Farance Inc.
555 Main Street · New York · New York · 10044-0150 · USA
Telephone: +1 212 486 4700 FAX: +1 212 759 1605
UUCP: uunet!farance!inc E-mail: inc@farance.com
MCI: 372 3616 ESL: 62727960 TWX: 510 100 4863 (FARANCE)

subject: **Model Electronic Document Project Status 1994-06-05**

date: **1994-06-05**

from: **Frank Farance**
+1 212 486 4700
frank@farance.com

DOC# WG14/N351 X3J11/94-036
File: x3j11/pdf/notes/nXX1.mm.100

Since the 1994-04-29 mailing, there have been several developments with the Model Electronic Document Project for X3J11.

X3 Ad Hoc Mechanization Committee

I have spoken several times with Dan Arnold at CBEMA with respect to the X3 Ad Hoc Mechanization Committee. Currently, there is no official standard for X3 documents. However, CBEMA is sending an RFI (Request For Information) on 1994-06-08 to get input on how to manage and distribute electronic documents. The RFI is due by 1994-07-22. I would like to ask X3J11's permission to respond on behalf of X3J11. I would be describing how our model project (in its infant stages) works and making recommendations for X3's handling of standing and transient documents.

Evaluation of Tools

I've evaluated the Adobe Acrobat, Distiller, Exchange, and Reader tools. These tools promise the management of electronic documents from the perspective of several users: the editor, the creator, the "converter", the reader, the publisher. In short, the tools are *very* disappointing. The Acrobat system creates a new type of exchange format: PDF (Portable Document Format). Basically, PDF is *very* similar to Postscript (not a surprise) except that it is a different, less readable (!) syntax. Furthermore, much of the document information (paragraph structure, lists, etc.) are **not** maintained in the PDF structure. I will briefly describe the PDF file format in my presentation.

In summary, the evaluation of publicly available tools has confirmed my original feelings that SGML (or equivalent) is the best choice.

Conversion of Sample Documents

Bill Plauger was kind enough to send me the RR and TC documents in machine readable form. These documents are in Ventura Publisher format. I've written a conversion program (a couple hundred lines of LEX script) to convert Ventura Publisher to the standard format. The conversion process takes about 3-4 minutes machine time and about 30-40 minutes of proofing and checking, i.e., under an hour to convert a document. I've attached some sample pages of the original and the converted documents.

43

I am about to write the conversion program for Microsoft Word RTF (Rich Text Format) files. After I get some samples, I will report back on the progress of this type of conversion.

Please send me machine readable documents so I have more samples.

I will be attempting OCR (Optical Character Recognition) scanning of documents during 1994-07/1994-08.

FTP Site

The Dave Keaton has confirmed that he will be able to maintain the storage of X3J11 documents on his FTP site `ftp.dmk.com`. Dave and I will be talking at the 1994-06 meeting about general procedures. Join in the conversation if you're interested.

Outstanding Items

The following is a summary of items that were on my schedule:

- 1994-06-01: Evaluation of SGML tools complete. **Status:** Done evaluating existing tools.
- 1994-07-15: RC and TC converted to SGML format. Templates published. **Status:** RC and TC are converted to a "standard" format (MM) via a LEX script. Once the SGML template is ready, the conversion program will generate SGML source rather than MM source.
- 1994-08-15: Technical papers converted from the pre/post San Jose and pre/post Tokyo mailings. NOTE: Only those papers with machine readable available. **Status:** Not yet started.
- 1994-12-01: Conversion of C Standard to SGML format. **Status:** Not yet started.

New Schedule

I will publish an updated schedule after the 1994-06 X3J11 meeting.

SAMPLE - ORIGINAL

Defect Report #013

Submission Date: 10 Dec 92
Submittor: WG14
Source: X3J11/90-047 (Sam Kendall)

Question 1

Compatible and composite function types

A fix to both problems Mr. Jones raises in X3J11 Document Number 90-006 is: In subclause 6.5.4.3 on page 68, lines 23-25, change the two occurrences of "its type for these comparisons" to "its type for compatibility comparisons, and for determining a composite type." This change makes the sentences pretty awkward, but I think they remain readable.

This change makes all three of Mr. Jones's declarations compatible:

```
int f(int a[4]);
int f(int a[5]);
int f(int *a);
```

This should be the case; it is consistent with the base document's idea of "rewriting" the parameter type from array to pointer.

Correction

In subclause 6.5.4.3, page 68, lines 22-25, change:

(For each parameter declared with function or array type, its type for these comparisons is the one that results from conversion to a pointer type, as in 6.7.1. For each parameter declared with qualified type, its type for these comparisons is the unqualified version of its declared type.)

to:

(In the determination of type compatibility and of a composite type, each parameter declared with function or array type is taken as having the type that results from conversion to a pointer type, as in 6.7.1, and each parameter declared with qualified type is taken as having the unqualified version of its declared type.)

Question 4

When a structure is incomplete

Reference subclause 6.5.2.3, page 62, lines 25-28:

> If a type specifier of the form
>
> ***struct-or-union identifier***
>
> occurs prior to the declaration that defines the content, the structure or union is an incomplete type.

In the following example, neither the second nor the third occurrence of **`struct foo`** seem adequately covered by this sentence:

```
struct foo {
     struct foo *p;
} a[sizeof (struct foo)];
```

In the second occurrence **`foo`** is incomplete, but since the occurrence is within "the declaration that defines the content," it cannot be said to be "prior" that declaration. In the third occurrence **`foo`** is complete, but again, the occurrence is within the declaration.

To fix the problem, change the phrase "prior to the declaration" to "prior to the end of the ***`struct-declaration-list`*** or ***`enumerator-list`***."

Correction

In subclause 6.5.2.3, page 62, line 27, change:

occurs prior to the declaration that defines the content

to:

occurs prior to the } following the ***`struct-declaration-list`*** that defines the content

SAMPLE - ORIGINAL

Question 5

Enumeration tag anomaly

Consider the following (bizarre) example:

```
enum strange1 {
     a = sizeof (enum strange1)     /* line [2] */
};
enum strange2 {
     b = sizeof (enum strange2 *)   /* line [5] */
};
```

The respective tags are visible on lines [2] and [5] (according to subclause 6.1.2.1, page 20, lines 39-40, but there is no rule in subclause 6.5.2.3, **Semantics** (page 62) that governs their meaning on lines [2] and [5]. Footnote 62 on page 62 seems to be written without taking this case into account.

The first declaration must be illegal. The second declaration should be illegal for simplicity.

Perhaps these declarations are already illegal, since no rule gives them a meaning. To clarify matters, I suggest in subclause 6.5.2.3 appending to page 62, line 35:

> A type specifier of the form
>
> **enum *identifier***
>
> shall not occur prior to the end of the ***enumerator-list*** that defines the content.

If this sentence is not appended, something like it should appear as a footnote.

Correction

Add to subclause 6.5.2.3, page 63, another Example:

An enumeration type is compatible with some integral type. An implementation may delay the choice of which integral type until all enumeration constants have been seen. Thus in:

```
enum f { c = sizeof(enum f) };
```

the behavior is undefined since the size of the respective enumeration type is not necessarily known when **sizeof** is encountered.

46

Defect Report #013

Submission Date: 10 Dec 92

Submittor: WG14

Source: X3J11/90-047 (Sam Kendall)

Question 1

Compatible and composite function types

A fix to both problems Mr. Jones raises in X3J11 Document Number 90-006 is: In subclause 6.5.4.3 on page 68, lines 23-25, change the two occurrences of ``its type for these comparisons'' to ``its type for compatibility comparisons, and for determining a composite type.'' This change makes the sentences pretty awkward, but I think they remain readable.

This change makes all three of Mr. Jones's declarations compatible:

```
int f(int a[4]);
int f(int a[5]);
int f(int *a);
```

This should be the case; it is consistent with the base document's idea of ``rewriting'' the parameter type from array to pointer.

In subclause 6.5.4.3, page 68, lines 22-25, change:

(For each parameter declared with function or array type, its type for these comparisons is the one that results from conversion to a pointer type, as in 6.7.1. For each parameter declared with qualified type, its type for these comparisons is the unqualified version of its declared type.)

to:

(In the determination of type compatibility and of a composite type, each parameter declared with function or array type is taken as having the type that results from conversion to a pointer type, as in 6.7.1, and each parameter declared with qualified type is taken as having the unqualified version of its declared type.)

Question 4

When a structure is incomplete

Reference subclause 6.5.2.3, page 62, lines 25-28:

> If a type specifier of the form
>
> ***`struct-or-union identifier`***
>
> occurs prior to the declaration that defines the content, the structure or union is an incomplete type.

In the following example, neither the second nor the third occurrence of `struct foo` seem adequately covered by this sentence:

SAMPLE - CONVERTED

```
struct foo {
        struct foo *p;
} a[sizeof (struct foo)];
```

In the second occurrence `foo` is incomplete, but since the occurrence is within ``the declaration that defines the content,'' it cannot be said to be ``prior'' that declaration. In the third occurrence `foo` is complete, but again, the occurrence is within the declaration.

To fix the problem, change the phrase ``prior to the declaration'' to ``prior to the end of the ***`struct-declaration-list`*** or ***`enumerator-list`***.''

In subclause 6.5.2.3, page 62, line 27, change:

occurs prior to the declaration that defines the content

to:

occurs prior to the `}` following the ***`struct-declaration-list`*** that defines the content

Question 5

Enumeration tag anomaly

Consider the following (bizarre) example:

```
enum strange1 {
        a = sizeof (enum strange1)      /* line [2] */
};
enum strange2 {
        b = sizeof (enum strange2 *)    /* line [5] */
};
```

The respective tags are visible on lines [2] and [5] (according to subclause 6.1.2.1, page 20, lines 39-40, but there is no rule in subclause 6.5.2.3, **Semantics** (page 62) that governs their meaning on lines [2] and [5]. Footnote 62 on page 62 seems to be written without taking this case into account.

The first declaration must be illegal. The second declaration should be illegal for simplicity.

Perhaps these declarations are already illegal, since no rule gives them a meaning. To clarify matters, I suggest in subclause 6.5.2.3 appending to page 62, line 35:

A type specifier of the form

```
enum identifier
```

shall not occur prior to the end of the ***`enumerator-list`*** that defines the content.

If this sentence is not appended, something like it should appear as a footnote.

Add to subclause 6.5.2.3, page 63, another Example:

An enumeration type is compatible with some integral type. An implementation may delay the choice of which integral type until all enumeration constants have been seen. Thus in:

```
enum f { c = sizeof(enum f) };
```

the behavior is undefined since the size of the respective enumeration type is not necessarily known when `sizeof` is encountered.