



中国科学技术大学
University of Science and Technology of China

011174.01: Operating System 操作系统原理与设计

Project 4: Memory Management

陈香兰(xlanchen@ustc.edu.cn)
高效智能计算实验室, CS, USTC @ 合肥
嵌入式系统实验室, CS, USTC @ 苏州

2024/5/11

实验4基础



中国科学技术大学
University of Science and Technology of China

- 本实验在实验3的基础上进行
- 在实验3提交的截止时间过后，同学们可以就实验3的内容互通有无
- 实验4可以在其他同学实验3的基础上进行
 - 无论你使用哪一个（包括自己的），请在实验报告中标注，实验4的基础来自哪个同学（可以是自己）
 - 给你使用的实验3打分

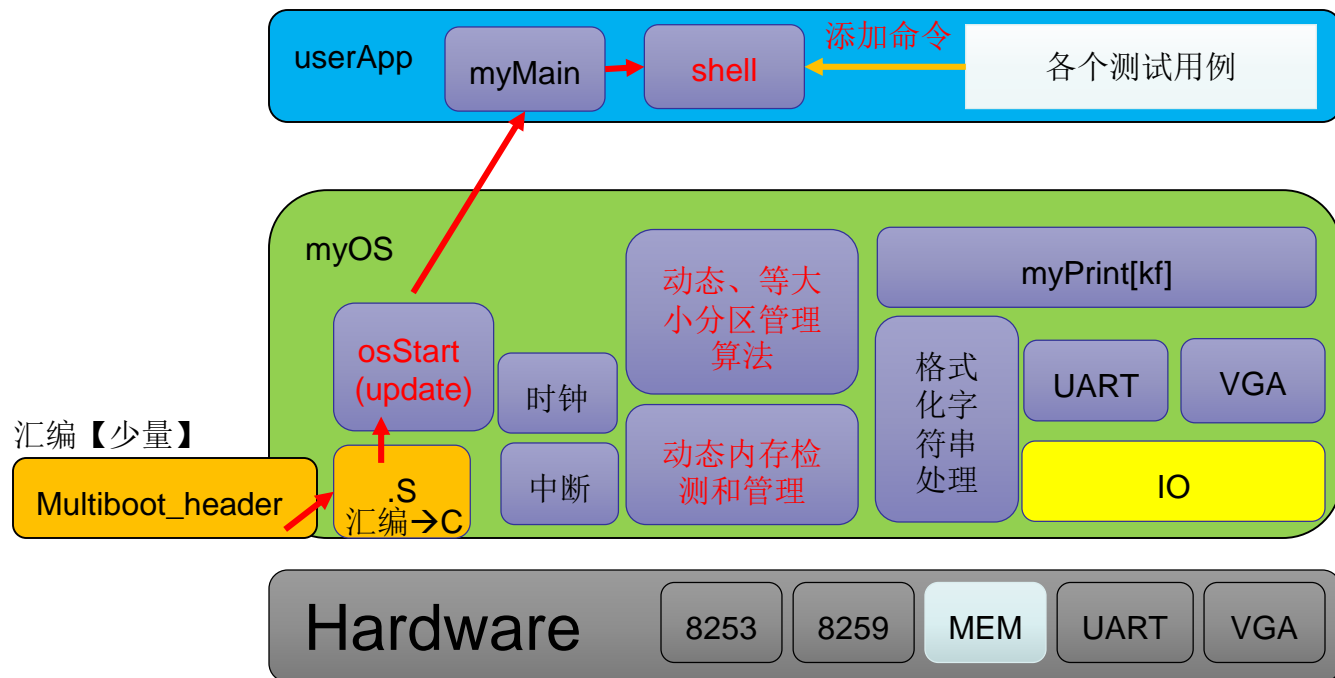
- **【必须】** 内存检测，确定动态内存的范围
- **【必须】** 提供动态分区管理机制dPartition
- **【必须】** 提供等大小固定分区管理机制ePartition
- **【必须】** 使用动态分区管理机制来管理所有动态内存
- **【必须】** 提供kmalloc/kfree和malloc/free两套接口，分别提供给内核和用户
- **【可选】** kmalloc/kfree、malloc/free相互隔离
- **【必须】** 自测
 - shell改用malloc/free来动态注册新命令（将涉及部分字符串处理）
 - 自编测试用例，并添加到shell命令列表中，执行新增的shell命令

1. 软件架构和功能说明
2. 主要功能模块说明
 1. 内存检测和动态内存
 2. 等大小分区管理算法
 3. 动态分区管理算法
 4. 动态内存管理、`kmalloc/kfree`和`malloc/free`
 5. 改造`shell`以动态分配命令结构
3. 关于头文件的使用
4. 验收标准

1 软件架构和功能



中国科学技术大学
University of Science and Technology of China



流程:

Multiboot_header

为进入C程序准备好上下文

初始化操作系统

调用userApp入口
myMain (自测) +shell

multiboot_header→myOS→userApp

主要功能模块【新】:

内核: 内存(算法、检测、管理)
用户: 新功能测试

测试:

被测功能: 动态内存管理功能、两种算法
自测: userApp
他测: 替换userApp或增加shell命令

参考目录结构



中国科学技术大学
University of Science and Technology of China

- Makefile
- README_mem.txt
- multibootheader
 - multibootHeader.S
- myOS
 - Makefile
 - dev
 - Makefile
 - i8253.c
 - i8259A.c
 - uart.c
 - vga.c
 - i386
 - Makefile
 - io.c
 - irq.S
 - irqs.c
 - include
 - i8253.h
 - i8259.h
 - io.h
 - irq.h
 - kmalloc.h
 - malloc.h
 - mem.h
 - myPrintk.h
 - string.h
 - uart.h
 - vga.h
 - vsprintf.h
 - wallClock.h

- kernel
 - Makefile
 - mem
 - Makefile
 - dPartition.c
 - eFPartition.c
 - malloc.c
 - pMemInit.c
 - tick.c
 - wallClock.c
- lib
 - Makefile
 - string.c
- myOS.ld
- osStart.c
- printk
 - Makefile
 - myPrintk.c
 - types.h
 - vsprintf.c
- start32.S
- userInterface.h
- source2img.sh
- userApp
 - Makefile
 - main.c
 - memTestCase.c
 - memTestCase.h
 - shell.c
 - shell.h

2.1 内存检测和动态内存



中国科学技术大学
University of Science and Technology of China

- OS通过检测了解系统中的内存配置情况
- 本实验采用简化版检测
 - 假设从1M开始（1M以内的不考虑【可选】）
 - 假设只有1块连续的内存空间（多个非连续的【可选】）
 - 通过检测得到这个内存块的大小
 - 检测算法 `void memTest(unsigned long start, unsigned long grainSize)`
 - 从start开始，以grainSize为步长，进行内存检测
 - 检测方法：
 - 1) 读出grain的头2个字节
 - 2) 覆盖写入0xAA55，再读出并检查是否是0xAA55，若不是则检测结束；
 - 3) 覆盖写入0x55AA，再读出并检查是否是0x55AA，若不是则检测结束；
 - 4) 写回原来的值
 - 5) 对grain的尾2个字节，重复2-4
 - 6) 步进到下一个grain，重复1-5，直到检测结束

2.1 内存检测和动态内存



- 可以自由分配管理的**动态内存**，应当不包含已用内存（范围在myOS.ld中定义，从1M到_end）

无论是从汇编文件还是C文件中都可以访问_end



注意：当前使用的栈在哪里？

```
5 SECTIONS {
6     . = 1M;
7     .text : {
8         *(.multiboot_header)
9         . = ALIGN(8);
10        *(.text)
11    }
12
13    . = ALIGN(16);
14    .data      : { *(.data*) }
15
16    . = ALIGN(16);
17    .bss       :
18    {
19        __bss_start = .;
20        __bss_start = .;
21        *(.bss)
22        __bss_end = .;
23    }
24    . = ALIGN(16);
25    _end = .;
26    . = ALIGN(512);
27 }
28
```

Zero out the BSS segment

zero_bss:

```
    cld                                # make direction flag count up
    movl    $_end, %ecx                # find end of .bss
    movl    $__bss_start, %edi         # edi = beginning of .bss
    subl    %edi, %ecx                 # ecx = size of .bss in bytes
    shrl    %ecx                       # size of .bss in longs
    shrl    %ecx

    xorl    %eax, %eax                # value to clear out memory
    repne                                 # while ecx != 0
    stosl                               # clear a long in the bss
```

例：从汇编文件中访问

```
extern unsigned long _end;
```

```
void pMemInit(void){
```

```
    unsigned long _end_addr = (unsigned long) &_end;
```

例：从C文件中访问

2.2 等大小分区管理算法



- 给定一个空闲内存块A，按等大小分区的方式加以管理
 - 指定大小（这个大小是用户所需大小）
 - 根据某些原则，调整大小为算法所需大小 不能影响最终用户可用大小
 - 将空闲内存块，按照算法所需大小划分成若干块，并按照算法所需进行管理（应**便于分配/回收**、省空间）
- 接口【必须统一】：

```
unsigned long eFPartitionTotalSize(unsigned long perSize, unsigned long n); //用于计算A的合理大小

unsigned long eFPartitionInit(unsigned long start, unsigned long perSize, unsigned long n); //返回句柄
unsigned long eFPartitionAlloc(unsigned long EFHandler);
unsigned long eFPartitionFree(unsigned long EFHandler, unsigned long mbStart);
```

- 建议用法：
 - 1) 计算A的合理大小，使用eFPartitionTotalSize
 - 2) 向系统申请分配A，使用kmalloc或malloc
 - 3) 对A进行划分和管理，使用eFPartitionInit
 - 4) 按需分配或回收，分别使用eFPartitionAlloc和eFPartitionFree

一般，适用于大小已知、最大个数已知且不变的数据结构、缓冲区等的管理

仅仅是几个建议：

1. 用户给定的大小，可能是不对齐的，建议对齐
 - 例如4、8、16、32字节对齐
2. 不建议使用着色算法（为什么需要着色？）
3. 建议充分利用空闲块本身进行空闲块管理，以减少消耗
4. 建议分配出来的内存块前后有隔离带（为什么需要隔离带？）
可以利用隔离带的空间存储某些信息以提高回收性能
5. 建议利用空闲链表来提高分配性能
6. 不建议将空闲块填充为特殊字符（调试除外）

2.3 动态分区管理算法



中国科学技术大学
University of Science and Technology of China

- 给定一个空闲块B，按动态分区的方式加以管理
一开始是一整块，随着分配/回收的进行，逐渐碎片化
 - 考虑分割和合并；考虑采用哪个分配算法？例如：firstfit? bestfit? worstfit?
 - 每次用户请求的大小不一，是否考虑对齐？

- 接口【必须统一】

```
unsigned long dPartitionInit(unsigned long start, unsigned long size); //返回句柄dpHandler
unsigned long dPartitionAlloc(unsigned long dpHandler, unsigned long size);
unsigned long dPartitionFree(unsigned long dpHandler, unsigned long start);
```

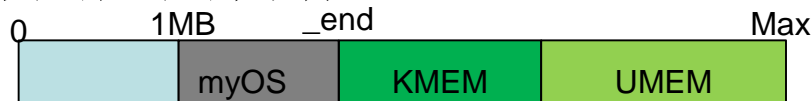
- 建议用法：
 - 1) 向系统申请分配B，使用kmalloc或malloc;
 - 2) 对B进行管理，使用dPartitionInit进行初始化
 - 3) 按需分配或回收，分别使用dPartitionAlloc和dPartitionFree，或封装后的接口

2.4 动态内存管理



- 建议:

- 采用动态分区管理算法来管理动态内存
- 将动态分区分成2大块，一块用作内核用内存KMEM，一块用作用户用内存UMEM



- KMEM和UMEM进一步采用动态分区管理算法来管理
 - KMEM上封装接口kmalloc和kfree
 - UMEM上封装接口malloc和free

```
unsigned long malloc(unsigned long size);  
unsigned long free(unsigned long start);
```

2.5 Shell的变化



中国科学技术大学
University of Science and Technology of China

- 接口:

```
void addNewCmd( unsigned char *cmd,           //命令名
               int (*func)(int argc, unsigned char **argv), //命令入口
               void (*help_func)(void),      //该命令的help入口, 可为空
               unsigned char* description);  //该命令的描述
```

- 功能: 注册一个新的命令

- 建议: 一个命令的元信息为

- (命令名, func, help_func, 描述命令的字符串)

动态分配cmd所需的空间;
等大小, 或不等大小

- Shell的其他部分配套修改

- 涉及的其他模块可能有: 字符串处理

根据具体实现, 可能涉及可能不涉及

例如长度受限的字符串拷贝

3 使用头文件



中国科学技术大学
University of Science and Technology of China

- 是时候把头文件使用起来了

```
1  #ifndef __XXXX_H__
2  #define __XXXX_H__
3
4  /*
5   * 此处各种声明:
6   * 1) 可以嵌套包含其他头文件
7   * 2) 可以使用宏定义
8   * 3) 可以声明变量
9   * 4) 函数原型
10  * 5) 等等
11  */
12
13  #endif
```

建议:

- 1) 模块化管理
- 2) 区分内核接口和用户接口

注意: 文件路径

涉及已实现的各个模块

【必须】内核提供给userApp的所有接口通过文件[myOS/userInterface.h](#)提供

4 验收标准



中国科学技术大学
University of Science and Technology of China

- 提交：源代码打包 + 实验报告；验收标准如下：
 - 完成源代码编写和调试，能够编译正确运行
 - 实现主流程，提供规定接口
 - 实现主要功能，提供规定接口
 - 将源代码进行合理的组织、提供相应的Makefile，能够生成myOS
 - 提供编译和运行脚本
 - 提交实验报告，实验报告中包括
 - 给出软件的框图，并加以概述
 - 详细说明主流程及其实现，画出流程图
 - 详细说明主要功能模块及其实现，画出流程图
 - 源代码说明（目录组织、Makefile组织）
 - 代码布局说明（地址空间）
 - 编译过程说明
 - 运行和运行结果说明
 - 遇到的问题和解决方案说明

Q & A