

闲的。如果我们强加一个双字的对齐约束条件,那么块大小就总是8的倍数,且块大小的最低3位总是零。因此,我们只需要内存大小的29个高位,释放剩余的3位来编码其他信息。在这种情况下,我们用其中的最低位(已分配位)来指明这个块是已分配的还是空闲的。例如,假设我们有一个已分配的块,大小为24(0x18)字节。那么它的头部将是

```
0x00000018 | 0x1 = 0x00000019
```

类似地,一个块大小为40(0x28)字节的空闲块有如下的头部:

```
0x00000028 | 0x0 = 0x00000028
```

头部后面就是应用调用 malloc 时请求的有效载荷。有效载荷后面是一片不使用的填充块,其大小可以是任意的。需要填充有很多原因。比如,填充可能是分配器策略的一部分,用来对付外部碎片。或者也需要用它来满足对齐要求。

假设块的格式如图 9-35 所示,我们可以将堆组织为一个连续的已分配块和空闲块的序列,如图 9-36 所示。

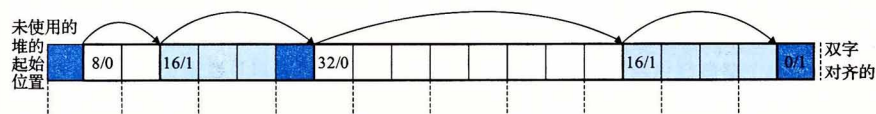



图 9-36 用隐式空闲链表来组织堆。阴影部分是已分配块。没有阴影的部分是空闲块。

头部标记为(大小(字节)/已分配位)

我们称这种结构为隐式空闲链表,是因为空闲块是通过头部中的大小字段隐含地连接着的。分配器可以通过遍历堆中所有的块,从而间接地遍历整个空闲块的集合。注意,我们需要某种特殊标记的结束块,在这个示例中,就是一个设置了已分配位而大小为零的终止头部(terminating header)。(就像我们将在 9.9.12 节中看到的,设置已分配位简化了空闲块的合并。)

隐式空闲链表的优点是简单。显著的缺点是任何操作的开销,例如放置分配的块,要求对空闲链表进行搜索,该搜索所需时间与堆中已分配块和空闲块的总数呈线性关系。

很重要的一点就是意识到系统对齐要求和分配器对块格式的选择会对分配器上的最小块大小有强制的要求。没有已分配块或者空闲块可以比这个最小值还小。例如,如果我们假设一个双字的对齐要求,那么每个块的大小都必须是双字(8 字节)的倍数。因此,图 9-35 中的块格式就导致最小的块大小为两个字:一个字作头,另一个字维持对齐要求。即使应用只请求一字节,分配器也仍然需要创建一个两字的块。

 **练习题 9.6** 确定下面 malloc 请求序列产生的块大小和头部值。假设:1)分配器保持双字对齐,并且使用块格式如图 9-35 中所示的隐式空闲链表。2)块大小向上舍入为最接近的 8 字节的倍数。

请求	块大小(十进制字节)	块头部(十六进制)
malloc(1)		
malloc(5)		
malloc(12)		
malloc(13)		

9.9.7 放置已分配的块

当一个应用请求一个 k 字节的块时,分配器搜索空闲链表,查找一个足够大可以放置