DBL_MIN` is inaccurate.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 3 (§R3.2.2.2, Doc. No. X3J11/88-124 Letter P14 p. 1)

Summary of Issue: Change description of `void` from "has no valid states" to "requires no storage"; remove "(nonexistent)".

Committee Response:
This was accepted as an editorial change to the Rationale.

The Committee realizes that `void` can be considered to have one valid state. We have decided against removing the word "nonexistent" from the Standard as it may clarify something for the less astute reader. However, your words for the Rationale are clearly superior to the existing ones and have been adopted by the Committee.

Item 4 (§3.3.6, Doc. No. X3J11/88-124 Letter P14 p. 1)

Summary of Issue: Attempt to permit pointer to one past the last member of an array was botched.

Committee Response:
Changes have been made along the lines you suggested.

An editorial change has been made to §3.3.6, although we chose to use words along the lines of the existing ones. We believe we have captured your intent.

Item 5 (§3.3.9, Doc. No. X3J11/88-124 Letter P14 p. 2)

Summary of Issue: Specify explicit properties for equality operators.

Committee Response:
Changes have been made along the lines you suggested.

Words have been added to §3.3.9 to be more explicit about the semantics inherited by the equality operators. We felt it was safer to refer the reader to §3.3.8 than to try to get the comparison semantics right in one more section.

Item 6 (§3.8, Doc. No. X3J11/88-124 Letter P14 p. 3)

Summary of Issue: Skipped-directive syntax needs clarification.

Committee Response:
This is a misinterpretation of correct wording in the document.

The Committee agrees that there may seem to be a contradiction between the grammar and the semantic description. However, the Committee feels that the Standard is clear as it stands. We believe that the lack of full syntactic processing is of fundamental importance for portability and reflects widespread existing practice.

Item 7 (§3.8, Doc. No. X3J11/88-124 Letter P14 p. 3)

Summary of Issue: Preprocessing syntax should apply left-to-right in conjunction with source file inclusion and conditional inclusion.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

Actually, your example translation unit is invalid and must be diagnosed. §2.1.1.2 specifies that translation phases 1 through 4 need to be processed recursively for each file.

Each preprocessing file (as indicated in the syntax of §3.8; note the word "file") requires each `#if` to be matched with a corresponding `#endif` in the same preprocessing file.

The example you give should result in a diagnostic for the `#endif` in the `u.h` file.

Item 8 (§3.8.1, Doc. No. X3J11/88-124 Letter P14 p. 4)

Summary of Issue: When expanding, you cannot predict whether there will be a `)`.

Committee Response:
The Committee chose a different approach to deal with this issue.

The Standard has been changed to clarify that the example you present has undefined behavior.

Item 9 (§3.8.3.4, Doc. No. X3J11/88-124 Letter P14 p. 4)

Summary of Issue: The phrase "nested replacements" is imprecise.

Committee Response:
This issue can be resolved by a careful reading of the Standard.

The Committee felt that the section on rescanning is clear enough. Since `f` is still being processed when rescanning after the expansion `g`, the second `f` is hidden and therefore not expanded. The entire description must be read as a unit in order to understand the replacement algorithm.

The text of the `#defines` are necessary to say whether the rules for the expansion are completely defined for the example.

```
#define f(a) a * g
#define g   f
f(2)(9)
```

is completely defined as expanding to:

```
2 * f(9)
```

But, if `g`'s definition instead is

```
#define g(a) f(a)
```

then you are correct in that the expansion rules do not specify whether one gets

```
2 * f(9)
```

or

```
2 * 9 * g
```

The Committee intentionally left this behavior ambiguous. We could see no purpose served by specifying the behavior in such borderline cases.

Item 10 (§A.6.2, Doc. No. X3J11/88-124 Letter P14 p. 5)

Summary of Issue: A case of undefined behavior was omitted.

Committee Response:
This editorial change has been made.

X3J11 Response to Letter P15

Correspondent's Name and Address:

Walter J. Murray
2381 Blue Lagoon Drive
Santa Clara, CA 95054

Item 1 (§2.1.1.3, Doc. No. X3J11/88-125 Letter P15 p. 1)

Summary of Issue: Strengthen requirement for diagnostics.

Committee Response:
The Committee chose a different approach to deal with this issue.

A footnote was added instead.

Item 2 (§3.2.1.1, Doc. No. X3J11/88-125 Letter P15 p. 1)

Summary of Issue: Define "integral promotions" on `int` and `long int`.

Committee Response:
This was considered a request for information, not an issue.

The integral promotions apply to `char` and `short` operands which are promoted to type `int` before being evaluated as part of an expression. The usual arithmetic conversions apply to `long` operands.

For other types of operands, the integral promotions have no effect.

Item 3 (§3.2.1.2, Doc. No. X3J11/88-125 Letter P15 p. 1)

Summary of Issue: Clarify wording about unsigned integer conversion.

Committee Response:
No change to the existing wording was considered necessary.

The case where the value cannot be represented is described in the last paragraph of §3.2.1.2, rendering an "else" unnecessary.

Item 4 (§3.3, Doc. No. X3J11/88-125 Letter P15 p. 1)

Summary of Issue: Clarify order of evaluation.

Committee Response:
This was considered a request for information, not an issue.

The order of evaluation is not specified for your example. This has been part of the C language since K&R.

If the + operators are replaced by commas, the order of evaluation is still not specified completely, but a partial ordering is specified in that `f1` must be evaluated before `f2`, and `f3` evaluated before `f4`.

Item 5 (§3.3.2.1, Doc. No. X3J11/88-125 Letter P15 p. 1)

Summary of Issue: Move comment about row-major to footnote.

Committee Response:
The Committee discussed this proposal but decided against it.

It doesn't hurt to have redundancy, and we left this paragraph in the official part of the document for historical reasons more than any other.

Item 6 (§3.4, Doc. No. X3J11/88-125 Letter P15 p. 2)

Summary of Issue: Relax or clarify rules for constant expressions.

Committee Response:
Changes have been made along the lines you suggested.

We included new language in §3.4 to allow an implementation to extend the definition of constant expressions. Hence, some implementations need not issue a diagnostic for this.

Item 7 (§3.5.7, Doc. No. X3J11/88-125 Letter P15 p. 2)

Summary of Issue: Correct typo regarding aggregate initialization.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 8 (§3.5.7, Doc. No. X3J11/88-125 Letter P15 p. 2)

Summary of Issue: Correct wording in array initialization example.

Committee Response:
Changes have been made along the lines you suggested.

Item 9 (§4.1.2, Doc. No. X3J11/88-125 Letter P15 p. 2)

Summary of Issue: Define "external identifier".

Committee Response:
The Committee chose a different approach to deal with this issue.

The term "external identifier" is no longer used.

Item 10 (§4.1.3, Doc. No. X3J11/88-125 Letter P15 p. 2)

Summary of Issue: Reserve identifiers beginning with `E` *etc.* if `<errno.h>` is included.

Committee Response:
Changes have been made along the lines you suggested.

We added a new section clarifying reserved identifiers in all situations.

Item 11 (§4.1.5, Doc. No. X3J11/88-125 Letter P15 p. 2)

Summary of Issue: Clarify definition of `offsetof` macro.

Committee Response:
Changes have been made along the lines you suggested.

Our response to your item 6 (§3.4) covers this issue. Other implementations must, as you point out, use a reserved keyword such as `__offset` to implement this.

Item 12 (§4.13, Doc. No. X3J11/88-125 Letter P15 p. 3)

Summary of Issue: Insert comma after "letters".

Committee Response:
This was accepted as an editorial change to the Standard.

Item 13 (§A.2, Doc. No. X3J11/88-125 Letter P15 p. 3)

Summary of Issue: Clarify wording about expressions of a `for` statement.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 14 (§A.6.3, Doc. No. X3J11/88-125 Letter P15 p. 3)

Summary of Issue: List the nature of diagnostics as implementation defined.

Committee Response:
This was accepted as an editorial change to the Standard.

Item 15 (§R1.6, Doc. No. X3J11/88-125 Letter P15 p. 3)

Summary of Issue: Correct typo about holes in `int`s containing bit fields.

Committee Response:
This was accepted as an editorial change to the Rationale.

Only structures and unions can contain bit fields. There may also be unnamed padding at the end of a structure or union.