哈爾濱工業大學

实验报告

实 验（八）

题　　目 Dynamic Storage Allocator

动态内存分配器

专　　业 计算机类

学　　号 1170300817

班　　级 1703008

学 生 姓 名 林之浩

指 导 教 师 郑贵滨

实 验 地 点 G712

实 验 日 期 12.10

计算机科学与技术学院

目 录

第 1 章 实验基本信息

1.1 实验目的

理解现代计算机系统虚拟存储的基本知识

掌握 C 语言指针相关的基本操作

深入理解动态存储申请、释放的基本原理和相关系统函数

用 C 语言实现动态存储分配器，并进行测试分析

培养 Linux 下的软件系统开发与测试能力

1.2 实验环境与工具

1.2.1 硬件环境

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

1.2.2 软件环境

Windows7 64 位以上；VirtualBox/Vmware 11 以上；Ubuntu 16.04 LTS 64 位/优麒麟 64 位

1.2.3 开发工具

gcc，coldblocks

1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT 或 PDF）

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。

熟知 C 语言指针的概念、原理和使用方法

了解虚拟存储的基本原理

熟知动态内存申请、释放的方法和相关函数

熟知动态内存申请的内部实现机制：分配算法、释放合并算法等

第 2 章 实验预习

总分 20 分

2.1 动态内存分配器的基本原理（5 分）

动态内存分配器维护着一个进程的虚拟内存区域，称为堆。

分配器将堆视为一组不同大小的块的集合来维护。每个块就是一个连续的虚拟内存片，要么是已分配的，要么是空闲的。已分配的块显式地保留为供应用程序使用。空闲块可以用来分配。空闲块保持空闲，直到它显式地被应用所分配。一个已分配的块保持已分配的状态，直到它被释放。

2.2 带边界标签的隐式空闲链表分配器原理（5 分）

在每个块的结尾处添加一个脚部，其中脚部就是头部的一个副本。如果每个块包括这样一个脚部，那么分配器就可以通过检查上一个块的脚部判断前一个块的初始状态和位置。

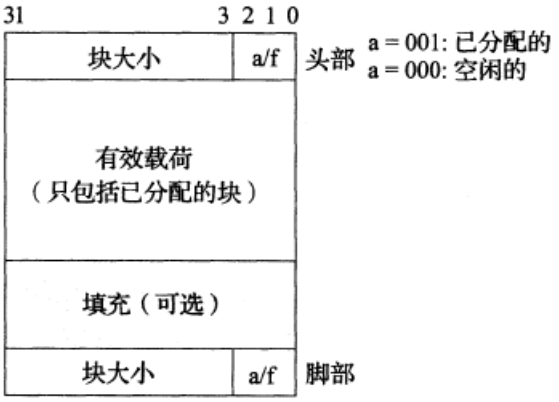

图 9-39 使用边界标记的堆块的格式

2.3 显示空间链表的基本原理（5 分）

succ(后继)指针，如图 9-48 所示。

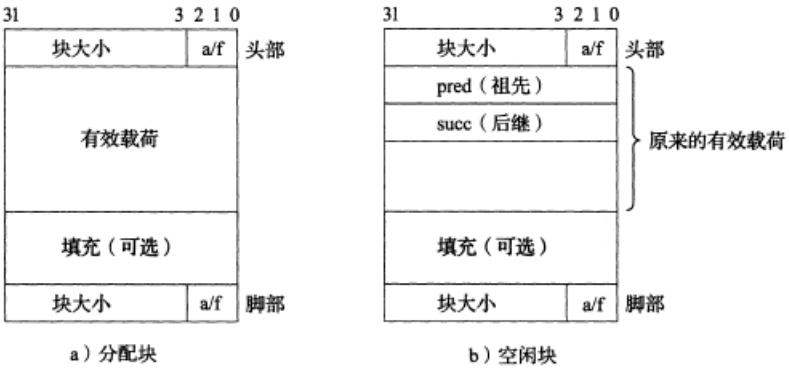

图 9-48　使用双向空闲链表的堆块的格式

显式空闲链表是将空闲块组织为某种形式的显式数据结构，比如链表。因为程序不需要一个空闲块的主体，所以实现这个数据结构的指针可以存放在这些空闲块的主体里面。如，堆可以组织成一个双向链表，在每个空闲块中，都包含一个前驱与一个后继指针。

维护链表的顺序有：后进先出（LIFO），将新释放的块放置在链表的开始处，使用 LIFO 的顺序和首次适配的放置策略，分配器会最先检查最近使用过的块，在这种情况下，释放一个块可以在线性的时间内完成，如果使用了边界标记，那么合并也可以在常数时间内完成。按照地址顺序来维护链表，其中链表中的每个块的地址都小于它的后继的地址，在这种情况下，释放一个块需要线性时间的搜索来定位合适的前驱。平衡点在于，按照地址排序首次适配比 LIFO 排序的首次适配有着更高的内存利用率，接近最佳适配的利用率。

2.4 红黑树的结构、查找、更新算法（5 分）

红黑树（RBTree）的定义如下:

1.任何一个节点都有颜色，黑色或者红色；

2.根节点是黑色的；

3.父子节点之间不能出现两个连续的红节点；

4.任何一个节点向下遍历到其子孙的叶子节点，所经过的黑节点个数须相等；

5.空节点被认为是黑色的

基本数据结构定义：

```
#define RED             0       // 红色节点
#define BLACK       1       // 黑色节点
```

```
typedef int Type;

// 红黑树的节点
typedef struct RBTreeNode{
    unsigned char color;            // 颜色(RED 或 BLACK)
    Type     key;                       // 关键字(键值)
    struct RBTreeNode *left;       // 左孩子
    struct RBTreeNode *right;      // 右孩子
    struct RBTreeNode *parent;     // 父结点
}Node, *RBTree;

// 红黑树的根
typedef struct rb_root{
    Node *node;
}RBRoot;
```

查找算法：

类似于 AVL 树，从树根节点与关键字进行比较，寻找左右孩子，逐层向下寻找，直到找到对应的值，若无，返回 NULL。

插入操作：

1．将红黑树当作一颗二叉查找树，将节点插入。

2. 将插入的节点着色为"红色"。

3. 通过一系列的旋转或着色等操作，使之重新成为一颗红黑树。

第 3 章 分配器的设计与实现

总分 50 分

3.1 总体设计（10 分）

介绍堆、堆中内存块的组织结构，采用的空闲块、分配块链表/树结构和相应算法等内容。

利用书本 9.9.14 伙介绍的内容。

采用分离的空闲链表，维护多个空闲链表，将所有可能的大小分成一些大小类，根据 2 的幂划分块大小。

{1}{2},{3，4}{5-8}{9-16}----{1025-2048}{2049-4096}。。。。。。

当分配器需要一个大小为 n 的块时，就搜索相应的空闲链表。首次匹配，查找到了，就可选地分割它，将剩余部分插入到适当的空闲链表中，没有合适的，就申请新的堆内存。从新的堆中分配一块，把剩余的块放在合适的链表里。

已分配的 Block 结构：

	31-3	2	1	0
头部	块大小			A
	负载			
尾部	块大小			A

未分配的 Block 结构

	31-3	2	1	0
头部	块大小			A
	指向链表前块的指针			
	指向链表后块的指针			
	负载			
尾部	块大小			A

因为链表是按照空间递增顺序维护的，所以采用的是首次适配的方法。

3.2 关键函数设计（40 分）

3.2.1 int mm_init(void)函数（5 分）

函数功能：初始化堆

处理流程：定义并初始化指向堆的头部的指针，初始化分离的空闲链表数组，定义对齐用的填充字，序言块和结尾块。最后调用 extend_heap 把初始化的堆扩展为指定的初始大小。返回 0。

要点分析：
1. 注意 mem_sbrk 的返回值如果是 NULL 则说明初始化失败了。
2. 检查 extend_heap 的返回值，如果是 NULL 则说明扩展堆失败了。
3. 序言块是 8 字节已分配块，只由一个头部和一个尾部组成。

3.2.2 void mm_free(void *ptr)函数（5 分）

函数功能：释放一个块。

参　　数：void *ptr 要释放的块的块指针。

处理流程：
1. 读取要释放的块的大小。
2. 把这个块的 realloc_tag 置 0。
3. 把这个块的头部和尾部都设置为未分配。
4. 把这个块插入空闲链表。
5. 合并可能的空闲块

要点分析：
1. 在释放当前块的时候一定要把头部和尾部都设置为未分配。
2. 在插入对应的空闲链表之后要调用 coalesce 函数

3.2.3 void *mm_realloc(void *ptr, size_t size)函数（5 分）

函数功能：将 ptr 指针指向的块扩展为 size 大小。

参　　数：void *ptr 要变更的 block　size_t size 变更后的大小

处理流程：
1. size 小于原来的值的话不用变动。
2. 扩展的情况：如果后块是结束块，就按缺少的大小申请新的堆空间，在链表中删除原块，设置新块
3. 如果后一个块是空闲块，判断合并后是否满足要求，
4. 如果满足，删除后块，更改当前块的大小

5. 不满足，只能 malloc 一个新的块，free 掉当前的
6. 如果不是，也是执行 5.申请新的块。

要点分析：
1. 要将 size 对齐，如果 size<8，手动对齐，否则调用 ALIGN 函数对齐。
2. 如果 size 小于原来的值的话不用变动，等于 0 直接返回。
3. 注意后一个块是空闲块的时候要先判断是否能够满足大小条件。

3.2.4 int mm_check(void)函数（5 分）

函数功能：堆的一致性检查

处理流程：
1. 扫描空闲链表数组里的每一个链表，查看是否有空闲块没有合并。确认空闲链表都指向有效的空闲块。
2. 检查每个块是否已经被占用，确认是否链表的每个块都标注为 free。
3. 遍历堆内所有的空闲块，如果数量相等，则说明每个空闲块都在空闲链表。
4. 如果堆遍历成功，则说明每个堆块都指向有效的地址。

要点分析：
1. 直接提取每个块的 tag 确认是否是 free 状态。
2. 提取前后块的 tag 查看是否出现未合并现象。
3. 用变量 count_1,count_2 分别记录链表和堆中的快数，相同说明每个空闲块都在空闲链表。
4. 检查空闲链表是否能成功遍历，能成功遍历说明有效。

3.2.5 void *mm_malloc(size_t size)函数（10 分）

函数功能：分配内存块。

参　　数：size_t size 所要求的字节数。

处理流程：
1. 计算最小块大小。
2. 计算需要在哪个链表搜索块
3. 在这个空闲链表中搜索符合大小要求的空闲块。
4. 如果没有符合要求的，计算如果需要扩展堆，扩展的大小。（双字对齐）
5. 如果有，返回块指针。

要点分析：

1. 要将 size 对齐，如果 size<8，手动对齐，否则调用 ALIGN 函数对齐。
2. 计算需要在哪个链表搜索块其实是对 searchsize 反复除以 2，因为链表都是 2 的 k 次幂大小类，除几次说明在第几个
3. 放置函数 place 的分割策略要看剩余的长度，如果剩余大小不超过 16B，则最多只能放下 free block 的附加信息，则不切分。

3.2.6 static void *coalesce(void *bp)函数（10 分）

函数功能：合并空闲块

处理流程：
1. 记录前后块是否已分配，记录当前块的大小
2. 分四种情况处理前后是否为空闲块（依照书上的四种情况）
3. 前后都已分配直接返回
4. 前块已分配后块未分配，则在空闲链表中删去当前节点和下一个块的节点，合并两个块成一个，把新块加入空闲链表。
5. 前块未分配后块已分配，和上一条同理。
6. 前后块都没有分配，删去三个节点，合并三个块，把新块加入空闲链表。

要点分析：
1. 如果前块的 realloc_tag 是 1，则把前块视为已分配，不进行合并
2. 合并块的过程要先计算新的 size 值，然后改变头部和尾部，最后把新的空闲块加入链表。

第 4 章测试

总分 10 分

4.1 测试方法

Make
以后输入
./mdriver -t traces/ -v
测试

4.2 测试结果评价

显式分离链表+维护大小递增+首次适配算法已经可以达到较好的效果。
分析 trace 文件，特别是后面两个 trace 做出针对性的优化就能有效提高利用率。

4.3 自测试结果

```
linleo@ubuntu:~/Desktop/sharefile/malloclab-handout$ make
gcc -Wall -O2 -m32   -c -o mm.o mm.c
gcc -Wall -O2 -m32 -o mdriver mdriver.o mm.o memlib.o fsecs.o fcyc.o clock.o ftime
r.o
linleo@ubuntu:~/Desktop/sharefile/malloclab-handout$ ./mdriver -t traces/ -v
Team Name:HIT IS MAD
Member 1 :林之浩1170300817:630073498@qq.com
Using default tracefiles in traces/
Measuring performance with gettimeofday().

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   99%    5694  0.000290 19662
 1       yes   99%    5848  0.000479 12214
 2       yes   97%    6648  0.000344 19348
 3       yes   99%    5380  0.000281 19125
 4       yes   99%   14400  0.000531 27098
 5       yes   93%    4800  0.000399 12024
 6       yes   92%    4800  0.000458 10490
 7       yes   81%   12000  0.000472 25402
 8       yes   88%   24000  0.000745 32210
 9       yes  100%   14401  0.000249 57859
10       yes   98%   14401  0.000142101344
Total          95%  112372  0.004390 25597

Perf index = 57 (util) + 40 (thru) = 97/100
linleo@ubuntu:~/Desktop/sharefile/malloclab-handout$
```

第 5 章 总结

5.1 请总结本次实验的收获

掌握了动态内存分配器的设计原理，深入理解了显式分离空闲链表和分离适配算法的原理，认识到了显式空闲链表的优越性。

5.2 请给出对本次实验内容的建议

临近期末，学习安排紧张，实验可以适当降低 na 难度。

注：本章为酌情加分项。

参考文献

为完成本次实验你翻阅的书籍与网站等

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A 集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/collection/anatmorp.