為何某些公司不允許使用C++ STL?

陳甫鵬 C語言與C++編程 今天

> **來自：知乎，作者：陳甫鵬**
> **鏈接：https://www.zhihu.com/question/20201972/answer/23454845**

> **你們公司允許使用C++STL嗎?**

最初開始禁用C++ STL，是因為早期項目編碼實踐中留下的慣例，被後來的程序員繼承下來。老項目中這種選擇尤其地多。不過如果有人將其上升到公司行為在不同項目中全面禁用STL，則沒有必要，而且我傾向於**做這種決定的人並不理解C++編譯系統。**

一般來說，項目中禁用C++多見於**兩種具體場景**：項目的產出產品為函數庫OR需要引用第三方函數庫。

具體地來說，有**三個主要原因**：

第一個原因是二進制邊界混亂。對需要在項目中使用第三方函數庫的程序員來說，二進制邊界是個頭痛的問題。C++在這一方面本身就處理得不算好，加上模板後起到的是雪上加霜的後果。沒有經驗的程序員會貪圖方便而在公開頭文件中使用C++模板，如果這時調用方的編譯器選項設置或STL版本和編譯方不同，那麼就可能出現同樣的頭文件在不同的環境下二進制佈局不符的情況。

順便說一句，在過去十年裡，各個主流編譯器附帶的STL 版本變化節奏不慢，所以這種由於編譯環境不同而導致的bug 並不算罕見，但缺乏彙編知識的用戶難以排查。

第二個原因是不願使用異常。如今除了Android上的STLPort關閉異常，大部分主流C++ STL實現裡都無法脫離異常使用STL。異常帶來的問題主要是兩個：**性能下降，代碼膨脹。**這幾年C++編譯器在性能方面的改進很多，good path的性能問題已經基本沒有，但代碼膨脹問題卻沒有太多改善，甚至這個性能問題的一部分解決方案就是以代碼膨脹為代價。我寫過一篇短文比對過Android上gcc 4.6在有無異常的情況下的彙編代碼邏輯，可以看到，啟動異常時生成的彙編代碼量多出了相當一部分（我的例子中是50%），用於處理各種隱含代碼中的異常問題。這一條在手機系統中有時候會引起意想不到的麻煩，比如軟件升級後導致app在低存儲容量的手機中安裝失敗。順便說一句，這個問題並不是gcc獨有，clang上生成的代碼是一樣的。參考：http://dummydigit.net/posts/2014-01-01-23-30-1.html

最后一个原因是 C 兼容。严格地说，STL 在这个问题上算是躺枪，这个坑在很多具体的场景中也是因为异常而引入，但这个问题的麻烦程度比前两个问题更高。比如 gcc 在编译纯 C 代码时默认关闭 -fexceptions 选项，因此这样编译出来的代码中没有异常处理相关的栈展开。如果某个 C++ 项目引用了一个第三方 C 项目，它很难确保那个 C 项目给出的二进制代码中正确进行了异常处理并保证代码服从异常安全操作。这种场景下混用 C/C++ ，就可能在抛出异常时莫名其妙地崩溃或者出现 C 代码区段中的资源泄漏，特别是 expat 那种大量利用回调的代码结构。要规避这种风险并非不可能，但需要 C 的架构部分做修改，比如使用 DOM 那种树形结构，这种做法对历史项目而言又很难办到。换言之，如果一个项目出于种种原因需要保持 C 兼容，而 STL 就属于其中一个不可控的变数，与其相信程序员不犯错，不如直接禁用更可控一些。参考：Code Gen Options

要解决二进制相关的问题很简单：整个项目的所有相关代码在同一个代码基上编译，强制打开编译选项添加异常代码，并去除一切二进制依赖。但对很多小公司来说，引入这样的系统对配置管理的要求较高。如果一部分依赖关系来自自己并不了解的第三方代码，轻易修改编译选项可能带来的风险与第三方代码库的规模成正比。退一步说，即便团队里真的有强大的配置管理工程师能够搞定一切，他们也不会有能力解决代码膨胀问题，除非他们有权决定换一个编译器。相比之下，前面朋友所说的所谓性能或者编译出错时糟糕的可读性，在我看来反倒是次要因素，而且这些缺陷都正在新的编译器中逐步得到解决或改善，比如 clang。

所以那些选择禁用 STL 的早期项目负责人，总有一些确实的理由，没有人那么别扭地想跟开发效率过不去。至于后来的人是真的仔细想过这些细节还是人云亦云，那是另一个问题。

--- EOF ---