第一章：STL概论

STL六大组件：容器，算法，迭代器，仿函数，配接器，配置器

第二章：空间配置器

空间配置器：为STL存放数值的各种容器配置空间的，空间包括但不限于内存

allocator有许多接口

自行实现一个简单的空间配置器：主要实现了分配内存并返回指针给对象的功能，用到了new，delete等运算符，还有对应对象的构造函数析构函数等。

次配置力的SGI空间配置器：SGI STL每一个容器都已经指定缺省的空间配置器为alloc，SGI也有一个符合标准的名为allocator的空间配置器，但是SGI不建议我们使用，因为效率不佳，只是把C++的new和delete做了一层包装。

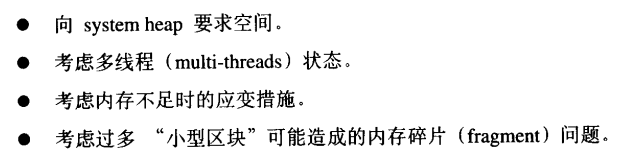
一般的内存分配：new和delete new先分配内存，再构造对象内容，delete先调用析构再释放内存

在STL空间配置器中：两阶段被分开，alloc::allocate()配置内存，alloc::deallocate()释放内存 对象构造由::construct()负责，析构由::destroy()负责

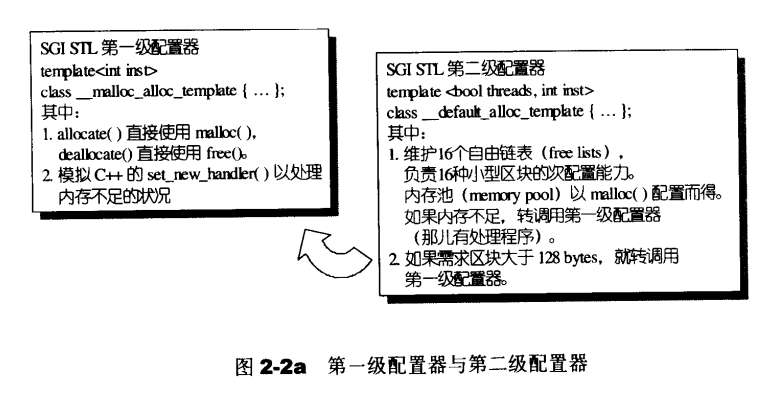
配置器定义在<memory>中：而SGI<memory>包含stl\_alloc.h和stl\_construct.h两个文件，stl\_alloc.h定义了一二级配置器，stl\_construct.h定义了全局函数construct() destroy()，负责对象的构造和析构。

stl\_construct.h中使用construct()接收一个指针p和一个初值value，这个函数用途是将初值设定到指针所指的空间上，new来构造对象，返回指针，destory()有直接析构的版本，接收指针，将指针所指之物析构掉，有接收迭代器的版本，将first和last范围的析构掉

stl\_alloc.h控制内存配置和释放，设计哲学

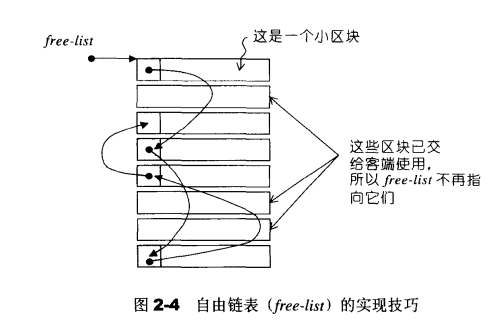


为了避免小型区块造成内存破碎，SGI使用了双层级配置器，第一层直接使用malloc和free，第二层则根据配置区间来决定，超过128字节使用第一级配置器，小于则使用内存池整理方式



C++ new handler机制是：你可以要求系统在内存配置需求无法满足时，调用一个你指定的函数。可以重复使用处理函数直到内存足够。

SGI STL二级配置器是每次配置一大块内存，并维护对应之自由链表。下次若再有相同的内存需求，就直接从free-lists中播出，如果客端释放归还小额区块，就由配置器会受到free-list中，并且SGI第二级配置器会主动将任何小额区块内存需求量上调至8的倍数并维护16个free-lists，为了节省内存，链表节点还用的是union，一处两用



空间配置函数allocate()，身为配置器的标准接口函数。首先判断区块大小，大于128byte调用第一级配置器，小于128byte检查对应的free list。如果freelist之内有可用的区块，就直接拿来用，如果没有可用区块，就将区块大小上调8倍数边界，然后调用refill()，准备为free list重新填充空间

空间释放函数deallocate()，大于128bytes就调用第一季配置器，小于128bytes就找出对应的freelist，将区块回收

refill()函数：当freelist没有区块了，就调用refill()函数，为freelist重新填充空间。新的空间取自内存池。缺省取得20个新节点。

内存池：从memory pool取得空间给free list使用，是chunk\_alloc()的工作。先判断内存池的容量，够就直接给20个到freelist，不够二十但还有一个往上就给这么多，完全没了就从堆里再申请新的内存给内存池，如果整个系统堆内存都不够了，再寻找没使用完区块的另一个freelist，找到了就分一点给现在这个，找不到就要用第一级配置器了，它有out of memory机制，或许能分配成功，最终失败则发出badalloc异常。