Lab4 实验文档

实验目标

为我们的OS内核实现内存管理，为用户提供malloc/free接口（选做：为系统提供kmalloc/kfree接口）。

框架图

```
.
|-- multibootheader
|   `-- multibootHeader.S
|-- myOS
|   |-- dev
|   |   `-- ...
|   |-- i386
|   |   `-- ...
|   |-- include                     <- 相应新增头文件
|   |   |-- kmalloc.h
|   |   |-- malloc.h
|   |   |-- mem.h
|   |   `-- ...
|   |-- kernel
|   |   |-- mem                     <- 新增内存管理模块
|   |   |   |-- Makefile
|   |   |   |-- dPartition.c        <- 动态分区管理算法实现
|   |   |   |-- eFPartition.c       <- 等大小分区管理算法实现
|   |   |   |-- malloc.c            <- 对外提供的接口
|   |   |   `-- pMemInit.c          <- 内存大小检测，调用相应算法初始化
|   |   |-- tick.c
|   |   `-- wallClock.c
|   |-- lib
|   |   `-- ...
|   |-- myOS.ld
|   |-- osStart.c
|   |-- printk
|   |   `-- ...
|   |-- start32.S
|   `-- userInterface.h             <- 内核提供给userApp的所有接口通过该文件提供
|-- source2img.sh
`-- userApp
    |-- main.c
    |-- memTestCase.c               <- 本次实验的测试样例
    |-- memTestCase.h
    |-- shell.c                     <- 需新增addNewCmd功能
    `-- shell.h
```

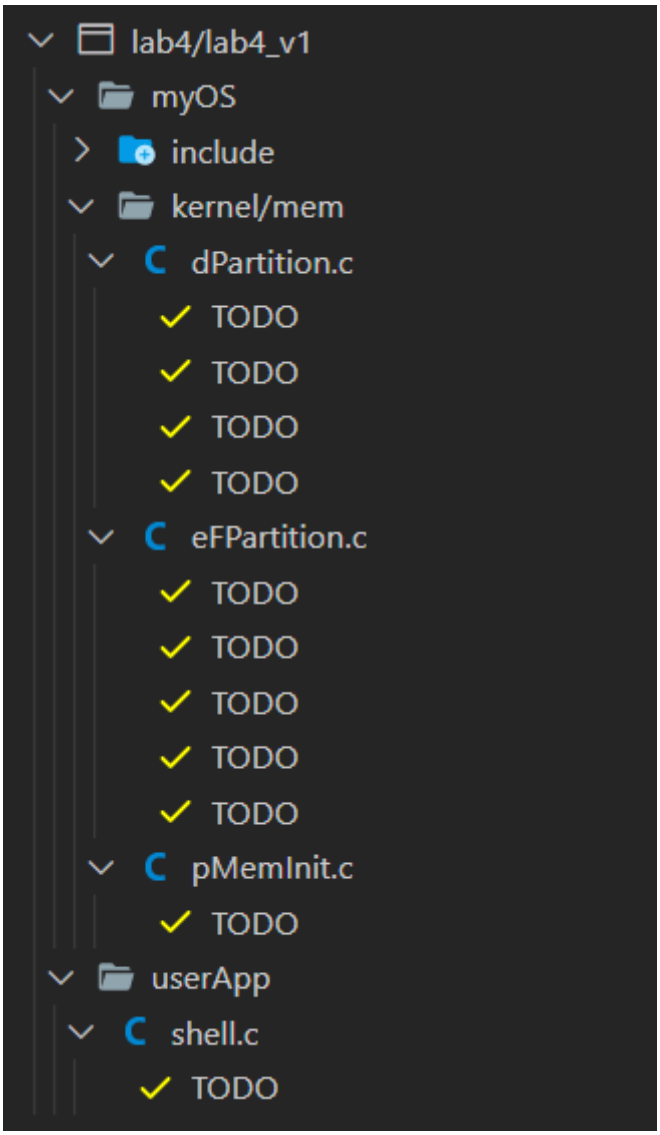
一些说明：

若要实现KMEM和UMEM分别管理，需要相应的新增 `kmalloc.c` 文件，在 `memTestCase.h` 中自己编写相应的测试。

实验任务

本次实验所涉及的大部分为课程所讲内容，重点是如何实现这些算法。

本实验所有需要实现的函数均使用TODO标出（使用VSCode中的 `Todo Tree` 插件可以清晰查看所有TODO任务）



完成实验的步骤推荐：

首先从流程上弄清楚从OS内核启动到进入 `userAPP`，本次实验相较于上一次在流程上多做了什么？（如 `osStart.c` 中新增的 `pMemInit();` 它背后又调用了哪些函数，实现了什么功能？）

其次从模块上弄清楚这次实验相较于上次新加了什么？新加的模块如何和原来的OS内核结合（即接口是什么，在哪里？）

在上面两个宏观层面搞明白后进入局部模块，即我们要完成的 `kernel/mem`，首先还是从宏观层面理解每个函数的作用是什么？函数之间的依赖是什么？函数如何一步步封装，最后形成 `malloc/free` 接口的。

在这之后就可以开始着手实现每个函数了。

- 一个易混淆的点
  - 可用内存检测是在检测可正常读写的内存大小，而不是空闲内存大小
- 几个难点
  - 对于空闲链表的理解以及它如何运作
  - 释放内存后相邻空闲块的合并
- 一个提醒
  - 对于KMEM的实现，实际上就是用实现的内存管理算法去管理另一块内存空间。和UMEM实现大同小异

完成这些函数后，启动shell在其中运行`userApp/memTestCase.c`中`memTestCaseInit`函数新增的那些命令，进而实现测试。

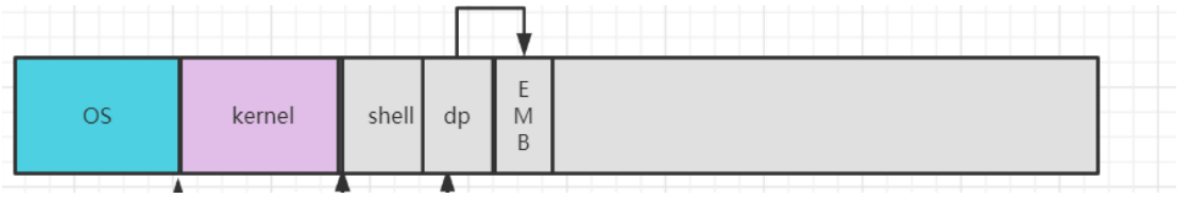
流程演示

下面以`memTestCase.c`中`testdP2`为例写一下动态内存分配的流程

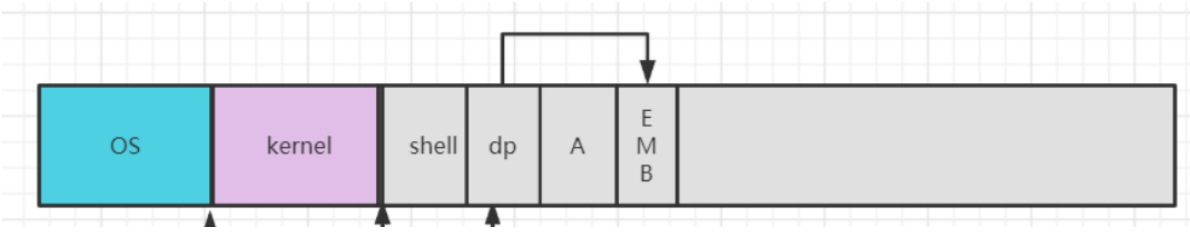
该过程依次分配A，B，C，然后依次释放A，B，C。

首先在实现KMEM情况下，在OS之后会分配kernel内存空间，然后对内存初始化，产生dp 结构体和第一个EMB结构体，调用`addNewCmd`函数之后会在user内核中占据一部分位置存储shell命令

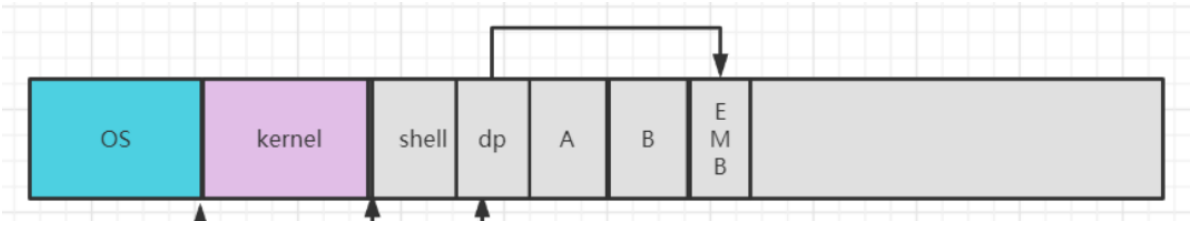
（dp 和 shell位置画反了，大家注意一下 QWQ）



申请A空间

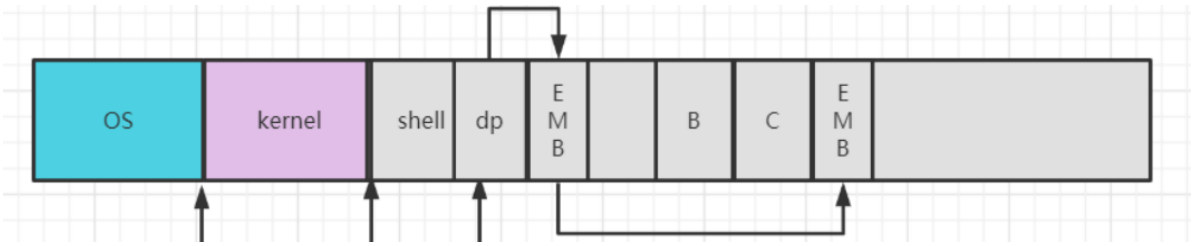


申请B空间



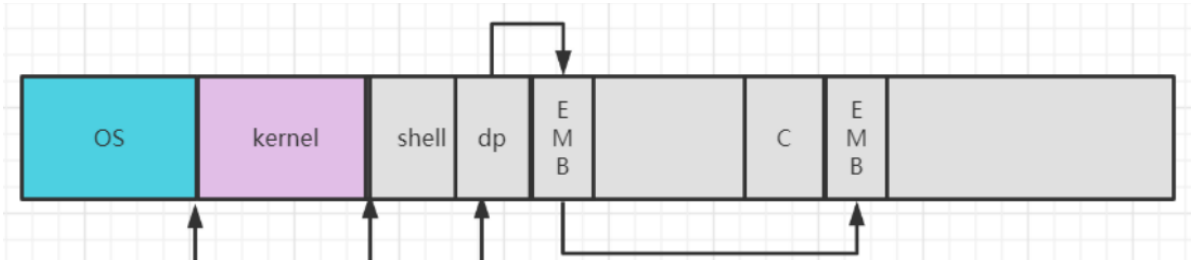
申请C空间

释放A空间（将产生两个EMB）

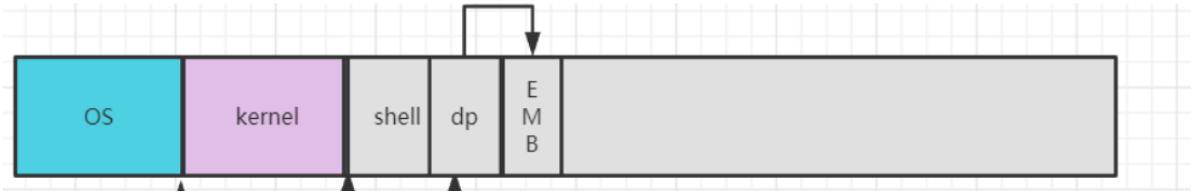


（忽略这里的的三个箭头）

释放B空间（相邻空闲块合并）



释放C空间（相邻空闲块合并）



思考题（问题1为5分，2为15分）

1. 请写出动态分配算法的malloc接口是如何实现的（即malloc函数调用了哪个函数，这个函数又调用了哪个函数...）
2. 运行`memTestCaseInit`那些新增的shell命令，会出现什么结果，即打印出什么信息（截图放到报告中）？是否符合你的预期，为什么会出现这样的结果。（详细地讲一两个运行结果，大同小异的可以从简）

提交要求

截止时间：2022.5.21晚23.59分！逾期不接受补交。

提交内容

- 提交平台：BB平台对应板块
- 提交方式：按照上面的框架图组织文件，将其**打包压缩**（压缩格式不限），命名为学号_姓名_lab4。并且将report**单独**再提交一次。
- 也就是说你最终要上传至BB系统的为 学号_姓名_lab4.zip和report.pdf这两个文件

实验报告要求

- 从简，重点是问题回答模块
- 格式要求为PDF，模板如下

```
Lab4 实验报告

一、实验目标
    简要说明这次实验需要完成什么

二、源代码说明
    对一些关键的设计进行说明，比如如何维护空闲链表，如何在释放后合并空间

三、问题回答
    回答上面提出的几个问题
    第二个问题相当于实验结果展示

四、遇到的问题和解决方法（可选，不计入评分）
    自己遇到的Bug和解决的方法。

也可以对实验进行建议和吐槽，非常感谢反馈！
```

其他要求

代码风格在cs学习过程中是非常重要的一部分，有合理缩进、标识符安排，以及适量有用的注释等等，都是良好的习惯。

评分标准

总分100分

评价项目	分值
能正常编译运行	60
实验报告	30（问题回答20分）
代码风格和注释	10
KMEM 和 UMEM 的实现	10 (bonus)

此处附上往年实验视频讲解链接 https://www.bilibili.com/video/BV1SA41137aG