Analysis of Include Dependencies in C++ Source Code

Bence Babati, Norbert Pataki, Zoltán Porkoláb

The C++ Standard Template Library is the flagship example for libraries based on the generic programming paradigm. The usage of this library is intended to minimize classical C/C++ errors, but does not warrant bug-free programs [1]. Furthermore, many new kinds of errors may arise from the inaccurate use of the generic programming paradigm, like dereferencing invalid iterators or misunderstanding remove-like algorithms [4].

Unfortunately, the C++ Standard does not define which standard header includes another standard headers [4]. It is easy to write code that works perfectly on an implementation but fails to compile with another implementation of STL. These unportable codes should be result in compilation error with every STL implementation [3]. However, in this case the compiler does not warn us that this code is erroneous.

In this paper we present our tool that is based on the Clang [2]. This tool is able to detect the missing include directives that are patched by the STL implementation's internal structure. It also reports the unnecessary include directives to avoid extra compilation time. The background of our tool is discovered and we briefly present the underlying data structures and algorithms. We analyse how these problems occur in open source libraries and programs. Which environment proves oneself to be lazy or strict? How the developers take advantage of this portability issue?

References

[1] Horváth, G., Pataki, N.: *Clang matchers for verified usage of the C++ Standard Template Library*, Annales Mathematicae et Informaticae, Vol. **44** (2015), pp. 99–109.

[2] Lattner, C.: *LLVM and Clang: Next Generation Compiler Technology*, The BSD Conference, 2008.

[3] Meyers, S.: *Effective STL*, Addison-Wesley, 2001.

[4] Pataki, N.: *C++ Standard Template Library by Safe Functors*, in Proc. of 8th Joint Conference on Mathematics and Computer Science, MaCS 2010, Selected Papers, pp. 363–374.

[5] B. Stroustrup: *The C++ Programming Language*, Addison-Wesley, special edition, 2000.