动态存储分配器：动态存储分配器维护一个进程的虚拟存储区域，称为heap堆，对于每个进程，内核维护着
(读作break)，它指向堆的顶部
分配器将堆视为一组不同的大小的块的集合来维护，每个块就是一个连续的虚拟存储器片，要么是已分配的
分配器有两种基本风格：显式分配器：要求应用显式地释放任何已分配的块;隐式分配器：除了包含显式分
另一方面要求检测一个已分配块何时不再被程序所使用，那么释放这个块，隐式分配器也叫做垃圾收集器，
C语言程序默认使用的是显示分配器，程序通过malloc函数来从堆中分配内存，
动态存储器分配器 例如malloc 可以通过mmap和munmap函数，显式地分配和释放堆存储器，或者还可以使用
sbrk函数通过将内核brk指针增加incr来扩展和收缩堆，如果成功，他就返回brk的旧值，使用负值或者零有
分配器的要求和目标
1：处理任意请求的序列
2: 立即响应请求
3：只是用堆
4：对齐块
5：不修改已分配的块
最大化吞吐量
最大化存储器利用率
内部碎片：是在一个已分配块比有效载荷大时发生的，一个分配器的实现可能对已分配块强加一个最小的大
以满足对齐约束条件 内部碎片的量化就是已分配块的大小和它们的有效载荷大小只差的和，在任意时刻，内部
实现方式
外部碎片：就是空闲存储器合计起来足够满足一个分配请求，但是没有一个单独的空闲块足够大可以处理这
外部碎片的量化，取决于以前的请求的模式和分配器的实现方式，还取决于将来的请求的模式，外部碎片难以
启发式策略试图维持少量的大的空闲块，而不是大量的小的空闲块
一个分配器的设计：
空闲块的组织;
放置;
分割;
合并;
隐式空闲链表：
是因为空闲块是通过头部中的大小字段隐含地链接
放置已分配的块：
当一个程序请求一个K字节的块，分配器搜索空闲链表，查找一个足够大的可以放置所请求块的空闲区，一
下一次适配和最佳适配
分割空闲块：
一旦分配器找到一个匹配的空闲块，通常会选择将这个空闲块分割为两部分，第一部分变成分配块，而剩下
获得额外的堆存储空闲：
当分配器在空闲块中找不到合适的块时，则分配器就会通过调用sbrk函数，向内核请求额外的堆存储器，并
去，然后将请求放置在这个新的空闲块中
合并空闲块：
当分配器释放一个已分配块时，可能有其他的空闲块在与这个新释放的空闲块相邻，这些临界的空闲块可能
为了解决假碎片的问题，任何实际的分配器都必须合并相邻的空闲块，这个过程称为合并， 但是一个很重
显式空闲链表：
对于隐式空闲链表来说，块的分配与堆的总数呈线的关系，隐式空闲链表是不合适的。一种更好的方式就是
某种形式的显示数据结构，堆可以组织为一个双向空闲链表，每个空闲块中都包含一个前驱和一个后继指针，
适配的分配时间从块的总数的线性时间减少到了空闲块的数量的线性时间，不过释放一个块的时间可以是线性
常数，取决于我们所选择的空闲链表块的排序策略 一种方法就是后进先出顺序来维护链表，将释放的块放置在
另一种就是地址顺序来维护链表，其中链表中每个块的地址都小于它的后继地址，在这种情况下，释放一个块
的搜索来定位合适的前驱，按照地址排序首次适配比LIFO排序的首次适配由更高的存储器利用率，接近最佳适
内存泄露发生在程序运行的时候