

本科生实验报告

实验课程: 操作系统原理实验

实验名称: lab9：实现malloc和free 专业名称: 计算机科学与技术

学生姓名: 张玉瑶

学生学号: 23336316

实验地点: 实验楼B203

实验成绩:

报告时间: 2025年7月12日

**Section 1 实验概述**

我们已经实现了以页为粒度的动态内存分配和释放。但是，我们在程序中使用的往往是以字节为粒度的动态内存管理机制，即我们可以分配和释放任意字节长度的内存。在本项目中，同学们需要实现系统调用malloc和free。malloc用于分配任意字节的内存，free用于释放任意字节的内存。在实现了malloc和free后，需要自行提供测例来测试malloc和free。根据测试方法和输出结果来解释自己程序的正确性。最后将结果截图并详细分析malloc和free的实现思路。

本实验是在lab8的src6下做的。

**Section 2 实验步骤与实验结果**

* 任务要求：实现malloc和free并测试。
* 思路分析：在 C 语言里，堆（Heap）是程序运行时可动态分配的内存区域，与之相关的核心函数是malloc和free。malloc的作用是在堆上分配指定大小的连续内存空间，其返回值是一个指向该内存区域起始位置的指针。free的功能是释放之前通过malloc、calloc或者realloc分配的堆内存，让这部分内存可以被系统再次使用。

而在我们的操作系统设计中，malloc的返回类型直接设置为int并返回分配的首地址即可，free得到需要释放的byte数目后对堆进行释放。

在memory文件声明并实现新的结构体Heap和类HeapManager，用于实现堆和malloc & free。

* 实验步骤：

1.设计结构体Heap。

Heap是我们在user pool中开辟的一块空间，大小为10页，从userpool的start address开始分配。这10页内存专门用来分配和释放以字节为单位的内存。遵循**先进后出**的原则，保证分配的内存永远是从首地址开始的连续的一段字节，未分配的内存也永远是连续的。

在memory.h中声明。

struct Heap{

int count;//分配页数

int startAddr;//分配起始地址

int shengyu;//记录剩余字节数

int tail;//记录heap的最后一个地址

Heap();

void initialize();

};

在memory.cpp中实现。Heap大小固定为10页，首尾地址也固定。

Heap::Heap(){

initialize();

}

void Heap::initialize(){

this->count=10;

this->startAddr=0x4070000;

this->shengyu=40960;

this->tail=startAddr+40960;

printf("build a heap,size is 40960B,from %x to %x.\n",startAddr,tail);

}

2.设计类HeapManager用于对Heap进行malloc和free。

在memory.h中声明。私有成员为Heap类型的heap对象。这就是我们管理的堆。

class HeapManager{

private:

Heap heap;

public:

HeapManager();

void initialize();

int malloc(int byte);

void free(int byte);

};

在memory.cpp中实现。其中，malloc函数是从上一次分配的结束位置开始连续分配，free函数是从已分配的连续字节的最末往前释放。

HeapManager::HeapManager(){

initialize();

}

void HeapManager::initialize(){

heap.initialize();

}

// 实现malloc函数

int HeapManager::malloc(int byte){

if(byte&gt;heap.shengyu){

printf("space is not enough!\n");

return -1;

}

heap.startAddr+=byte;

heap.shengyu-=byte;

printf("malloc from %x to %x, now from %x to %x is available.\n",heap.startAddr-byte,heap.startAddr,heap.startAddr,heap.tail);

return heap.startAddr-byte;

}

// 实现free函数

void HeapManager::free(int byte){

if(byte&gt;40960-heap.shengyu){

printf("out of range!\n");

return;

}

heap.startAddr-=byte;

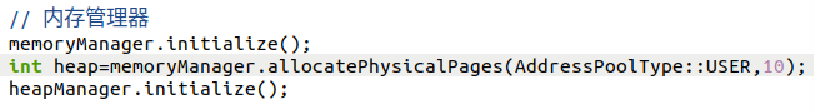
heap.shengyu+=byte;

printf("free from %x to %x, now from %x to %x is available.\n",heap.startAddr,heap.startAddr+byte,heap.startAddr,heap.tail);

}

3.测试。

在set\_up.cpp初始化，因为我们是占用了user pool的前十页内存做的堆，所以我们提前分配掉前十页，避免页分配时会起冲突。



然后，在first\_process中设计测试样例。

void first\_process()

{

int startAddr=heapManager.malloc(100);

heapManager.free(10);

startAddr=heapManager.malloc(100);

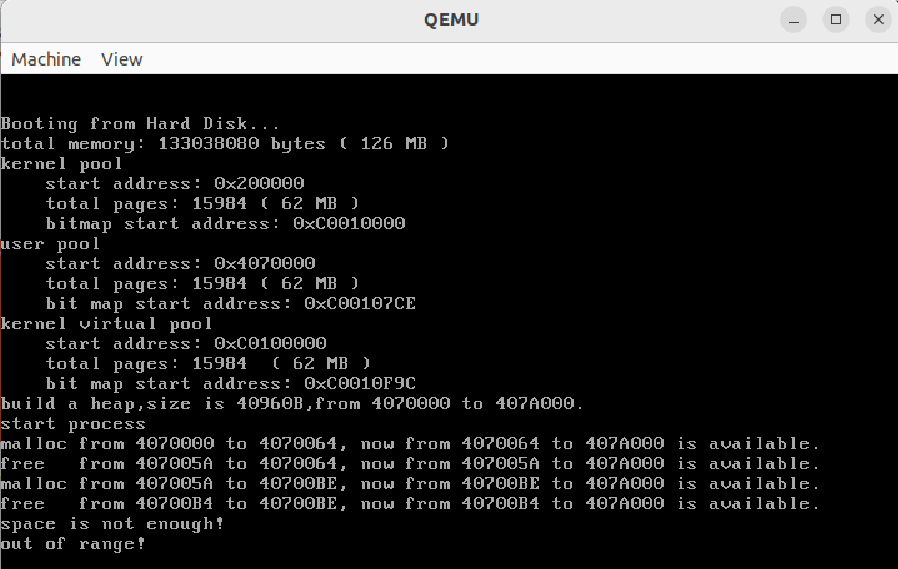
heapManager.free(10);

startAddr=heapManager.malloc(100000);

heapManager.free(1000);

}

得到输出如下。可以看到分配和释放是严格按照先进后出原则进行的。



**Section 5 实验总结与心得体会**

这个实验也不知道我做的符不符合要求，不过还是感慨颇多。这门课程终于结束啦！一开始完成实验还是有点吃力需要到处求助的，慢慢地我可以逐渐克服困难独立完成，还是收获了慢慢的成就感的。对操作系统的认识逐步增加的过程中，也是对我们的专业知识理解更深了，更加体会到自己是一个学计算机的学生，虽然仍然很菜喵喵喵。希望我的操作系统原理最后的分不要那么低，希望我的实验没有A-。感谢老师和助教一个学期的认真教导，给我多多的平时分吧！