**操作系统实验4报告**

**软件框图**

**主流程及其实现**

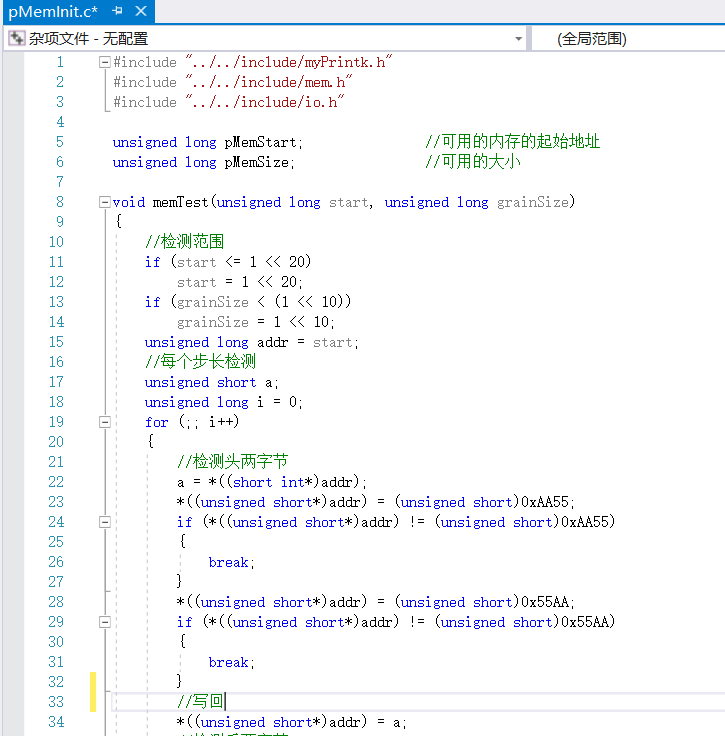
在上次实验的基础上，增加下列内容。

**主要功能模块及其实现**

本次实验中要实现的功能模块分别按五个部分实现

**源代码说明**

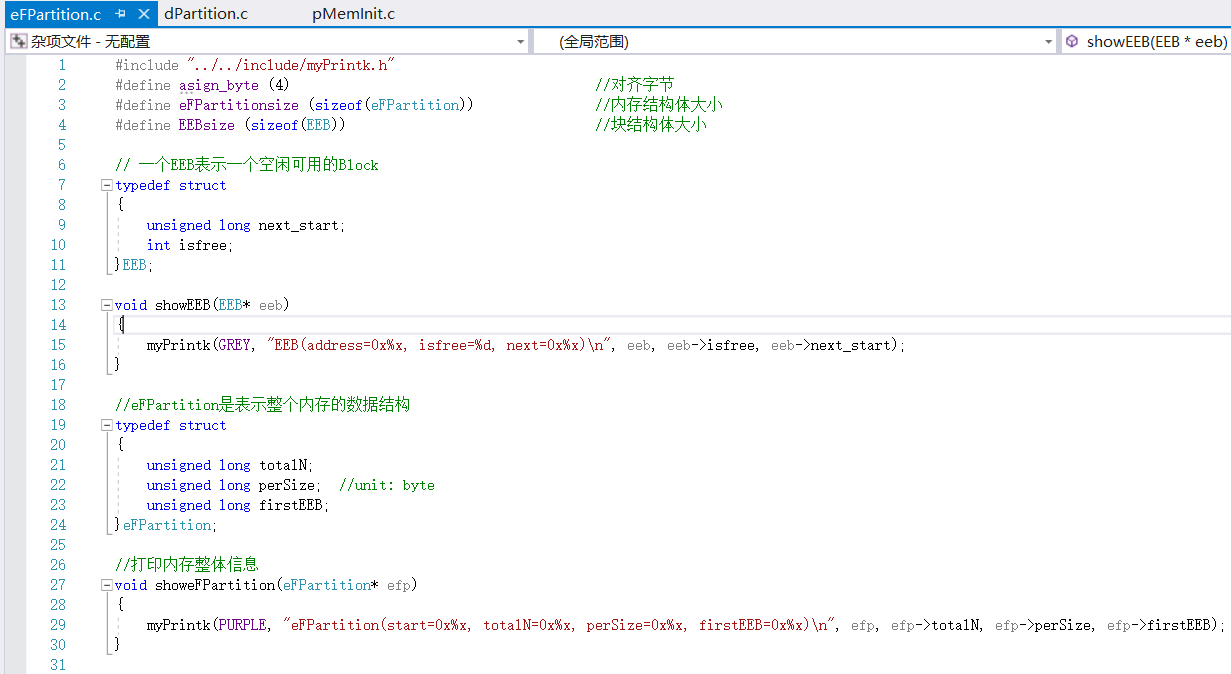
在本次实验的根目录下，除了配置了makefile与DS文件，还有4个文件夹：multibootheader、myOS、output、userApp 。multibootheader中放置了有关multibootheader协议的S文件。在output中，原本为空文件夹，用来输出我们编译后生成的文件。在该文件夹中的目录结构与根目录相似，主要是为了对每个文件输出时不产生混淆。在userApp中存放的是main文件、memTestCase文件与该文件夹下的makefile文件，main文件是用来测试本次实验的功能的入口，memTestCase文件则是用来测试本次实验中实现的内存管理功能的。在myOS文件夹中存放了本次实验的主要源文件。直接存放在此该目录下的文件有DS文件、该文件夹下的makefile文件、链接器文件、一个配置地址的S文件和一个包含了main函数的与multibootheader中的S文件相关联的C文件。这个文件也是从汇编到C的转变。在该目录下则有6个实现相关功能的文件夹：dev、i386、printk、kernel、lib、include。每个文件夹下都有一个相应的makefile文件以及实现功能的C语言源文件。相比于上一次实验，框架中include文件夹下已经将之前所有的myOS中的头文件给包含在了其中，除此之外，我自行添加了颜色宏的头文件，方便用宏定义写入myPrint的颜色参数。lib中的String文件更新了相关的一些字符串处理功能，vsprint里更新了一些例如十六进制数的格式化处理函数等。kernel文件夹中增加pMemInit、dPartition和eFPartition代码文件以实现内存管理。每个子目录下的makefile文件的输出目录都是在output中的同名目录。下面展示相关源代码。



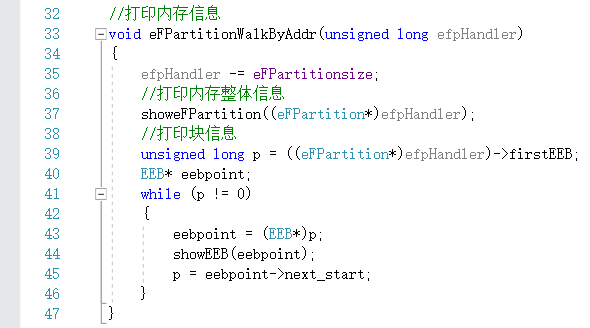
根据老师给的PPT中的检测内存算法实现检测内存功能，图1



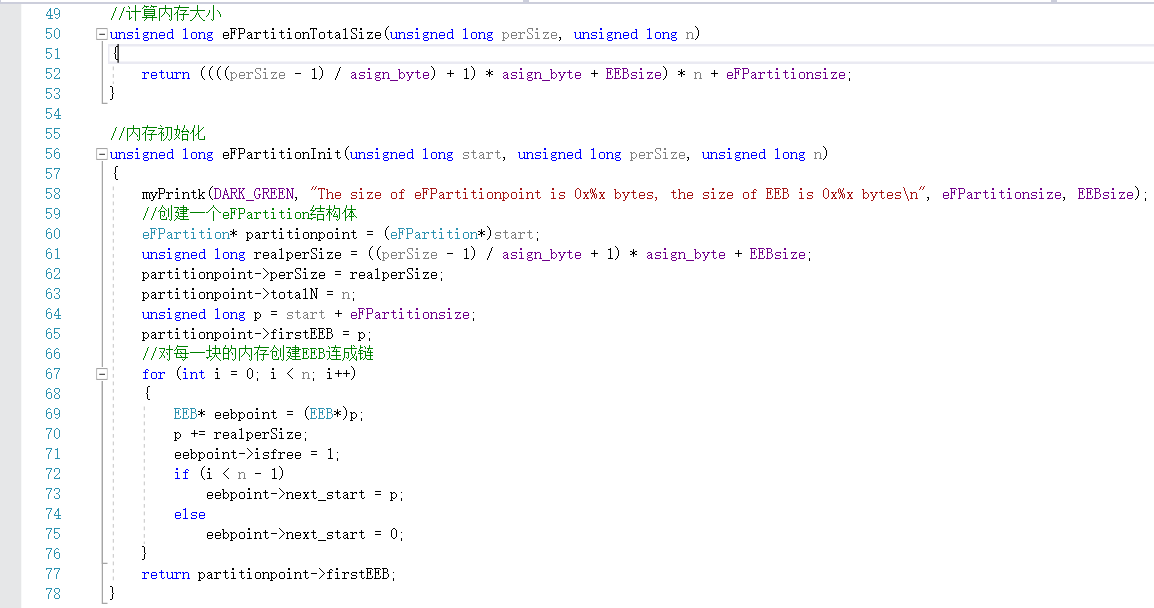
根据老师给的PPT中的检测内存算法实现检测内存功能，图2



实现固定大小分区内存管理，建立结构体信息，主要思想是用isfree标记是否利用了内存。



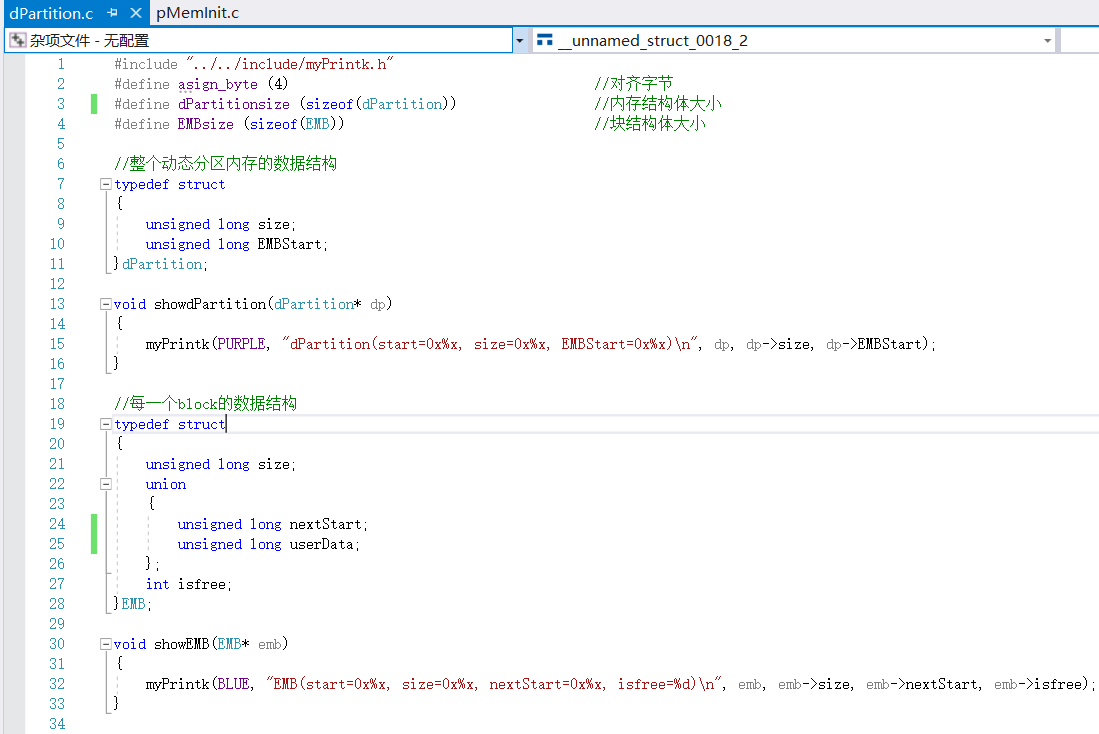
实现打印固定大小分区内存的信息



固定大小分区内存的初始化



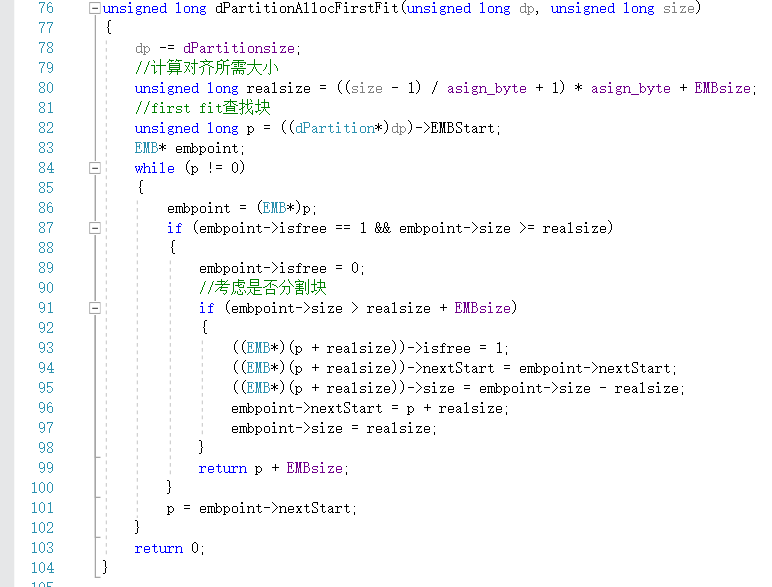
实现固定大小分区内存管理的分配与释放。isfree标志内存是否被占用以避免操作块链表这样的复杂结构。



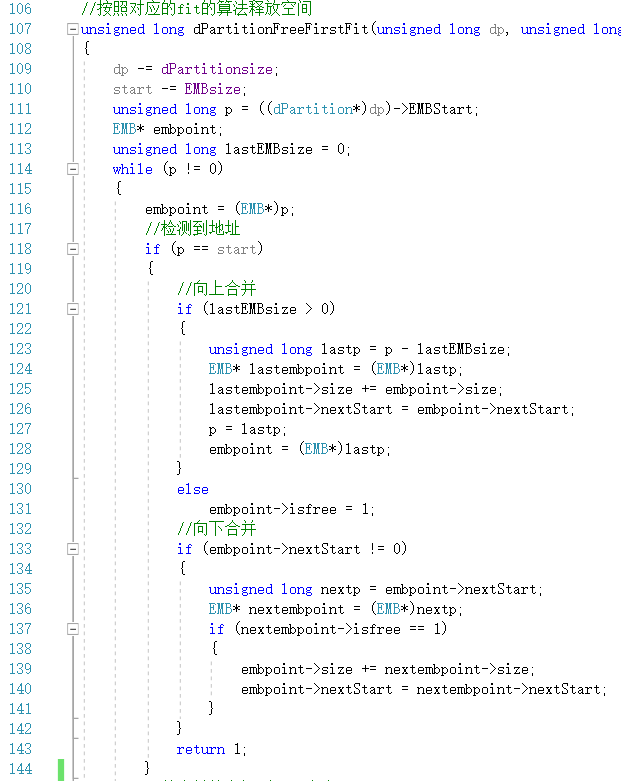
实现动态内存管理的结构体建立



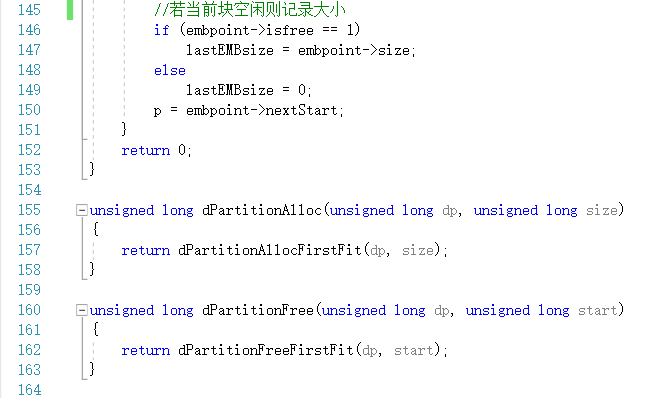
实现动态内存管理的内存信息打印与初始化



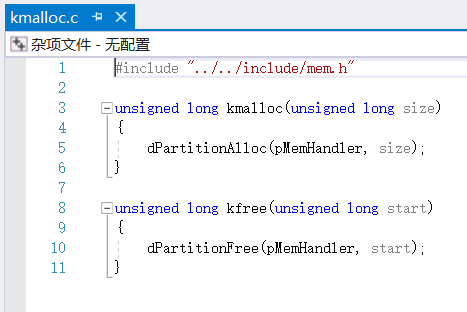
实现动态内存管理的分配功能。利用isfree与size，通过第一适应来确定空闲块



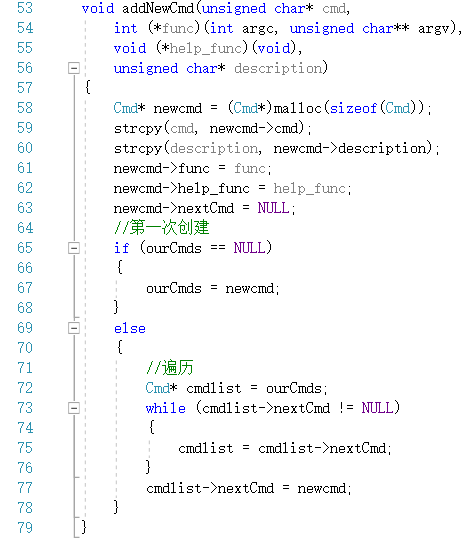
实现动态内存管理的释放功能，实现释放时相比固定大小分区的内存管理，还需要考虑合并空闲块，分别考虑向上与向下合并，其中向上合并时额外记录上一个块的大小。



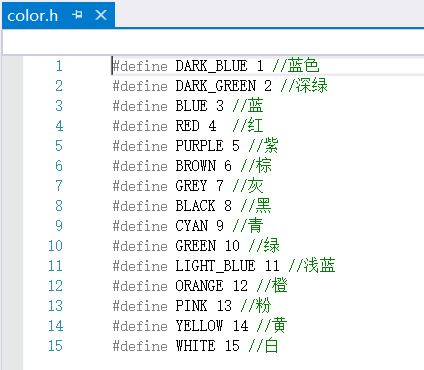
实现动态内存管理的分配功能，此处展示了记录当前内存块大小以便下一个内存块需要合并时使用。然后实现了动态内存管理的分配、释放调用。



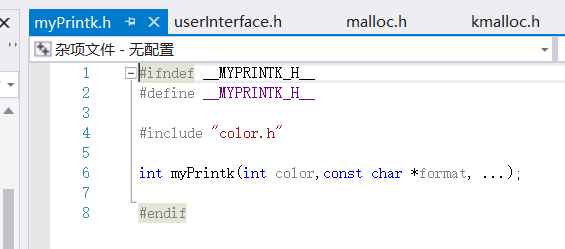
实现kmalloc，malloc的实现类似



实现shell动态添加命令。先动态申请内存，再进行赋值，然后添加到命令集的尾部



颜色宏头文件，对应了myprint中的颜色参数



在内核态的打印函数的头文件中添加了颜色宏头文件，这样在调用printk时可以直接输入颜色宏代替数字参数。



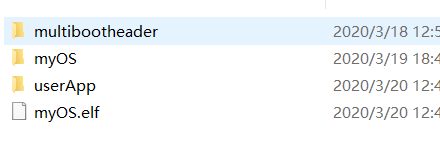
用户态的头文件被整合。用户态的打印函数的头文件中调用颜色宏头文件，使得printf也可以使用宏定义的颜色参数

**代码布局说明**

所有的引导模块将按页（4KB）边界对齐，物理内存地址从1M处开始。本次实验的内存管理统一以4字节对齐，不足4字节补全。

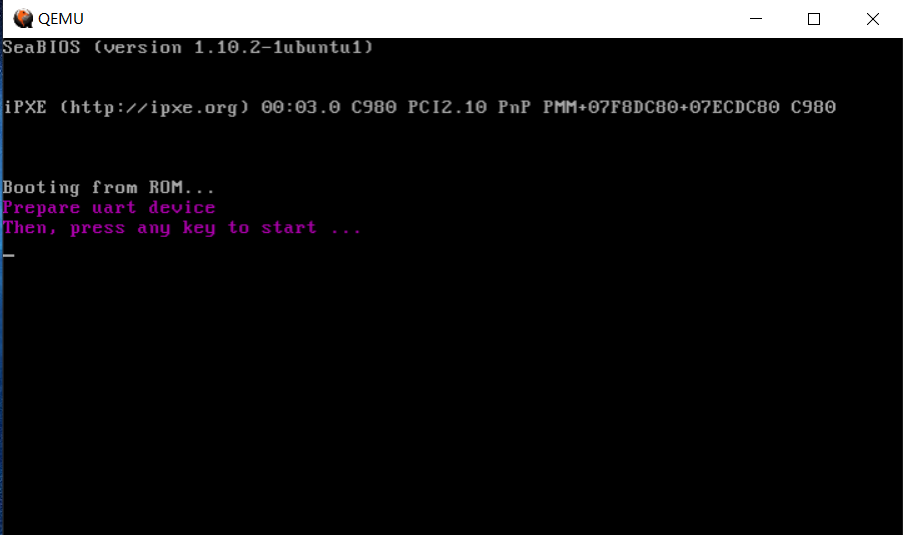
**编译过程说明：**

在Ubuntu中先搜索到lab4的目录，然后通过指令make完成编译，可以看到在output目录中的对应目录中分别输出了与根目录下对应文件相同文件。

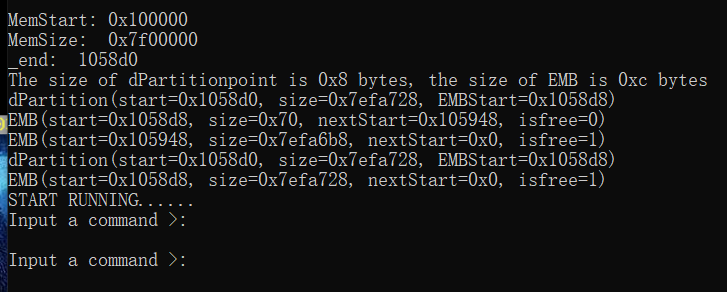


**运行和运行结果说明：**

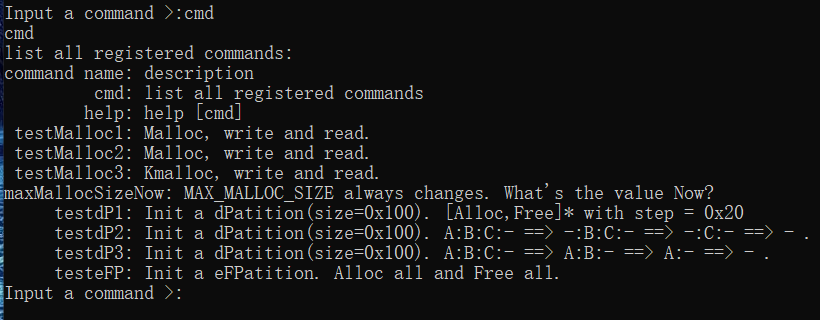
在Ubuntu中通过QEMU启动已经编译生成的bin文件，得到Linux的图形化界面运行结果，然后再通过Ubuntu启动一个交互界面，用于输入与输出。



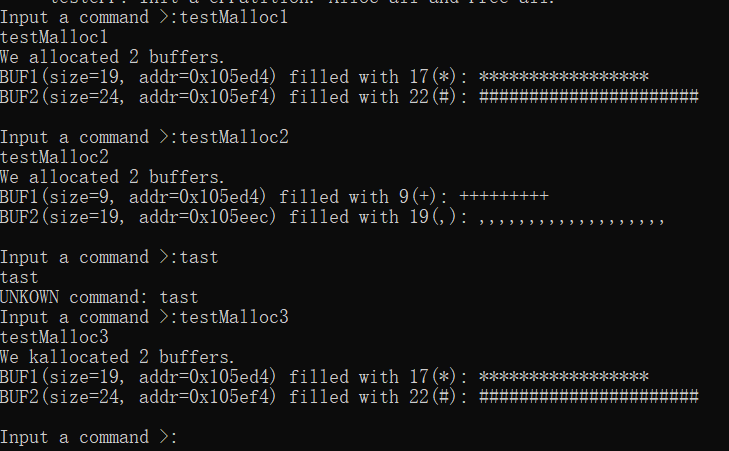
刚运行内核时的QEMU界面。本次实验中在进入main前加入了一个验证输入任意键的过程。



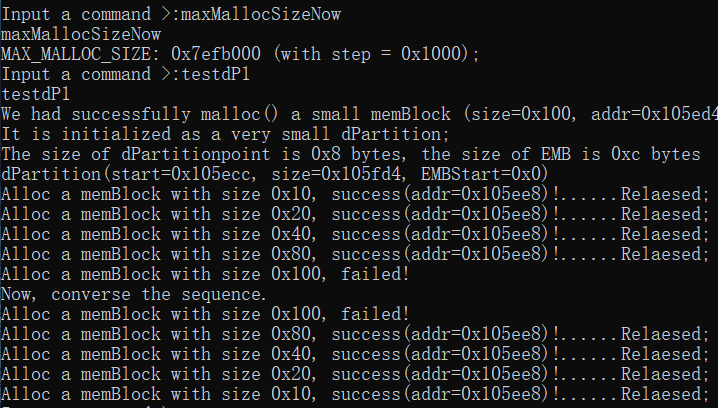
启动了screen界面后按下任意键的结果，可以看到内存初始化同时利用动态内存申请了一块内存并释放前后的内存信息，EMB块从两块变成一块，以及其起始地址可占用大小。



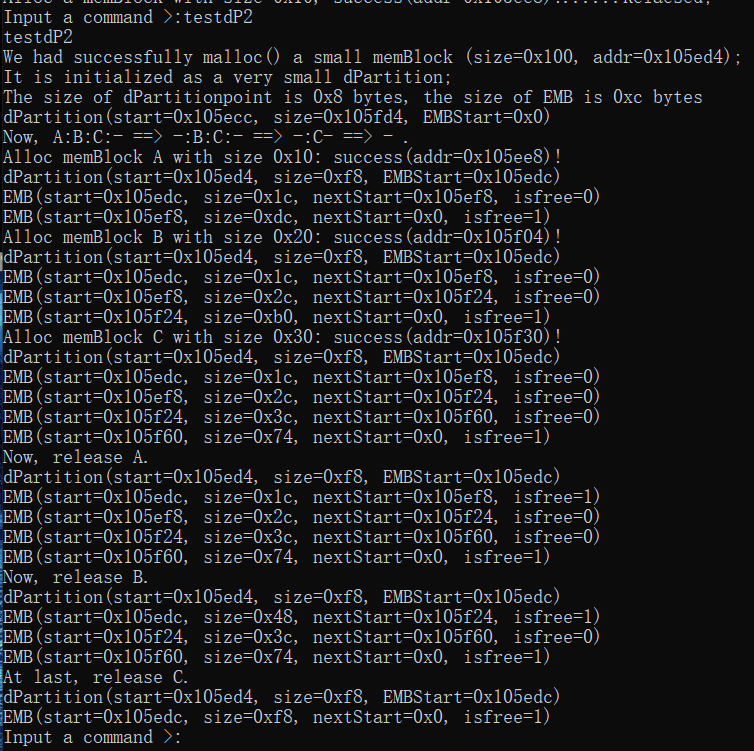
输出cmd查看本次实验中动态添加的所有命令



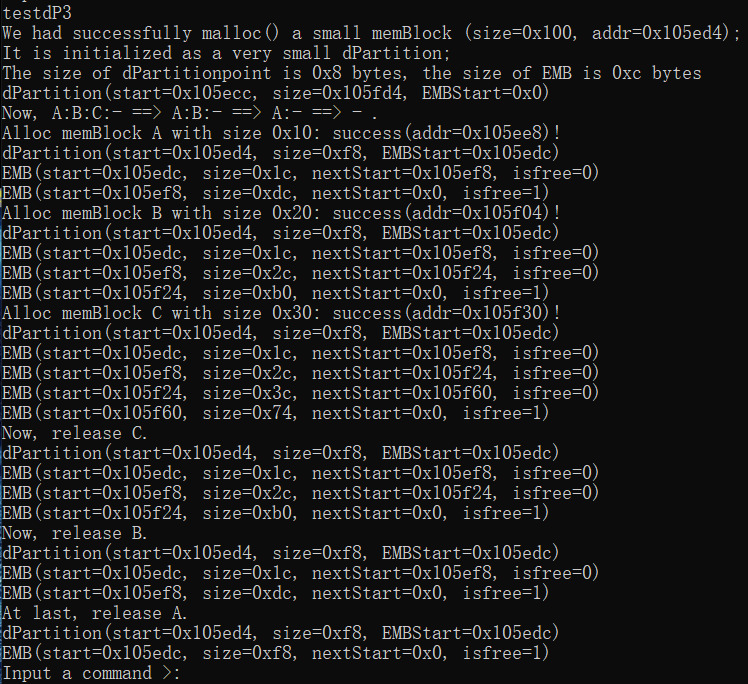
对前3个内存测试指令进行调用的结果，其中前两个是malloc，第三个是kmalloc



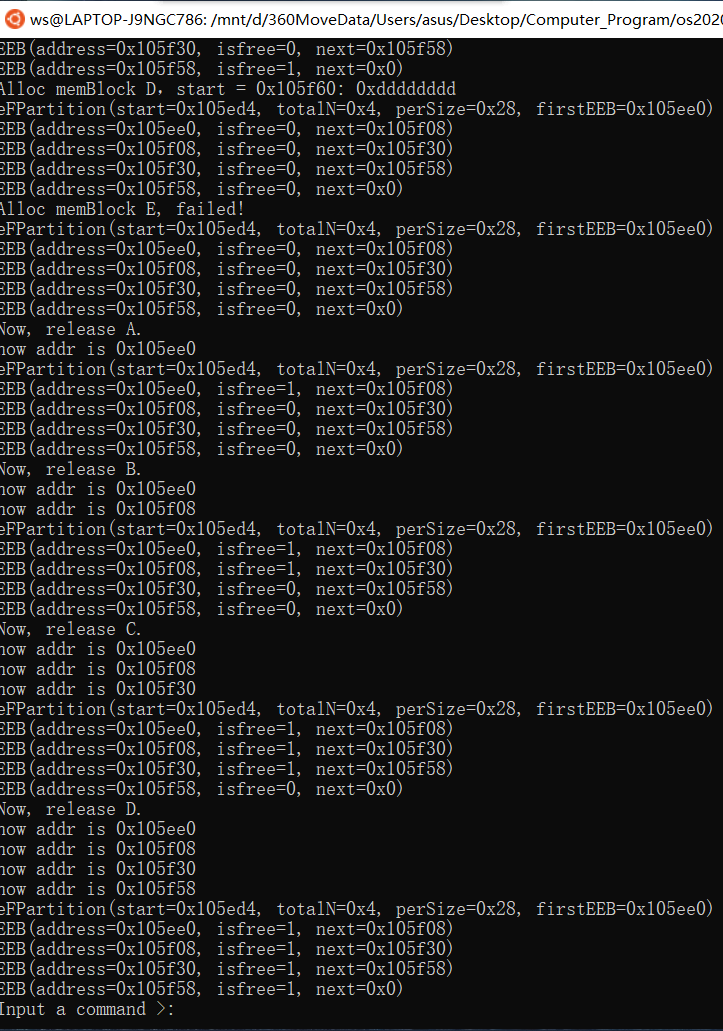
测试最大可分配内存以及第一个固定大小内存分配



测试一个动态内存分配



测试另一个顺序释放的动态内存分配



测试一个固定大小分区的内存分配与释放

**遇到的问题和解决方案：**

1. 不能理解两种内存管理机制

通过网络工具查询和了解。

1. 不理解覆盖写入的操作方式

咨询助教并选择了直接用C语言的指针进行修改值。