<https://blog.csdn.net/carol123456/article/details/76461311>

代码里一直有一个warning，因为不影响编译，一直没有改，谁知道报bug了，就改了下。

string strA = "text";

编译的时候报warning: ISO C++ forbids converting a string constant to 'char\*'

改也很简单

string strA = (char\*)"text";

搜索了一下

<https://stackoverflow.com/questions/1524356/c-deprecated-conversion-from-string-constant-to-char>

|  |
| --- |
| This is an error message you see whenever you have a situation like the following:  char\* pointer\_to\_nonconst = "string literal";  Why? Well, C and C++ differ in the type of the string literal. In C the type is array of char and in C++ it is constant array of char. In any case, you are not allowed to change the characters of the string literal, so the const in C++ is not really a restriction but more of a type safety thing. A conversion from const char\* to char\* is generally not possible without an explicit cast for safety reasons. But for backwards compatibility with C the language C++ still allows assigning a string literal to a char\* and gives you a warning about this conversion being deprecated.  So, somwehere you are missing one or more consts in your program for const correctness. But the code you showed to us is not the problem as it does not do this kind of deprecated conversion. The warning must have come from some other place. |

C和C++不同的字符串类型。C类型为char的数组，C++是字符常量数组。在任何情况下，不允许你改变字符串的字符，所以在C++的const是不是一个真正的限制，但更多的类型安全的事。如果没有出于安全原因的显式转换，从const char到char \*的转换通常是不可能的。但向后兼容C语言，C++仍然允许指定一个字符串到字符串文字，给你一个关于这个转换被弃用的警告性。

|  |
| --- |
| The warning:  deprecated conversion from string constant to 'char\*'  is given because you are doing somewhere (not in the code you posted) something like:   1. void foo(char\* str); 2. foo("hello");   The problem is that you are trying to convert a string literal (with type const char[]) to char\*.  You can convert a const char[] to const char\* because the array decays to the pointer, but what you are doing is making a mutable a constant.  This conversion is probably allowed for C compatibility and just gives you the warning mentioned. |