Document Number: P0283R1
Date: 2016-03-15
Reply to: josedaniel.garcia@uc3m.es
Audience: Core Working Group.

Standard and non-standard attributes

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

Changes

Changes from P0283R0 [1]:

- Removed references to the *unqualified attribute* issue as it was rejected by Evolution Working Group.

1 Introduction

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

- What happens if an implementation finds a reference to an attribute namespace that it does not know about?

Attributes [2] 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 improve their use attributes namespaces 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.

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 non-normative note to section 7.6.1 [dcl.attr.grammar]:

[Note: All occurrences of an attribute namespace that is not conditionally supported by the implementation should be ignored. — end note]

Acknowledgments

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

This work has received funding from the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement n. 609666.

References

[1] J. Daniel Garcia. Standard and non-standard attributes. Working paper P0283R0, ISO/IEC JTC1/SC22/WG21, February 2016.

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