[**C++中正确使用PRId64**](http://blog.csdn.net/win_lin/article/details/7912693)

标签： [c++](http://www.csdn.net/tag/c%2b%2b)[macros](http://www.csdn.net/tag/macros)[compiler](http://www.csdn.net/tag/compiler)[printing](http://www.csdn.net/tag/printing)[library](http://www.csdn.net/tag/library)

2012-08-27 16:25 9938人阅读 [评论](http://blog.csdn.net/win_lin/article/details/7912693#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/win_lin/article/details/7912693#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

例子参考高性能流媒体服务器SRS：<https://github.com/winlinvip/simple-rtmp-server>

SRS中广泛使用PRId64实现32和64位系统通用。

c++使用PRID64，需要两步：

1. 包含头文件：<inttypes.h>
2. 定义宏：\_\_STDC\_FORMAT\_MACROS，可以通过编译时加-D\_\_STDC\_FORMAT\_MACROS，或者在包含文件之前定义这个宏。

int64\_t用来表示64位整数，在32位系统中是long long int，在64位系统中是long int，所以打印int64\_t的格式化方法是：

**[cpp]** [view plain](http://blog.csdn.net/win_lin/article/details/7912693) [copy](http://blog.csdn.net/win_lin/article/details/7912693)

[在CODE上查看代码片](https://code.csdn.net/snippets/168849)

1. printf("%ld", value); // 64bit OS
2. printf("%lld", value); // 32bit OS

当然有跨平台的方法：

**[cpp]** [view plain](http://blog.csdn.net/win_lin/article/details/7912693) [copy](http://blog.csdn.net/win_lin/article/details/7912693)

[在CODE上查看代码片](https://code.csdn.net/snippets/168849)

1. #include <inttypes.h>
2. printf("%" PRId64 "\n", value);
3. // 相当于64位的：
4. printf("%" "ld" "\n", value);
5. // 或32位的：
6. printf("%" "lld" "\n", value);

其中，printf("abc" "def" “ghi")这样写多个字符串是没有问题的。

但是，死活都编译不过，错误是：error: expected ‘)’ before ‘PRId64’

找了一下这个宏的定义，/usr/include/inttypes.h：

**[cpp]** [view plain](http://blog.csdn.net/win_lin/article/details/7912693) [copy](http://blog.csdn.net/win_lin/article/details/7912693)

[在CODE上查看代码片](https://code.csdn.net/snippets/168849)

1. /\* The ISO C99 standard specifies that these macros must only be
2. defined if explicitly requested.  \*/
3. #if !defined \_\_cplusplus || defined \_\_STDC\_FORMAT\_MACROS
5. # if \_\_WORDSIZE == 64
6. #  define \_\_PRI64\_PREFIX    "l"
7. #  define \_\_PRIPTR\_PREFIX   "l"
8. # else
9. #  define \_\_PRI64\_PREFIX    "ll"
10. #  define \_\_PRIPTR\_PREFIX
11. # endif
13. /\* Macros for printing format specifiers.  \*/
15. /\* Decimal notation.  \*/
16. # define PRId8      "d"
17. # define PRId16     "d"
18. # define PRId32     "d"
19. # define PRId64     \_\_PRI64\_PREFIX "d"

原来这个是定义给c用的，C++要用它，就要定义一个\_\_STDC\_FORMAT\_MACROS宏显示打开它。

**[cpp]** [view plain](http://blog.csdn.net/win_lin/article/details/7912693) [copy](http://blog.csdn.net/win_lin/article/details/7912693)

[在CODE上查看代码片](https://code.csdn.net/snippets/168849)

1. /\* test\_int64.cpp
2. g++ -D\_\_STDC\_FORMAT\_MACROS -o test\_int64 -g -O0 test\_int64.cpp
3. \*/
4. #include <stdio.h>
5. #include <inttypes.h>
7. **int** main(**int** argc, **char**\*\* argv){
8. int64\_t value = 0xFFFFFFFFFFFF;
9. printf("int64\_t=%"PRId64", sizeof(int64\_t)=%d\n", value, **sizeof**(int64\_t));
10. }

编译并执行：

g++ -D\_\_STDC\_FORMAT\_MACROS -o test\_int64 -g -O0 test\_int64.cpp

./test\_int64

int64\_t=281474976710655, sizeof(int64\_t)=8

对于C++新标准-std=[**C++**](http://lib.csdn.net/base/cplusplus)0x，还可以使用更好的方式：

**[cpp]** [view plain](http://blog.csdn.net/win_lin/article/details/7912693) [copy](http://blog.csdn.net/win_lin/article/details/7912693)

[在CODE上查看代码片](https://code.csdn.net/snippets/168849)

1. /\* test\_int64\_1.cpp
2. g++ -o test\_int64\_1 -g -O0 test\_int64\_1.cpp
3. \*/
4. #include <stdio.h>
5. #include <cinttypes>
6. **using** **namespace** std;
8. **int** main(**int** argc, **char**\*\* argv){
9. int64\_t value = 0xFFFFFFFFFFFF;
10. printf("int64\_t=%"PRId64", sizeof(int64\_t)=%d\n", value, **sizeof**(int64\_t));
11. }

不用定义那个宏了，编译和执行：

g++ -o test\_int64\_1 -g -O0 test\_int64\_1.cpp -std=c++0x

./test\_int64\_1

int64\_t=281474976710655, sizeof(int64\_t)=8

当然得指定一个新的参数：-std=c++0x，否则会报错“#error This file requires compiler and library support for the upcoming ISO C++ standard, C++0x. This support is currently experimental, and must be enabled with the -std=c++0x or -std=gnu++0x compiler options.”

若能使用较新的g++编译，可以使用后者，否则可以用前者直接定义宏。