Title: Tag Compatibility
Date: 21 December 1995
Author: Tom MacDonald
Cray Research, Inc.
655F Lone Oak Drive
Eagan MN 55121
Email: tam@cray.com
Document Number: WG14 N522 (a.k.a. X3J11/95-123) Irvine
Related Documents: WG14 N453 (a.k.a. X3J11/95-054) Nashua
WG14 N404 (a.k.a. X3J11/95-005) Copenhagen
WG14/N396 (a.k.a. X3J11/94-081) Plano Minutes
DR#139

Abstract:

Currently, two structure, union, and enumeration types declared in separate translation units ignore the tag name when determining if the two types are compatible. The language is better specified if the tag names are considered for all type compatibility (except if no tag name is specified).

Proposal: Change the following words in the current C Standard:

6.1.2.6 Compatible Type and composite type

From:

> Moreover, two structure, union, or enumeration types declared in separate translation units are compatible if they have the same number of members, the same member names, and compatible member types; for two structures, the members shall be in the same order; for two structures or unions, the bit-fields shall have the same widths; for two enumerations, the members shall have the same values.

To:

> Moreover, two structure, union, or enumeration types declared in separate translation units are compatible if their tags and members satisfy the following requirements. If one is declared with a tag, the other shall be declared with the same tag. If both are complete types, then the following additional requirements apply. There shall be a one-to-one correspondence between their members such that each pair of corresponding members are declared with compatible types, and such that if one member of a corresponding pair is declared with a name, the other member is declared with the same name. For two structures, corresponding members shall be declared in the same order. For two structures or unions, corresponding bit-fields shall have the same widths. For two enumerations, corresponding members shall have the same values.

Example:

The following example is no longer considered to be portable because "struct tag1" in no longer compatible with either "struct tag2" or "struct tag3" (different tag names). This is useful because a translator can assume that pointers such as "pst2" and "pst3" refer to different objects. Cray Research compilers already make this assumption when the highest optimization level is specified, and so when the example below is run on a Cray-C90 computer both with and without optimization, the following results are obtained:

```
$ cc -O3 x.c y.c
x.c:
y.c:
$ ./a.out
optimized
$ cc -O0 x.c y.c
x.c:
y.c:
$ ./a.out
unoptimized
```

x.c

```
#include <stdio.h>
int func();

struct tag1 {
   int m1, m2;
} st1;

main() {
   if (func(&st1, &st1)) {
      printf("optimized\n");
   } else {
      printf("unoptimized\n");
   }
}
```

y.c

```
struct tag2 {
   int m1, m2;
};
struct tag3 {
   int m1, m2;
};

int func(struct tag2 *pst2,
         struct tag3 *pst3) {
   pst2->m1 = 2;
   pst3->m1 = 0;    /* alias? */
   return pst2->m1;
}
```

Comments:

The above proposal makes an unnamed struct, union or enum incompatible with a named one. It appears the committee favors this approach.