Document Number: P0283R0
Date: 2016-02-12
Reply to: josedaniel.garcia@uc3m.es
Audience: Evolution Working Group.

Standard and non-standard attributes

J. Daniel Garcia
Computer Science and
Engineering Department
University Carlos III of Madrid

1 Introduction

This paper tries to clarify the use of attributes namespaces. In particular, it tries to answer the following questions:

- What happens if an implementation finds a reference to an attribute namespace that it does not know about?
- What happens if an implementation finds a reference to an unqualified attribute which is not defined in the standard?

Attributes [1] provide a useful way to add annotations to source code with implementation defined effects. Implementations are expected to add their own attribute namespace where their attributes are defined. In fact, scoped attributes —those under a specific namespace— are specified as conditionally supported. This approach provides a clean way for different implementations to add their own attributes.

2 Problem

Attributes have proved to be a very useful way to perform source code annotations. However to increase their use some issues need to be better clarified.

2.1 Handling unknown attribute namespaces

During the committee meeting at Kona it was pointed out that *"We don't have a requirement that implementations ignore attribute namespaces that they do not understand. So my users hide attributes behind macros"*. This is something that needs to be clarified in order to avoid attributes being hidden by macros.

Currently, the standard states in 7.6.1/3 that *"The use of an attribute-scoped-token is conditionally-supported, with implementation-defined behavior"*. Making namespaced attributes conditionally supported means that they will not be understood by implementations that do not cover that namespace.

2.2 Non standard qualified attributes

Currently, the standard states in 7.6.1/5 that *"For an attribute-token not specified in this International Standard, the behavior is implementation-defined"*. However this wording (combined with namespaces being conditionally supported) has resulted in implementations adding a number of non-standard attributes as extensions.

For example, clang/llvm includes the following:

`CarriesDependency`	`CXX11NoReturn`	`Deprecated`	`FallThrough`
`DisableTailCalls`	`NoDuplicate`	`NotTailCalled`	`OptimizeNone`
`TypeVisibility`	`WarnUnusedResult`	`NoSanitize`	`Capability`
`AssertCapability`	`AcquireCapability`	`TryAcquireCapability`	`ReleaseCapability`
`RequiresCapability`	`InternalLinkage`		

In general, the current status quo favors the fact that different implementations pollute the global namespace with non-standard attributes in different ways, making very difficult standardization of attributes in future versions.

3 Proposal

This paper proposes possible solutions for the previously identified problems

3.1 Handling unknown attribute namespaces

Having scoped attributes as conditionally supported provides the degree of freedom that allows an implementation not to support a specific attribute namespace. However, this is not enough to clarify what are the valid options for an implementation when it finds an attribute namespace it does not know about.

This paper proposes to add the following:

All occurrences of an attribute namespace that is not conditionally supported by the implementation will be ignored.

3.2 Non standard unqualified attributes

The current wording of 7.6.1/5 makes all unqualified attributes that are not defined in the standard as implementation defined. This allows implementations to add new attributes to the global namespace.

As it has been mentioned above implementations do already make use of this clause to define their own attributes in the global namespace. This paper proposes to deprecate that use by changing the clause and adding additional rules.

This paper proposes to modify 7.6.1/5 as follows:

Any occurrence of an attribute-token not specified in this International Standard is deprecated.

This paper also proposes to add the following:

For an attribute-scoped-token not specified in this International Standard, the behavior is implementation-defined.

Acknowledgments

Thanks to Chandler Carruth for pointing out initial ideas. Thanks to Bjarne Stroustrup and Michael Wong for useful feedback prior to the writing of this paper.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement n. 609666.

References

[1] Jens Maurer and Michael Wong. Towards support for attributes in C++. Working paper N2761, ISO/IEC JTC1/SC22/WG21, September 2008.