free() 函数只传入一个内存地址，为什么能知道要释放多大的内存?

CPP开发者 2024年08月15日 11:50 浙江

The following article is from CppPlayer Author CppPlayer

free() 函数只传入一个内存地址，为什么能知道要释放多大的内存?

free()函数，要求传入的参数必须是malloc的返回值。在进行malloc函数申请内存时，操作系统实际会申请大于malloc要求的长度。

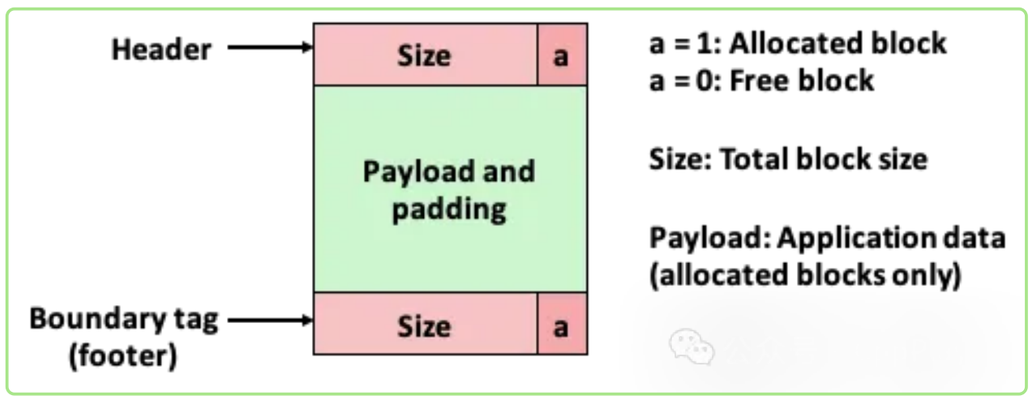
malloc分配的内存为一个个chunk，以下是一个典型的 malloc_chunk 结构定义（以 glibc 为例）：

```
struct malloc_chunk {
    size_t prev_size;  /* 前一个内存块的大小（如果合并的话） */
    size_t size;       /* 当前内存块的大小，包括边界标记 */
    struct malloc_chunk *fd;  /* 指向前一个空闲内存块的指针（用于空闲内存列表） */
    struct malloc_chunk *bk;  /* 指向下一个空闲内存块的指针（用于空闲内存列表） */
};
```

prev_size 字段表示前一个内存块的大小（如果当前内存块与前面的内存块合并在一起）。size 字段表示当前内存块的大小，包括边界标记。fd 和 bk 字段分别表示指向前后空闲内存块的指针，用于维护空闲内存列表。

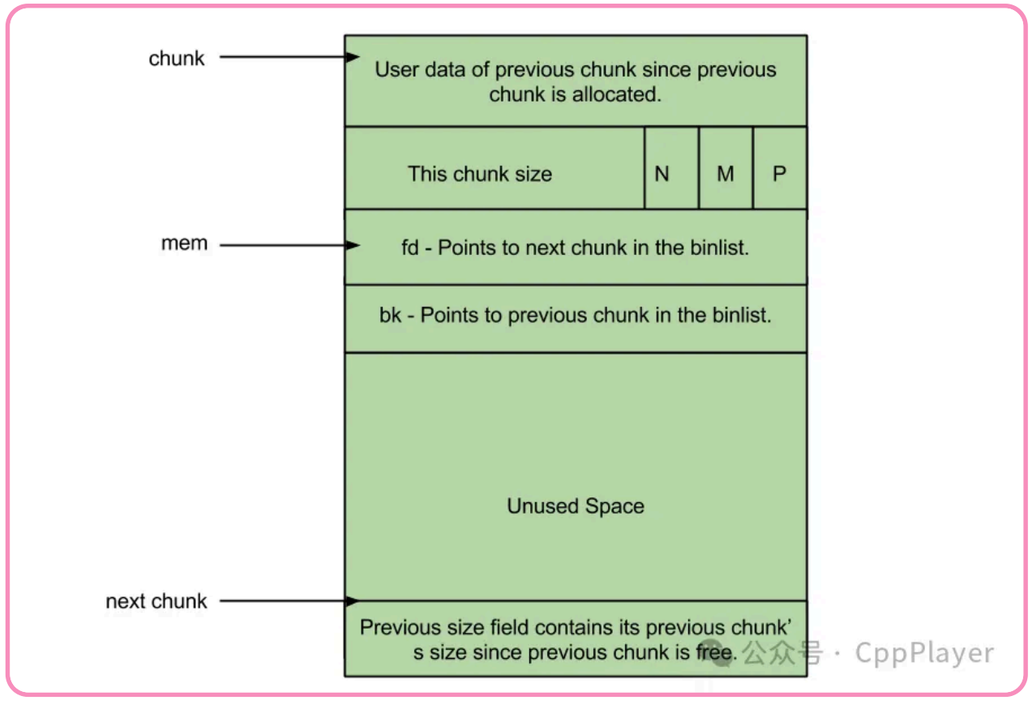
当释放一个内存块时，内存分配器可以检查 prev_size 字段以确定前一个内存块是否为空闲。如果前一个内存块为空闲，内存分配器可以将这两个相邻的空闲内存块合并成一个更大的空闲内存块。这样可以使内存分配更加高效，减少内存碎片

图解chunk内存块：



malloc返回的指针不是指向了Header，而是指向了Payload开始处。

所以空间的大小记录在参数指针指向地址的前面，free的时候通过这个记录即可知道要释放的内存有多大。



不同的内存分配器实现不一样，glibc的ptmalloc采用的是这种隐藏头风格，gemalloc采用其他的实现方式。

关于malloc还有几个知识点：

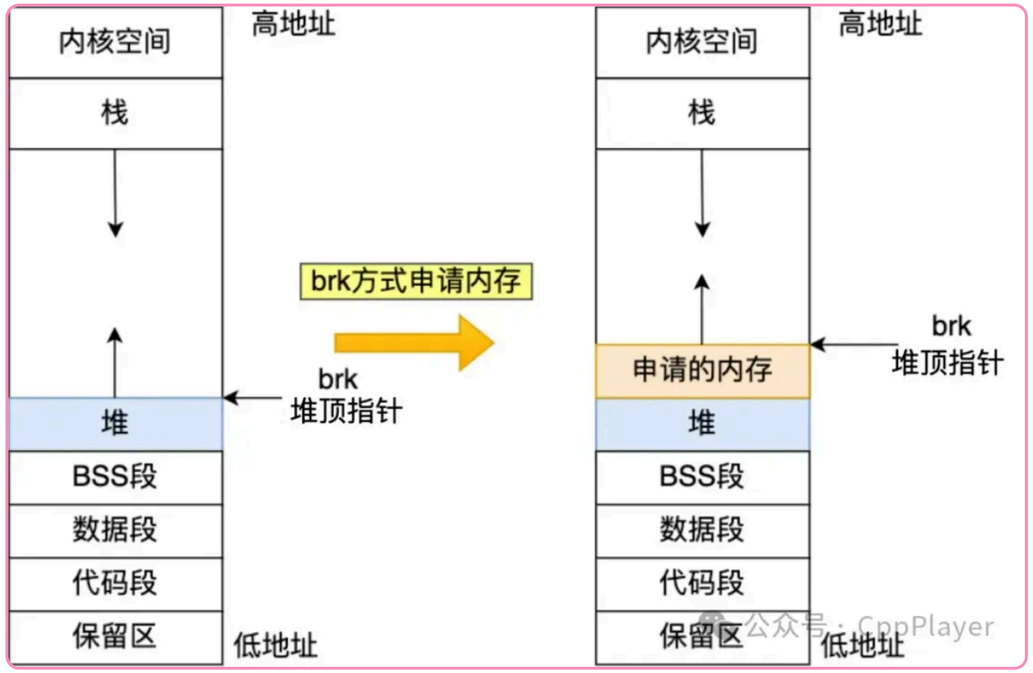
malloc是系统调用吗？

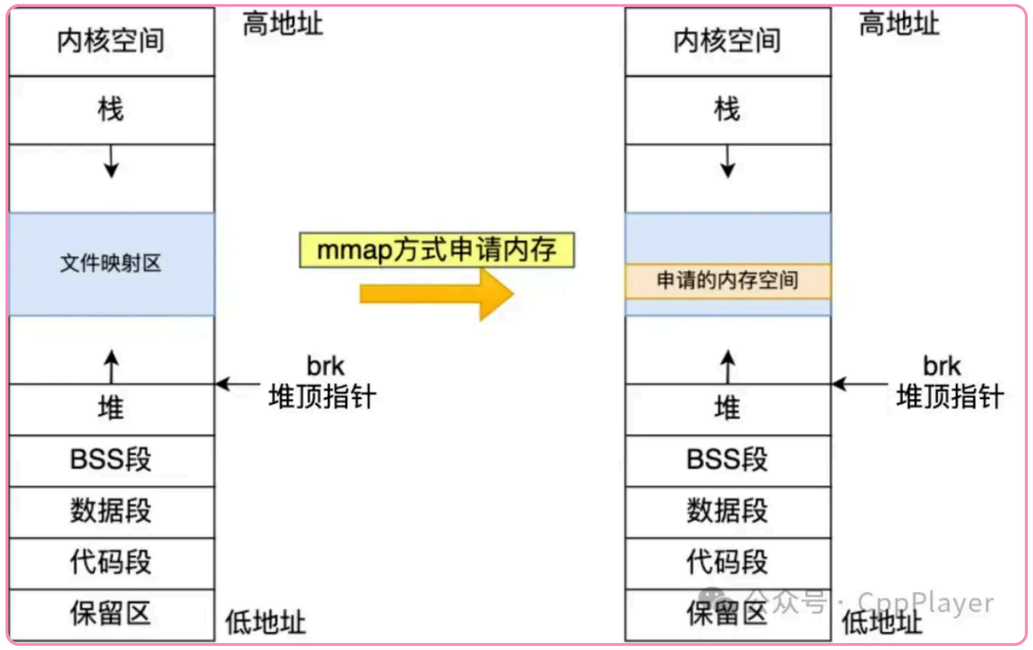
malloc是C语言库函数不是系统函数调用。

malloc 申请内存的时候，会有两种方式向操作系统申请堆内存。

方式一：通过 brk() 系统调用从堆分配内存

方式二：通过 mmap() 系统调用在文件映射区域分配内存；





malloc分配的是物理内存吗？

答案：不是物理内存，是虚拟内存。

现在的操作系统，内存管理通常是基于虚拟内存的，所以应用程序看到的内存地址（虚拟地址）与实际的物理内存地址（物理地址）是不同的。操作系统通过内存管理单元（MMU）来将虚拟地址转换为物理地址。

当应用程序首次访问这块内存时，操作系统发现对应的物理内存尚未分配，它会从可用的物理内存中分配相应的空间，并更新页表项以完成虚拟地址到物理地址的映射。

如果这块内存从来没有被访问，那么就不会分配实际的物理内存，节约了内存。

调用free后，内存会被操作系统立马回收吗？

答案：不会。

当程序员使用free函数释放内存后，这块内存并不会立即被归还给操作系统。相反，这些被释放的内存首先会被内存管理器（如ptmalloc）保存起来，以便后续重用。这样做的主要目的是减少与操作系统的内存交互次数，从而降低系统调用的开销。

内存管理器会使用双链表等方式来管理这些被释放的内存块，当程序再次申请内存时，内存管理器会首先尝试从这些已释放的内存块中找到合适的块返回给程序，而不是直接向操作系统请求新的内存。这种机制有助于减少内存碎片和提高内存使用效率。