<https://www.cnblogs.com/qiaoconglovelife/p/5851163.html>

**1 教科书里的单例模式**

　　我们都很清楚一个简单的单例模式该怎样去实现：构造函数声明为private或protect防止被外部函数实例化，内部保存一个private static的类指针保存唯一的实例，实例的动作由一个public的类方法代劳，该方法也返回单例类唯一的实例。

　　上代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class singleton  {  protected:      singleton(){}  private:      static singleton\* p;  public:      static singleton\* instance();  };  singleton\* singleton::p = NULL;  singleton\* singleton::instance()  {      if (p == NULL)          p = new singleton();      return p;  } |

　　这是一个很棒的实现，简单易懂。但这是一个完美的实现吗？不！该方法是线程不安全的，考虑两个线程同时首次调用instance方法且同时检测到p是NULL值，则两个线程会同时构造一个实例给p，这是严重的错误！同时，这也不是单例的唯一实现！

**2 懒汉与饿汉**

　　单例大约有两种实现方法：懒汉与饿汉。

* + 懒汉：故名思义，不到万不得已就不会去实例化类，也就是说在第一次用到类实例的时候才会去实例化，所以上边的经典方法被归为懒汉实现；
  + 饿汉：饿了肯定要饥不择食。所以在单例类定义的时候就进行实例化。

　　特点与选择：

* + 由于要进行线程同步，所以在访问量比较大，或者可能访问的线程比较多时，采用饿汉实现，可以实现更好的性能。这是以空间换时间。
  + 在访问量较小时，采用懒汉实现。这是以时间换空间。

**3 线程安全的懒汉实现**

　　线程不安全，怎么办呢？最直观的方法：加锁。

* **方法1：加锁的经典懒汉实现：**

[复制代码](javascript:void(0);)

class singleton

{

protected:

singleton()

{

pthread\_mutex\_init(&mutex);

}

private:

static singleton\* p;

public:

static pthread\_mutex\_t mutex;

static singleton\* initance();

};

pthread\_mutex\_t singleton::mutex;

singleton\* singleton::p = NULL;

singleton\* singleton::initance()

{

if (p == NULL)

{

pthread\_mutex\_lock(&mutex);

if (p == NULL)

p = new singleton();

pthread\_mutex\_unlock(&mutex);

}

return p;

}

[复制代码](javascript:void(0);)

* **方法2：内部静态变量的懒汉实现**

　　此方法也很容易实现，在instance函数里定义一个静态的实例，也可以保证拥有唯一实例，在返回时只需要返回其指针就可以了。推荐这种实现方法，真得非常简单。

[复制代码](javascript:void(0);)

class singleton

{

protected:

singleton()

{

pthread\_mutex\_init(&mutex);

}

public:

static pthread\_mutex\_t mutex;

static singleton\* initance();

int a;

};

pthread\_mutex\_t singleton::mutex;

singleton\* singleton::initance()

{

pthread\_mutex\_lock(&mutex);

static singleton obj;

pthread\_mutex\_unlock(&mutex);

return &obj;

}

[复制代码](javascript:void(0);)

**4 饿汉实现**

　　为什么我不讲“线程安全的饿汉实现”？因为饿汉实现本来就是线程安全的，不用加锁。为啥？自己想！

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class singleton  {  protected:      singleton()      {}  private:      static singleton\* p;  public:      static singleton\* initance();  };  singleton\* singleton::p = new singleton;  singleton\* singleton::initance()  {      return p;  } |

　　是不是特别简单呢？

　　以空间换时间，你说简单不简单？

　　面试的时候，线程安全的单例模式怎么写？肯定怎么简单怎么写呀！饿汉模式反而最懒**[**正经脸**]**！