

# 实验七 内存管理算法

姓名：姜庆彩

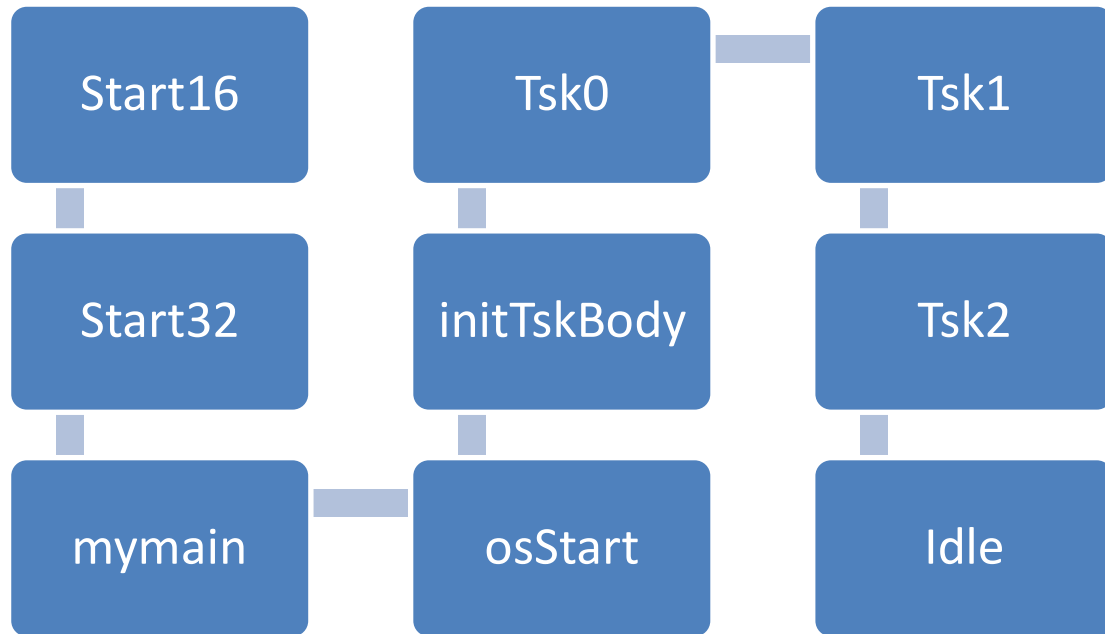
学号：PB15051087

## 计算机科学与技术学院

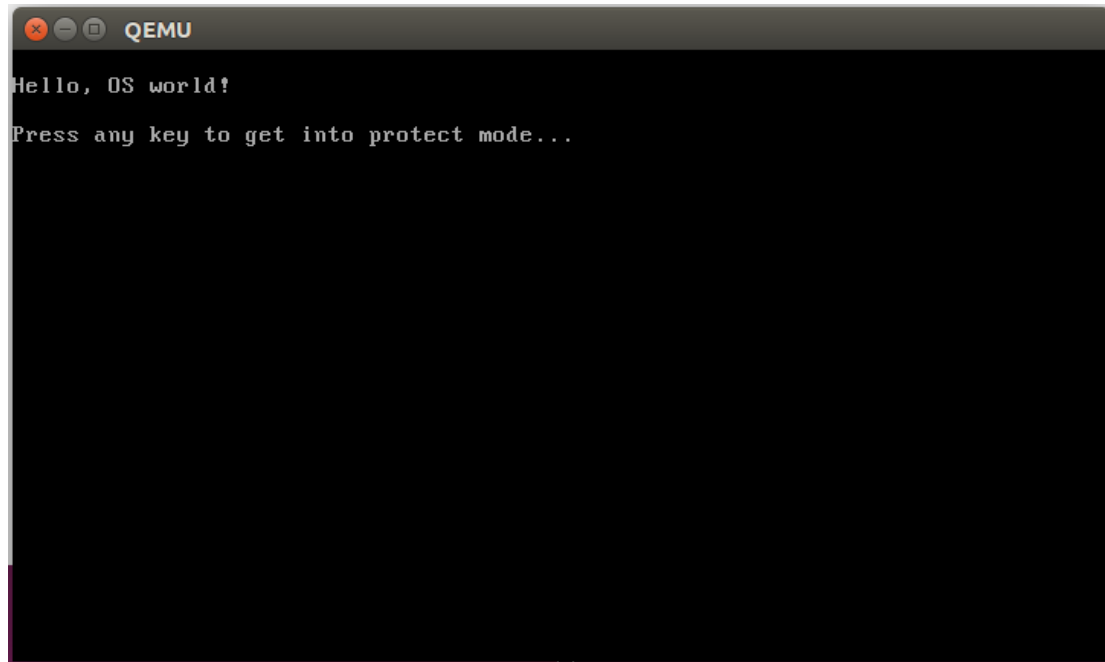
### ● 实验内容：

1. 检测物理内存：实现内存有效性检查方案以检查你所管理的动态内存。
  - 为简单起见，允许直接从 1MB 开始。
  - 接口：pMemStart（?=1MB）和 pMemSize。
  - 操作系统初始化过程中进行检查，并汇报 pMemStart 和 pMemSize 的值（要求 16 进制表示）
  - 附加（非强制要求）：从 0 地址开始。
2. 实现动态分区算法：dPartition
  - 给定任意大小的内存区域，按照动态分区算法对其进行初始化。接口：dPartitionInit()。（要求：若内存区域太小，小于最小大小，进行失败提示）
  - 从动态分区中分配一个指定大小的分区（允许进行 4 字节、8 字节或你自定义的某个大小对齐）。接口：dPartitionAlloc()
  - 释放一个分区。接口：dPartitionFree()
  - 将检测到的动态物理内存，按照动态分区的方式进行管理
  - 重新封装动态物理内存的分配和回收接口，提供 malloc 和 free 接口给 userApp。
  -
3. 实现等大小固定分区算法：eFPartition
  - 给定分区大小和分区个数，结合你的内部管理数据结构开销，计算出总大小。接口：eFPartitionTotalSize()
  - 将给定内存区域初始化为若干个等大小固定分区。初始化接口：eFPartitionInit()
  - 分配一个固定大小分区。接口：eFPartitionAlloc()
  - 释放一个固定大小分区。接口：eFPartitionFree()
  - 修改你的任务池分配算法：从动态物理内存中，分配一个动态分区，该动态分区能容纳指定个数的任务和内部数据管理开销，使用等大小固定分区算法管理这个动态分区。
4. 修改 osStart 原语，以增加内存管理功能

- 测试用例使用老师给的三个 Task
- 实验流程：



- 实验结果：



内存有效性检查：每 2MB 输出一个 BlockCheckDone

```
QEMU
myMain:HELLO WORLD!,10
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
Block Check Done
```

检测动态分区空间：  
每次空间大小增加 0x100，直至越界

```
QEMU
dPartition Size Test: i= 0xe6000 Accessed!
dPartition Size Test: i= 0xe7000 Accessed!
dPartition Size Test: i= 0xe8000 Accessed!
dPartition Size Test: i= 0xe9000 Accessed!
dPartition Size Test: i= 0xea000 Accessed!
dPartition Size Test: i= 0xeb000 Accessed!
dPartition Size Test: i= 0xec000 Accessed!
dPartition Size Test: i= 0xed000 Accessed!
dPartition Size Test: i= 0xee000 Accessed!
dPartition Size Test: i= 0xef000 Accessed!
dPartition Size Test: i= 0xf0000 Accessed!
dPartition Size Test: i= 0xf1000 Accessed!
dPartition Size Test: i= 0xf2000 Accessed!
dPartition Size Test: i= 0xf3000 Accessed!
dPartition Size Test: i= 0xf4000 Accessed!
dPartition Size Test: i= 0xf5000 Accessed!
dPartition Size Test: i= 0xf6000 Accessed!
dPartition Size Test: i= 0xf7000 Accessed!
dPartition Size Test: i= 0xf8000 Accessed!
dPartition Size Test: i= 0xf9000 Accessed!
dPartition Size Test: i= 0xfa000 Accessed!
dPartition Size Test: i= 0xfb000 Accessed!
dPartition Size Test: i= 0xfc000 Accessed!
```

固定大小分区检测：进行三轮，每次输出分配空间的起始位置

```
QEMU
dPartition Size Test: i= 0x2fd000 Accessed!
dPartition Size Test: i= 0x2fe000 Accessed!
dPartition Size Test: i= 0x2ff000 Accessed!
dPartition Size Test: i= 0x300000 No Enough DPARTITION ROOM!
TSK2: MAX_MALLOC_SIZE: 0x300000
dP:0x100004:0x100
EMB:0x100008:0x10 EMB:0x100008:0x20 EMB:0x100008:0x30 EMB:0x100008:0x40 EMB:0x100008:0x50 EMB:0x100008:0x60 EMB:0x100008:0x70 EMB:0x100008:0x80 EMB:0x100008:0x90 EMB:0x100008:0xa0 EMB:0x100008:0xb0 EMB:0x100008:0xc0 EMB:0x100008:0xd0 EMB:0x100008:0xe0 EMB:0x100008:0xf0 No Enough DPARTITION ROOM!
EMB_again:0x100008:0x10
efBlock: 9
X:0x100004:36
efBlock: 9
efpartHead: 0x100004 efpartTail: 0x100094
xHandler : 0x100004
X1:0x100008 X2:0x100008
Y0:0x100008 Y1:0x10002c Y2:0x100050 Y3:0x100074
X3:0x100008

*****
* Idle Idle ! *
*****
```

结束，进入死循环。

### ● 物理内存检测：

实现方法与老师说的类似，向内存中写入 0x6666，再读出该地址内容，若没有改变说明该内存有效，否则达到了最大内存。

### ● 动态内存分区：

内存检测后设置动态内存空间，后面分配的字节都来源于此段；free 后将该段字节还给内存空间，下次分配时可以使用。

分配算法使用 4 字节对齐，不足则 4 字节补齐。在每次动态内存分配时多加一个块，用于保存分配的块数，这样可以更好的利用碎片空间，分配时从起始地址开始检查块的值是否为 0，若都为 0，则分配成功；内存回收时传入第二个块的起始位置，找到地址，从这个地址的连续块置为 0，回收成功。

### ● 等大小固定分区：

每部分大小相同，按顺序分配，根据接口中给出每块大小，块数计算总空间大小，分配相应空间。

实现时设置一个块记录该段内存块是否被使用，其余算法与动态分配类似。