设计模式之单例模式，懒汉模式与饿汉模式

作者：bfchen
原文链接：https://ld246.com/article/1513423968065
来源网站：链滴

<h3>单例模式是一种常用的软件设计模式，它保证在软件运行过程中对象在内存中只存在一份实例
单例模式通常有懒汉模式与饿汉模式。</h3>
<h4>1.懒汉模式</h4>
懒汉模式是在第一次调用的时候才创建实例，但是在多线程的环境下很可能创建多个实例，因此
线程不安全的。
C++代码：
<pre><code>class Singleton
{
    protected:
        Singleton(){}
        Singleton(const Singleton& singleton)(){}
    private:
        static Singleton* p;
    public:
        static Singleton* instance();
}
Singleton* Singleton::p = NULL;

Singleton* Singleton::instance()

{

if(NULL == p)

{

//多线程不安全进入

p = new Singleton();

}

return p;

}

</code></pre>

这里切记除了给类添加一个保护的构造函数外，还需添加一个保护的拷贝构造函数，不然编译器
给我们添加一个public访问标志的构造函数，这时我们按照下面这样可以生成多个实例：
<pre><code>Singleton* singleton = Singleton::instance();
//调用编译器提供的默认构造函数
Singleton singleton1 = *singleton;
Singleton singleton2 = *singleton;
</code></pre>

<h4>2.饿汉模式</h4>
饿汉模式是在程序加载的时候就已经创建了一个实例，不存在多线程环境下创建多实例的问题，
此是线程安全的。
C++代码：
<pre><code>class Singleton
{
    protected:
        Singleton(){}
        Singleton(const Singleton& singleton)(){}
    private:

```
        static Singleton* p;
    public:
        static Singleton* instance();
}
Singleton* Singleton::p = new Singleton();

Singleton* Singleton::instance()
{
return p;
}</code></pre>
```

原文链接：设计模式之单例模式，懒汉模式与饿汉模式