C2x Proposal: WG14 N2607

Title: **Compatibility of Pointers to Arrays with Qualifiers (updates N2497)**
Authors: **Martin Uecker, University Medical Center Göttingen**
Date: **2020-10-31**

Introduction

Pointer conversion rules regarding qualifiers as specified in **6.5.16.1 Simple assignment** and referenced at various places allow to convert a pointer to a non-qualified type to a pointer to a qualified type. These rule does not work correctly for pointers to arrays. This issue was discussed in detail in **N1923** and wording changes were proposed in **N2497.** This paper proposes revised wording based on the feedback from the committee.

Example:

```
void matrix_fun(int N, const float x[N][N]);

int N = 100;
float x[N][N];

matrix_fun(N, x);
```

Suggested Wording Changes

6.2.5(26)
Any type so far mentioned is an unqualified type. Each unqualified type has several qualified versions of its type, corresponding to the combinations of one, two, or all three of the const, volatile, and restrict qualifiers. The qualified or unqualified versions of a type are distinct types that belong to the same type category and have the same representation and alignment requirements. **An array and its element type are always considered to be identically qualified*) Any other ~~A~~** derived type is not qualified by the qualifiers (if any) of the type from which it is derived.

***) This does not apply to the _Atomic qualifier. Note that qualifiers do not have any direct effect on the array type itself, but affect conversion rules for pointer types that reference an array type.**

6.7.6.2(3) Array Declarators

If, in the declaration "T D1", D1 has one of the forms:

D [type-qualifier-list_opt assignment-expression_opt]
D [type-qualifier-list_opt assignment-expression]
D [type-qualifier-list static assignment-expression]
D [type-qualifier-list_opt]

and the type specified for ident in the declaration "T D" is "derived-declarator-type-list T", then the type specified for ident is "derived-declarator-type-list array of T".145,***)** (See 6.7.6.3 for the meaning of the optional type qualifiers and the keyword static.)

***) The array is considered identically qualified to T according 6.2.5(26).**

6.7.3(10)
If the specification of an array type includes any type qualifiers, **both the array and** the element type ~~**is**~~ **are** so-qualified~~**, not the array type**~~. If the specification of a function type includes any type qualifiers, the behavior is undefined.139)

139) ~~**Both of these**~~ **This** can occur through the use of typedefs. **Note that this rule does not apply to the _Atomic qualifier, and that qualifiers do not have any direct effect on the array type itself, but affect conversion rules for pointer types that reference an array type.**

6.7.3(2)
Types other than pointer types whose referenced type is an object type **and (possibly multi-dimensional) array types with such pointer types as element type** shall not be restrict-qualified.