Document Number: WG14 N498/X3J11 95-099

C9X Revision Proposal

Title: Unnamed structure/union members
Author: Ken Thompson
Author Affiliation: AT&T Bell Labs
Postal Address: Room 2c519; Murray Hill, NJ 07974
E-mail Address: ken@plan9.att.com
Telephone Number: +1 908 582-2394
Fax Number: +1 908 582-5857
Sponsor: _____
Date: Nov 26 1995
Proposal Category:
__ Editorial change/non-normative contribution
__ Correction
XX New feature
__ Addition to obsolescent feature list
__ Addition to Future Directions
__ Other (please specify) _____
Area of Standard Affected:
__ Environment
XX Language
__ Preprocessor
__ Library
__ Macro/typedef/tag name
__ Function
__ Header
__ Other (please specify) _____
Prior Art: Plan 9, Microsoft and GCC C compilers
Target Audience: System Programmers

Related Documents (if any): none

Proposal Attached: Yes

Abstract: Allow structures/unions with no names as members of structures/unions. Allow promotion of names and types of the sub-structure/sub-union members into the names and types of the enclosing structure/union.

Proposal:

This proposal involves 1 syntactic change and 2 semantic changes. The 2 semantic changes should be treated as 2 independent proposals. Both semantic proposals require the syntactic change.

1. Syntactic Change

current syntax:

```
        struct-declaration:
                specifier-qualifier-list struct-declarator-list ;
```

proposed syntax:

```
        struct-declaration:
                specifier-qualifier-list struct-declarator-list ;
                struct-or-union-specifier ;
                typedef-name ;
```

This allows unnamed structure or union (SU) members if they are also SUs. (In the syntax above, the typedef-name must be a typedef of a SU.)

2. Semantic change 1 - name promotion.

A member name is interpreted in a SU as follows:

1. if the member name is a member in the SU, then that member is uniquely chosen regardless of unnamed sub-SUs
2. if the member name is ambiguous in any unnamed sub-SU then the name is ambiguous
3. if the member name is uniquely chosen from exactly one unnamed sub-SU then the name is uniquely chosen
4. if the member name is uniquely chosen from more than one unnamed sub-SU then the name is ambiguous.

An ambiguous member name is illegal and, of course, no match is also illegal.

This proposal does not alter the interpretation of any existing legal programs.

given:

```
struct
{
        struct
        {
                int     aa;
                int     bb;
        };
        int     aa;
        struct
        {
                int     bb;
                int     cc;
                union
                {
                        int     dd;
                };
        };
} x;
```

then:

```
x.aa gets the .aa member of x without any extension
x.bb is ambiguous and therefore illegal
x.cc gets the .cc member of the second
        unnamed sub-structure of x
x.dd gets the .dd member in the unnamed sub-union
        within the second unnamed sub-structure of x.
```

rationale:

Lots of programs contain fictitious SU names to overlay storage. There are many pieces of code with references like "p->s1.u2.s4.y" that can be changed to "p->y" along with very natural rewriting of the SUs.

3. Semantic change 2 - type promotion.

In

an assignment statement of a SU to a variable of the type of an unnamed sub-SU

OR

an assignment statement of a pointer-to SU to a variable of the type of pointer-to an unnamed sub-SU,

then the assignment is of the unnamed SU OR of the pointer-to the unnamed SU.

The same type promotion occurs if a SU (OR a pointer to a SU) is passed as an argument to a function that has been prototyped to be of the type of an unnamed sub-SU.

(note: NOT part of the proposal: It would be consistent, but not useful, to allow this conversion in the subtraction of pointers and comparison of pointers. In the implementation of this extension, modification needs to be added in the compiler only at the place where the constant, 0, is converted to a pointer of the appropriate type.)

If there are two unnamed sub-SUs with the same type then the assignment (or argument passing) is ambiguous and illegal.

This proposal does not alter the interpretation of any existing legal programs.

given:

```
typedef struct
{
        (internal stuff)
} Lock;

void    lock(Lock*);

struct
{
        (internal stuff)
        Lock;
        (more internal stuff)
} x;
```

then:

In a program, it is proper to call the function, lock, with the argument, &x.

The function, lock, expects a pointer-to a Lock structure. The structure, x, has an unnamed sub-structure of type, Lock. Passing &x to lock will cause a type conversion to Lock* by passing the address of the unnamed Lock sub-structure.

rationale:

Any name that the programmer assigns to the Lock sub-structure is simply to tag it in passing it to the lock function. There is no reason to invent these names. This is a limited form of type inheritance.