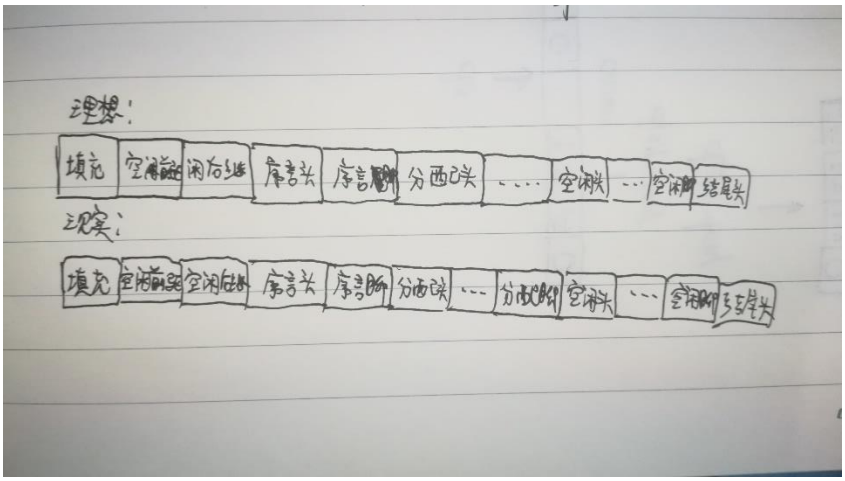
操作系统实验 3

动态内存分配器的实现

姓名：王嵘晟　学号：PB17111614

1. 主要步骤

(1) 仔细翻阅《Computer Systems A Programmer’s Perspective》这本书，学习关于内存分配的相关知识以及隐式空闲链表实现动态内存分配器的方法，进而自己写代码做到用显示空闲链表完成动态内存分配器。（这也导致了没有仔细看实验 PDF，没有按照要求，给分配块分配了脚部……）



(2) 额外声明两个指针作为空闲块的前驱和后继指针，一个全局指针 free_list_head 来指示空闲链表的表头，在隐式空闲链表代码的基础上修改 mm_init extend_heap coalesce mm_malloc mm_free mm_realloc 函数。

```
52
53  #define PRED(bp) ((char*)(bp))              //前驱指针
54  #define SUCC(bp) ((char*)(bp)+WSIZE)        //后继指针
55
```

空闲块的前驱指针和后继指针是全局变量

```
static char *free_list_head = NULL;
```

空闲链表头指针

```
int mm_init(void)
{
    if((heap_listp = mem_sbrk(6*WSIZE))==(void *)-1)
        return -1;
    PUT(heap_listp,0);                          // Alignment padding
    PUT(heap_listp+(1*WSIZE),0);                // Free PREV LINKNODE
    PUT(heap_listp+(2*WSIZE),0);                // Free NEXT LINKNODE
    PUT(heap_listp+(3*WSIZE),PACK(DSIZE,1));    // Prologue header
    PUT(heap_listp+(4*WSIZE),PACK(DSIZE,1));    // Prologue footer
    PUT(heap_listp+(5*WSIZE),PACK(0,1));        // Epilogue header
    free_list_head=heap_listp+WSIZE;
    heap_listp+=(4*WSIZE);
    if (extend_heap(CHUNKSIZE/WSIZE) == NULL)
        return -1;
    return 0;
}
```

为了双字对齐分配了 6 个单字大小，分别是填充块，前驱后继指针，序言块的头和脚，结尾块的头

(3) 额外写了四个函数：insert_to_emptylist，把空闲块插入到空闲链表的头部，fix_linklist 修复链表，当有空闲块被插入链表或者有块被分配掉移出链表时，修复链表。find_fit 通过首次适配寻找大小合适的块来分配，place 函数时分配函数，把空闲块分配。

(4) 使用 LIFO 策略分配空闲块，使得碎片尽可能小（少）。

2. 运行结果截图

```
david@ubuntu: ~/oslab/lab3/lab3-code/lab-3-stu
led by default]
         tmpP=GET(NEXT_LINKNODE_RP(tmpP));
              ^
gcc -g -Wall -O2 -m32 -o mdriver mdriver.o mm.o memlib.o fsecs.o fcyc.o clock.o
ftimer.o
david@ubuntu:~/oslab/lab3/lab3-code/lab-3-stu$ ./mdriver -v
Using default tracefiles in ./traces/
Measuring performance with gettimeofday().

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   89%    5694  0.000229 24821
 1       yes   55%   12000  0.004983  2408
 2       yes   51%   24000  0.004067  5901
 3       yes   92%    5848  0.000155 37656
 4       yes   66%   14400  0.000177 81310
 5       yes   94%    6648  0.000509 13071
 6       yes   96%    5380  0.000294 18318
 7       yes   88%    4800  0.000558  8604
 8       yes   85%    4800  0.000574  8368
Total          80%   83570  0.011545  7238

Perf index = 4.78 (util) + 4.00 (thru) = 8.8/10
david@ubuntu:~/oslab/lab3/lab3-code/lab-3-stu$
```

最终评分 8.8

3. 实验过程中遇到的技术问题和解决方法

(1) 由于没有仔细阅读实验 PDF 要求，写代码时保留了分配块的脚部，尝试了修改代码解决问题，但是由于时间有限并且有考试要复习，最终选择放弃。

(2) 尝试优化时适当百度了一下，学习到了 inline 这个函数声明，可以把在程序中反复调用的函数声明为内联函数，可以优化调用此函数所需要的时间。因此对 coalesce insert_to_emptylist 和 fix_linklist 这三个函数做了优化。

```
static void *extend_heap(size_t words);
inline void *coalesce(void *bp);
static void *find_fit(size_t size);
static void place(void *bp,size_t size);
inline void insert_to_Emptylist(char *p);
inline void fix_linklist(char *p);
```

(3) 在最后想尽一切办法提高评分时，把 extend_heap 的参数由 CHUNKSIZE/WSIZE 变为了 CHUNKSIZE/DSIZE 成功使评分有了一定的提高，原因不明，询问助教。

答：样例有关

4. 实验总结

本次试验作为第三次操作系统实验，在学习完内存管理后通过动态内存分配器实验对内存管理有了更深入的理解。

反思：今后实验一定仔细看实验要求……

对于指针的使用一直是 C 语言的难点，本次试验再次锻炼了对于指针的使用。对于内存分配策略的首次适配也同样有了更深入的理解。