CSAPP

实 验 报 告

学生姓名 吴语港

学生学号 SA19225404

实验日期 2019/11/5

实　验　报　告

一、实验名称：Malloc lab

二、实验学时： 3

三、实验内容和目的：

目的

在该实验中，需要用 C 语言实现一个动态存储分配器（dynamic storage allocater）。需要实现 malloc、free、realloc 等功能。当然不仅要正确的实现相关功能也要满足速度效率等要求。

步骤

tar xvf malloclab-handout.tar 解压文件

我们需要修改的唯一文件是 mm.c，包含如下几个需要实现的函数

int mm_init(void);

void *mm_malloc(size_t size);

void mm_free(void *ptr);

void *mm_realloc(void *ptr, size_t size)

解释

mm_init:在调用 mm_malloc，mm_realloc 或 mm_free 之前，调用 mm_init 进行初始化，正确返回 0。

mm_malloc：在堆区域分配指定大小的块，分配的空间，返回的指针应该是 8 字节对齐的

mm_free:释放指针指向的 block

mm_realloc:返回指向一个大小为 size 的区域指针，满足一下条件：

if ptr is NULL, the call is equivalent to mm_malloc(size);

if size is equal to zero, the call is equivalent to mm_free(ptr);

if ptr is not NULL：先按照 size 指定的大小分配空间，将原有数据从头到尾拷贝到新分配的内存区域，而后释放原来 ptr 所指内存区域

可以调用的函数

void *mem_sbrk(int incr): Expands the heap by incr bytes, where incr is apositive non-zero integer and returns a generic pointer to the first byte of the newly allocated heap area.

„ void *mem_heap_lo(void):Returns a generic pointer to the first byte in the heap.

„ void *mem_heap_hi(void): Returns a generic pointer to the last byte in the heap.

„ size_t mem_heapsize(void):Returns the current size of the heap in bytes.

„ size_t mem_pagesize(void): Returns the system’s page size in bytes (4K onLinux systems).

验证方法

mdriver.c：负责测试 mm.c 的正确性,空间利用率和吞吐量

-f <tracefile>: -f 后添加一些 trace file 来测试我们实现的函数

-V:打印出诊断信息

./mdriver -V -f short1-bal.rep

编程规则

不能改变 mm.c 中函数接口

不能直接调用任何内存管理的库函数和系统函数 malloc, calloc, free, realloc, sbrk, brk

不能定义任何全局或者静态复合数据结构如 arrays, structs, trees，允许使用 integers, floats, and pointers 等简单数据类型

只要提交 mm.c 文件

四、实验步骤及结果：

mm_init 函数

空闲块的组织方法-Segregated free list 方法
segregated free list 中大小类的分类方法如下，并且将该 list 表放在 heap 的头部，通过序言块将它与数据隔离。在每一个大小类中，空闲块按照 size 由大到小排序。

```
/*
* mm_init - initialize the malloc package.
* The return value should be -1 if there was a problem in performing the initialization, 0
otherwise
*/
int mm_init(void)
{
if((heap_listp = mem_sbrk(14*WSIZE))==(void *)-1) return -1;

PUT(heap_listp,0); /*block size list<=8*/
PUT(heap_listp+(1*WSIZE),0); /*block size list<=16*/
PUT(heap_listp+(2*WSIZE),0); /*block size list<=32*/
PUT(heap_listp+(3*WSIZE),0); /*block size list<=64*/
PUT(heap_listp+(4*WSIZE),0); /*block size list<=128*/
PUT(heap_listp+(5*WSIZE),0); /*block size list<=256*/
PUT(heap_listp+(6*WSIZE),0); /*block size list<=512*/
PUT(heap_listp+(7*WSIZE),0); /*block size list<=2048*/
PUT(heap_listp+(8*WSIZE),0); /*block size list<=4096*/
PUT(heap_listp+(9*WSIZE),0); /*block size list>4096*/
PUT(heap_listp+(10*WSIZE),0);
PUT(heap_listp+(11*WSIZE),PACK(DSIZE,1));
PUT(heap_listp+(12*WSIZE),PACK(DSIZE,1));
PUT(heap_listp+(13*WSIZE),PACK(0,1));
```

```
block_list_start = heap_listp;
heap_listp += (12*WSIZE);
if((extend_heap(CHUNKSIZE/DSIZE))==NULL) return -1;
#ifdef DEBUG
mm_check(__FUNCTION__);
#endif // DEBUG
return 0;
```

```
}
/*最小 Block4 字（16 字节）*/
static void *extend_heap(size_t dwords)
{
char *bp;
size_t size;
size = (dwords % 2) ? (dwords+1) * DSIZE : dwords * DSIZE;
if((long)(bp = mem_sbrk(size))==(void *)-1)
return NULL;
PUT(HDRP(bp),PACK(size,0));
PUT(FTRP(bp),PACK(size,0));
PUT(NEXT_LINKNODE_RP(bp),NULL);
PUT(PREV_LINKNODE_RP(bp),NULL);
PUT(HDRP(NEXT_BLKP(bp)),PACK(0,1));
return coalesce(bp);
}
```

空闲块查找方法 - best fit

因为同一大小类中空闲块由小到大排序，所以查找是第一个合适的就是最适配的

mm_malloc 函数

```
/*
* mm_malloc - Allocate a block by incrementing the brk pointer.
* Always allocate a block whose size is a multiple of the alignment.
*/
void *mm_malloc(size_t size)
{
size_t asize;
size_t extendsize;
char *bp;
if(size ==0) return NULL;

if(size <= DSIZE)
{
asize = 2*(DSIZE);
```

```
}
else
{
asize = (DSIZE)*((size+(DSIZE)+(DSIZE-1)) / (DSIZE));
}
if((bp = find_fit(asize))!= NULL)
{
place(bp,asize);
#ifdef DEBUG
mm_check(__FUNCTION__);
#endif // DEBUG
return bp;
}
/*apply new block*/
extendsize = MAX(asize,CHUNKSIZE);
if((bp = extend_heap(extendsize/DSIZE))==NULL)
{
return NULL;
}
place(bp,asize);
#ifdef DEBUG
mm_check(__FUNCTION__);
#endif // DEBUG
return bp;
}
```

mm_free 函数

```
/*
* mm_free - Freeing a block does nothing.
*/
void mm_free(void *bp)
{
```

```
if(bp == 0)
return;
size_t size = GET_SIZE(HDRP(bp));
PUT(HDRP(bp), PACK(size, 0));
PUT(FTRP(bp), PACK(size, 0));
PUT(NEXT_LINKNODE_RP(bp),NULL);
PUT(PREV_LINKNODE_RP(bp),NULL);
coalesce(bp);
#ifdef DEBUG
mm_check(__FUNCTION__);
#endif // DEBUG
}
```

mm_realloc 改进

对于请求的 newsize>原始的 oldsize 这种情况，我们将运用类似 coalesce 中的方法，先去检查前后是否有空闲块，并是否满足前后空闲块和当前已分配的空闲块 size 相加大于 newsize，如果是则合并，不需要再重新请求空闲块。如果不行，则需要重新 mm_malloc 一块新的空间

mm_realloc 函数：

```
/*
* mm_realloc - Implemented simply in terms of mm_malloc and mm_free
*/
void *mm_realloc(void *ptr, size_t size)
{
size_t oldsize = GET_SIZE(HDRP(ptr));
void *newptr;
size_t asize;

if(size == 0){
mm_free(ptr);
return 0;
}
```

```
if(ptr == NULL) return mm_malloc(size);
```

```
if(size <= DSIZE){
asize = 2*(DSIZE);
}else{
asize = (DSIZE)*((size+(DSIZE)+(DSIZE-1)) / (DSIZE));
}
```

```
if(oldsize==asize) return ptr;
```

```
if(oldsize<asize)
{
int isnextFree;
char *bp = realloc_coalesce(ptr,asize,&isnextFree);
if( isnextFree==1){
realloc_place(bp,asize);
return bp;
} else if(isnextFree ==0 && bp != ptr){
memcpy(bp, ptr, size);
realloc_place(bp,asize);
return bp;
}
else
{
newptr = mm_malloc(size);
memcpy(newptr, ptr, size);
mm_free(ptr);
return newptr;
}
}
else
{
realloc_place(ptr,asize);
return ptr;
}
}
```

实验结果：

测试用例 1

```
(base) wyg@wyg-virtual-machine:~/下载/CSAPP-Labs-master/yzf-malloclab-handout$ .
/mdriver -V  -f short1-bal.rep
Team Name:SA19225404
Member 1 :吴语港:ygwu@mail.ustc.edu.cn
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: short1-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   66%      12  0.000002  6000
Total          66%      12  0.000002  6000

Perf index = 40 (util) + 40 (thru) = 80/100
```

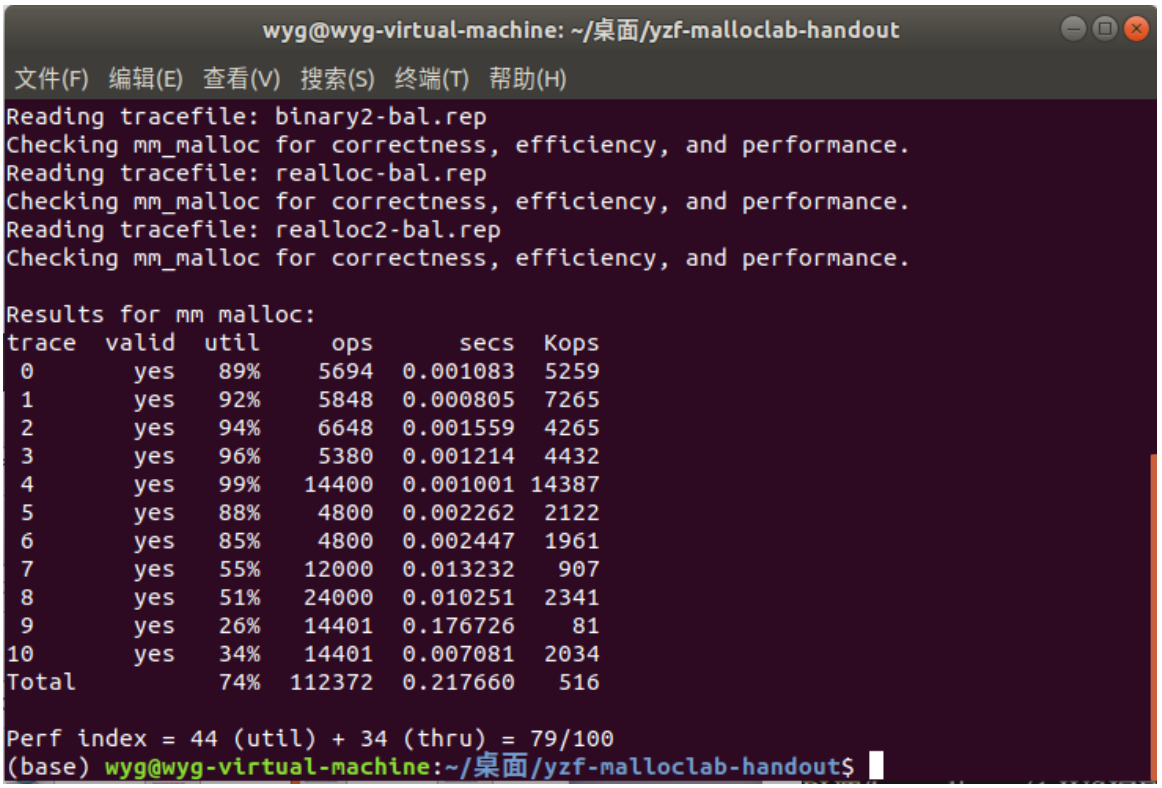
80 分

测试用例 2

```
(base) wyg@wyg-virtual-machine:~/下载/CSAPP-Labs-master/yzf-malloclab-handout$ .
/mdriver -V  -f short2-bal.rep
Team Name:SA19225404
Member 1 :吴语港:ygwu@mail.ustc.edu.cn
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: short2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   89%      12  0.000002  5217
Total          89%      12  0.000002  5217

Perf index = 54 (util) + 40 (thru) = 94/100
```

94 分

综合测试


wyg@wyg-virtual-machine: ~/桌面/yzf-malloclab-handout
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Reading tracefile: binary2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Results for mm malloc:
trace valid util ops secs Kops
0 yes 89% 5694 0.001083 5259
1 yes 92% 5848 0.000805 7265
2 yes 94% 6648 0.001559 4265
3 yes 96% 5380 0.001214 4432
4 yes 99% 14400 0.001001 14387
5 yes 88% 4800 0.002262 2122
6 yes 85% 4800 0.002447 1961
7 yes 55% 12000 0.013232 907
8 yes 51% 24000 0.010251 2341
9 yes 26% 14401 0.176726 81
10 yes 34% 14401 0.007081 2034
Total 74% 112372 0.217660 516
Perf index = 44 (util) + 34 (thru) = 79/100
(base) wyg@wyg-virtual-machine:~/桌面/yzf-malloclab-handout$

79 分