C++ 这样的语言的收集器通常不能维持可达图的精确表示。这样的收集器也叫做保守的垃圾收集器(conservative garbage collector)。从某种意义上来说它们是保守的，即每个可达块都被正确地标记为可达了，而一些不可达节点却可能被错误地标记为可达。

收集器可以按需提供它们的服务，或者它们可以作为一个和应用并行的独立线程，不断地更新可达图和回收垃圾。例如，考虑如何将一个 C 程序的保守的收集器加入到已存在的 malloc 包中，如图 9-50 所示。

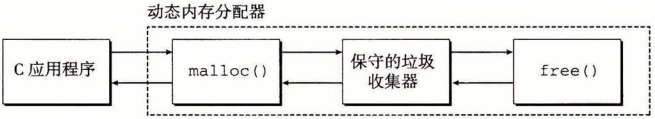


图 9-50　将一个保守的垃圾收集器加入到 C 的 malloc 包中

无论何时需要堆空间时，应用都会用通常的方式调用 malloc。如果 malloc 找不到一个合适的空闲块，那么它就调用垃圾收集器，希望能够回收一些垃圾到空闲链表。收集器识别出垃圾块，并通过调用 free 函数将它们返回给堆。关键的思想是收集器代替应用去调用 free。当对收集器的调用返回时，malloc 重试，试图发现一个合适的空闲块。如果还是失败了，那么它就会向操作系统要求额外的内存。最后，malloc 返回一个指向请求块的指针(如果成功)或者返回一个空指针(如果不成功)。

### 9.10.2　Mark & Sweep 垃圾收集器

Mark&Sweep 垃圾收集器由标记(mark)阶段和清除(sweep)阶段组成，标记阶段标记出根节点的所有可达的和已分配的后继，而后面的清除阶段释放每个未被标记的已分配块。块头部中空闲的低位中的一位通常用来表示这个块是否被标记了。

我们对 Mark&Sweep 的描述将假设使用下列函数，其中 ptr 定义为 typedef void *ptr：

- ptr isPtr (ptr p)。如果 p 指向一个已分配块中的某个字，那么就返回一个指向这个块的起始位置的指针 b。否则返回 NULL。
- int blockMarked(ptr b)。如果块 b 是已标记的，那么就返回 true。
- int blockAllocated(ptr b)。如果块 b 是已分配的，那么就返回 true。
- void markBlock(ptr b)。标记块 b。
- int length(b)。返回块 b 的以字为单位的长度(不包括头部)。
- void unmarkBlock(ptr b)。将块 b 的状态由已标记的改为未标记的。
- ptr nextBlock(ptr b)。返回堆中块 b 的后继。

标记阶段为每个根节点调用一次图 9-51a 所示的 mark 函数。如果 p 不指向一个已分配并且未标记的堆块，mark 函数就立即返回。否则，它就标记这个块，并对块中的每个字递归地调用它自己。每次对 mark 函数的调用都标记某个根节点的所有未标记并且可达的后继节点。在标记阶段的末尾，任何未标记的已分配块都被认定为是不可达的，是垃圾，可以在清除阶段回收。

清除阶段是对图 9-51b 所示的 sweep 函数的一次调用。sweep 函数在堆中每个块上反复循环，释放它所遇到的所有未标记的已分配块(也就是垃圾)。

图 9-52 展示了一个小堆的 Mark&Sweep 的图形化解释。块边界用粗线条表示。每个方块对应于内存中的一个字。每个块有一个字的头部，要么是已标记的，要么是未标记的。