w和delete的底层实现原理

Heaphaestus，RC 2019-05-10 22:57:42 2024 收藏 8

分类专栏： C++

w/new[]和delete/delete[]是什么?

new[]和delete/delete[]是**操作符**；**是C++用来实现动态内存管理的操作符；**

ew/new[] 操作符是用来申请空间的；

elete/delete[]操作数是用来释放动态申请出来的空间；

w/delete的实现原理

]delete是用户进行动态内存申请和释放的操作符，operator new 和operator delete是系统
]全局函数，
E底层调用operator new全局函数来申请空间;
e在底层通过operator delete全局函数来释放空间;

tor new ()全局函数原型：

```
/*
operator new：该函数实际通过malloc来申请空间，当malloc申请空间成功时直接返回；申请空间失败，尝试
执行空 间不足应对措施，如果改应对措施用户设置了，则继续申请，否则抛异常。
*/
void *__CRTDECL operator new(size_t size) _THROW1(_STD bad_alloc)
{
 // try to allocate size bytes
 void *p;
 while ((p = malloc(size)) == 0)
 if (_callnewh(size) == 0)
 {
 // report no memory
 // 如果申请内存失败了，这里会抛出bad_alloc 类型异常
 static const std::bad_alloc nomem;
 _RAISE(nomem);
 }
 return (p);
}
```

tor delete ()全局函数原型：

```
/*
operator delete: 该函数最终是通过free来释放空间的
*/
void operator delete(void *pUserData)
{
    _CrtMemBlockHeader * pHead;

    RTCCALLBACK(_RTC_Free_hook, (pUserData, 0));

    if (pUserData == NULL)
        return;

    _mlock(_HEAP_LOCK); /* block other threads */
    __TRY
        /* get a pointer to memory block header */
        pHead = pHdr(pUserData);

        /* verify block type */
        _ASSERTE(_BLOCK_TYPE_IS_VALID(pHead->nBlockUse));

         _free_dbg( pUserData, pHead->nBlock
```

```
    __FINALLY
        _munlock(_HEAP_LOCK); /* release other threads */
    __END_TRY_FINALLY

     return;
}
/*
free的实现
*/
#define free(p) _free_dbg(p, _NORMAL_BLOCK)
```

这两个全局函数的实现，知道了**operator new 实际是通过malloc来申请空间的，operator e实际是通过free来释放空间的；**

对于不同的类型，new和delete的处理方式是不同的：

内置类型：
如果申请的是内置类型的空间，new和malloc，delete和free基本类似；
不同之处：
new在申请空间失败时会抛异常；
malloc在申请空间失败时会返回NULL；

自定义类型：

- new的原理：
    1. 调用operator new函数申请空间；
    2. 在申请的空间上执行构造函数，完成对象的构造；
- delete的原理：
    1. 在空间上执行析构函数，完成对象中资源的清理工作；
    2. 调用operator delete函数释放对象的空间；
- new[N]的原理：
    1. 调用operator new[]函数，在operator new[]中实际调用operator new函数完成N个对象空间的申请；
    2. 在申请的空间上执行N次构造函数；
- delete[N]的原理：
    1. 在释放的对象空间上执行N次析构函数，完成N个对象中资源的清理；
    2. 调用operator delete[]释放空间，实际在operator delete[]中调用operator delete来释放空间；

这里，可能会有读者想问在new[T]和delete[T]过程中如何确定本次调用的是第几次函数？

对new[ ]时，调用operator new[](size_t size) 函数，传参时传入的不是 sizeof(类型)*对象个数，而是**在对象数组的大小上加上一个额外数据，用于编译器区分对象数组指针和对象指针以及对象大小。**在VS2008下这个额外数据占4个字节，一个int大小。
这个额外数据对外是不可见的。

Running