# 引言

## 内存池简介

内存池是一种内存分配方式，通常我们习惯用malloc和new进行内存申请，这样做的缺点在于所申请的内存块大小不定，当频繁使用时，会造成大量的碎片进而降低性能。

## 内存池的优点

内存池是在真正使用内存前，预先申请分配一定数量、大小相等的内存块留作备用。当有新的内存需求时，就从内存池中分一部分内存块。若内存块不够再继续，就申请新的内存。这样做的一个显著优点是，使得内存分配效率得到提升。

## 内存池的分类

按照程序定义的内存池根据不同的场景又有不同的类型。从线程安全角度分，内存池可以分为单线程和多线程内存池。单线程内存池整个生命周期只能被一个线程使用，因而不需要考虑互斥访问的问题。多线程内存池有可能被多个线程共享，因此需要在每次分配和释放内存时加锁。相对而言，单线程内存池的性能更高，而多线程内存池的适用范围更广。

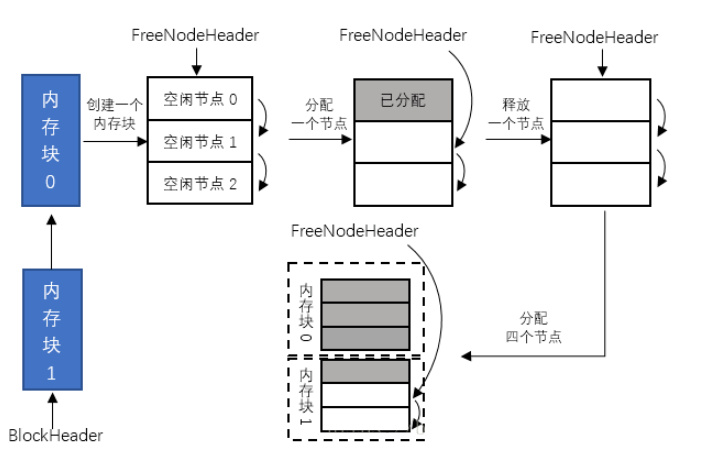
从内存池可分配大小来看，可以分为固定内存池和可变内存池。所谓固定内存池是指，应用程序每次从内存池中分配出来的内存单元大小已经事先确定，是固定不变的；而可变内存池则每次分配的内存单元大小可按序变化，应用范围更广，而性能比固定内存池要低。

# 经典的内存池技术

经典内存池技术，是一种用于分配大量大小相同的小对象的技术。通过该技术可以极大加速内存分配/释放过程。既然是针对特定对象的内存池，所以内存池一般设计为类模板，根据不同的对象来进行实例化。

## 经典内存池设计流程

1. 先申请一块连续的内存空间，该段内存空间能够容纳一定数量的对象。用一个链表维护内存块，每个内存块都维护一个指向下一内存块的指针。
2. 每个对象与一个指向下一对象的指针一起构成一个内存节点。一个内存块中可以分为多个内存节点。各个空闲的内存节点通过指针形成一个链表，链表的每个内存节点都是一块可分配的内存空间。
3. 某个内存节点一旦分配出去，就从空闲链表中删除。
4. 一旦释放了某个内存节点，就重新加到空闲表的表头。
5. 如果一个内存块中所有的节点分配完毕，则申请一个新的内存块，并让其作为块链表的表头，加入到块链表中。



**图：内存池原理实现**