C++中namespace的使用

http://blog.sina.com.cn/s/blog_986c99d601010hiv.html

命名空间(namespace)是一种描述逻辑分组的机制，可以将按某些标准在逻辑上属于同一个任务中的所有类声明放在同一个命名空间中。标准C++库（不包括标准C库）中所包含的所有内容（包括常量、变量、结构、类和函数等）都被定义在命名空 间std（standard标准）中了。

定义命名空间

有两种形式的命名空间——有名的和无名的。

命名空间的定义格式为：（取自C++标准文档）

```
named-namespace-definition:
    namespace identifier { namespace-body }
unnamed-namespace-definition:
    namespace { namespace-body }
namespace-body:
    declaration-seqopt
```

```
有名的命名空间：
    namespace 命名空间名 {
        声明序列可选
    }
无名的命名空间：
    namespace {
        声明序列可选
    }
```

命名空间的成员，是在命名空间定义中的花括号内声明了名称。可以在命名空间的定义内，定义命名空间 的成员（内部定义）。也可以只在命名空间的定义内声明成员，而在命名空间的定义之外，定义命名空间的成员（外部定义）。

命名空间成员的外部定义的格式为：

命名空间名::成员名 ……

例如：

```
// out.h
namespace Outer { // 命名空间Outer的定义
    int i; // 命名空间Outer的成员i的内部定义
    namespace Inner { // 子命名空间Inner的内部定义
        void f() { i++; } // 命名空间Inner的成员f()的内部定义，其中的i为Outer::i
        int i;
        void g() { i++; } // 命名空间Inner的成员g()的内部定义，其中的i为Inner::i
        void h(); // 命名空间Inner的成员h()的声明
    }
    void f(); // 命名空间Outer的成员f()的声明
    // namespace Inner2; // 错误，不能声明子命名空间
}
void Outer::f() {i--;} // 命名空间Outer的成员f()的外部定义
void Outer::Inner::h() {i--;} // 命名空间Inner的成员h()的外部定义
// namespace Outer::Inner2 {} // 错误，不能在外部定义子命名空间
```

注意：

不能在命名空间的定义中声明（另一个嵌套的）子命名空间，只能在命名空间的定义中定义子命名空间。

也不能直接使用“命名空间名::成员名 ……”定义方式，为命名空间添加新成员，而必须先在命名空间的定义中添加新成员的声明。另外，命名空间是开放的，即可以随时把新的成员名称加入到已有的命 名空间之中去。方法是，多次声明和 定义同一命名空间，每次添加自己的新成员和名称。例如：

```
namespace A {
    int i;
    void f();
} // 现在A有成员i和f()
namespace A {
    int j;
    void g();
} // 现在A有成员i、f()、j和g()
```

还可以用多种方法，来组合现有的命名空间，让它们为我所用。例如：

```
namespace My_lib {
    using namespace His_string;
    using namespace Her_vector;
    using Your_list::List;
    void my_f(String &, List &);
}
……
using namespace My_lib;
……
Vector<String> vs[5];
List<int> li[10];
my_f(vs[2], li[5]);
```

使用命名空间

作用域解析运算符（::）

对命名空间中成员的引用，需要使用命名空间的作用域解析运算符::。例如：

```
// out1.cpp
#include "out.h"
#include <iostream>
int main ( ) {
    Outer::i = 0;
    Outer::f(); // Outer::i = -1;
    Outer::Inner::f(); // Outer::i = 0;
    Outer::Inner::i = 0;
    Outer::Inner::g(); // Inner::i = 1;
    Outer::Inner::h(); // Inner::i = 0;
    std::cout << "Hello, World!" << std::endl;
    std::cout << "Outer::i = " << Outer::i << ", Inner::i = " << Outer::Inner::i << std::endl;
}
```

using指令（using namespace）

为了省去每次调用Inner成员和标准库的函数和对象时，都要添加Outer::Inner::和 sta::的麻烦，可以使用标准C++的using编译指令来简化对命名空间中的名称的使用。格式为：

using namespace 命名空间名[::命名空间名……];

在这条语句之后，就可以直接使用该命名空间中的标识符，而不必写前面的命名空间定位部分。因为 using指令，使所指定的整个命名空间中的所有成员都直接可用。例如：

```
// out2.cpp
#include "out.h"
#include <iostream>
// using namespace Outer; // 编译错误，因为变量i和函数f()有名称冲突
using namespace Outer::Inner;
using namespace std;
int main ( ) {
    Outer::i = 0;
    Outer::f(); // Outer::i = -1;
    f(); // Inner::f()，Outer::i = 0;
```

```
    i = 0; // Inner::i
    g(); // Inner::g()，Inner::i = 1;
    h(); // Inner::h()，Inner::i = 0;
    cout << "Hello, World!" << endl;
    cout << "Outer::i = " << Outer::i << ", Inner::i = " << i << endl;
```