# C++

## STL之内存管理

STL的内存管理分为两级

第一个级别用于较大内存分配与释放的管理(>128byte), malloc\_alloc

第二个级别用于小于128byte的内存管理,default\_alloc

stl有很多种allocator，根据C++的标准，stl的allocator把对象的申请和释放分为四个步骤：

（1） 申请内存空间，对应的函数是allocator：：allocate（）

（2） 执行构造函数，对应的函数是allocator：：construct（）

（3） 执行析构函数，对应的函数是allocator：：destory（）

（4） 释放内存空间，对应的函数是allocator：：的allocate（）

## STL相关概念

1. SGI 全称为 Silicon Graphics Inc. 硅图公司，SGI\_STL是SGI实现的C++的[标准模板库](https://so.csdn.net/so/search?q=%E6%A0%87%E5%87%86%E6%A8%A1%E6%9D%BF%E5%BA%93&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_20756957/article/details/_blank)。[美国](http://baike.baidu.com/view/2398.htm" \t "https://blog.csdn.net/pizi0475/article/details/_blank)硅图公司成立于1982年，是一个生产高性能计算机系统的跨国公司，总部设在美国加州[旧金山](http://baike.baidu.com/view/7407.htm" \t "https://blog.csdn.net/pizi0475/article/details/_blank)[硅谷](http://baike.baidu.com/view/3243.htm" \t "https://blog.csdn.net/pizi0475/article/details/_blank)

## \_default\_alloc\_template分配器

这个分配器采用的是内存池的思想，有效的避免了内碎片的问题，内碎片是已经被分配出去了，外碎片是由于太小无法分配出去的空间。

（1）如果申请的空间大于128bytes，就将申请的操作移交给\_malloc\_alloc\_template分配器去处理，如果申请内存大小小于128bytes，就从本分配器维护的内存池中分配内存。

（2）该分配器用空闲链表的方式维护内存池中的空闲空间，空闲链表的形状类似于下图所示：

## 总结

以上就是今天要讲的内容，后续会有更多内容。

## 参考资料

版权声明：本文参考了其他资料和CSDN博主的文章，遵循CC 4.0 BY-SA版权协议，现附上原文出处链接及本声明。

1. https://blog.csdn.net/cgcoder/article/details/74468915?spm=1001.2101.3001.6650.7&utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EVECTOR\_COMMERCIAL%7ERRF-7-74468915-blog-73730267.235%5Ev43%5Epc\_blog\_bottom\_relevance\_base9&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7EVECTOR\_COMMERCIAL%7ERRF-7-74468915-blog-73730267.235%5Ev43%5Epc\_blog\_bottom\_relevance\_base9&utm\_relevant\_index=14
2. [https://blog.csdn.net/tangchenchan/article/details/46794633](https://blog.csdn.net/tangchenchan/article/details/46794633" \t "https://blog.csdn.net/tangchenchan/article/details/_blank)
3. [https://blog.csdn.net/m0\_37947204/article/details/80315111](https://blog.csdn.net/m0_37947204/article/details/80315111" \t "https://linxuejiao.blog.csdn.net/article/details/_blank)
4. [https://blog.csdn.net/Mr\_Garfield\_\_/article/details/81274529](https://blog.csdn.net/Mr_Garfield__/article/details/81274529" \t "https://blog.csdn.net/Mr_Garfield__/article/details/_blank)