malloclab实验报告

------2020200982 闫世杰

malloclab:实现段式的动态内存分配

采用分离适配的策略,将内存块分为15类,使用15个链表进行管理(free\_heap\_list[15]),内存大小从24B(header(4B)+footer(4B)+pre\_ptr(8B)+suc\_ptr(8B))开始,逐级递增为上一级的2倍.

管理策略:

1)对于每次新来的空闲块,简单的插入到对应的链表的队头.

2)对于每次分配请求,在对应链表中采用best\_fit进行内存分配

**具体实现:**

**宏定义:**

打包一些基本且常用的基础函数

**接口函数(基本与csapp相同,但多一些对链表的维护操作):**

mm\_init:为堆进行初始化

分配序言块,设置结尾,初始化15个空闲内存管理链表,然后扩展堆,进行第一次的内存分配并插入合适的free\_heap\_list[i]中

mm\_malloc(size):分配size大小的空间

根据size的大小选择合适的free\_heap\_list[i]进行检查,采用best\_fit的策略进行寻找合适的空闲块

1. 在该大小类中找到,进行分配,并将剩余的内存插入到合适的free\_heap\_list[j]中
2. 在该大小类中没找到,逐级递增寻找,直到找到为止

如果所有的free\_heap\_list[i]中均无合适的空闲块,说明当前堆中无合适的内存,进行堆的扩展并分配

mm\_free(ptr):释放ptr指向的内存块

需要考虑ptr前后的块是否是空闲块,根据前后块的分配情况进行合并,并将合并之后的内存插入到free\_heap\_list[i]中

mm\_realloc(ptr,size):改变ptr的大小为size

1)size==0:相当于释放内存,直接调用mm\_free,return NULL

2)size!=0:由于最初分配的内存可能会有剩余

判断当前块的大小(present\_size)和size的大小:

1. present\_size >= size 当前块已经满足需求,直接return ptr
2. present\_size < size 当前块不满足,需额外分配内存

i)如果当前块是堆的末尾:直接扩展堆,return ptr

ii)如果后面的块是空闲块,并且next\_size+present\_size满足需求:将next\_block添 加到当前块中,return ptr

iii)否则说明无法拓展当前块使满足需求:重新分配内存并复制原始数据,free oldptr

return newptr

**辅助函数:**

GET\_LEVEL(size):将size依次与每个free\_heap\_list[i]的类大小进行比较,返回合适i

Insert\_free\_node(ptr,size):简单地将空闲块插入到合适的free\_heap\_list[i]的队首,更改链表指针的指向即可

Delete\_free\_node(ptr,size):将ptr从所在地free\_heap\_list[i]中删除,更改链表指针

coalesce(bp):根据bp前后内存的分配情况,合并bp的前后块,并从free\_heap\_list中删除合并的几个块,将插入合并之后的新块

extend\_heap(size):调用mem\_sbrk拓展堆的大小,同时需要注意判断是否向前合并

place(ptr,size):由于分配时分配的块可能会远超请求的内存,需要判断分配的块大小和所需的块大小关系,根据剩余块大小surplus进行操作

1. surplus < MIN\_BLOCK\_SIZE(24B) 剩余空间不足利用,直接返回即可
2. surplus >> size 剩余空间远大于所需内存,将分配内存转移至尾部,头部设为空闲

尽量地把小块内存和大块内存分离开来,防止分配的小块alloced\_block把大块free\_block分割开的情况出现(不断尝试,发现采用surplus>=10size为基准,效果较好)

iii)surplus > size 将ptr对应的块切割,一部分分配,另一部分插入free\_heap\_list

经过不断的修改和尝试,最终获得95分

