**操作系统课程设计实验报告**

——实验三：VMM实验

负责人姓名：石浩然

学号：14061122

日期：2016.5.7

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 詹子豪 | 14061137 | 实验一 |
| 2 | 于福洋 | 14061134 | 实验二 |
| 3 | 于建勋 | 14061125 | 实验三 |
| 4 | 石浩然 | 14061122 | 实验四 |

目录

[1.实验目的 4](#_Toc446001831)

[2.需求说明 4](#_Toc446001832)

[2.1基本要求 4](#_Toc446001833)

[2.2 提高要求 4](#_Toc446001834)

[2.3 完成情况 5](#_Toc446001835)

[3.设计说明 6](#_Toc446001836)

[3.1 程序流程图 6](#_Toc446001837)

[3.2基本要求实现说明 6](#_Toc446001838)

[3.3 提高要求实现说明 9](#_Toc446001839)

[4.收获和感想 11](#_Toc446001840)

# 1.实验目的

1.了解Linux的内存管理机制。

2.掌握页式虚拟存储技术，理解虚拟地址到实地址的定位过程。

3.掌握“最不频繁使用淘汰算法”，即LFU页面淘汰算法。

# 2.需求说明

## 2.1基本要求

## 通过本实验，要求学生能够了解Linux系统下页式存储管理机制，并实现一个简 单的虚存管理模拟程序 具体要求如下：

## 设计并实现一个虚存管理模拟程序，模拟一个单道程序的页式存储管理，用一个一维数组模拟实存空间，用一个 文本文件模拟辅存空间。

## 建立一个一级页表。

## 程序中使用一个函数do\_request()随机产生访存请求，访存操作包括读取、 写入、执行三种类型 。

## 实现一个函数do\_response()响应访存请求，完成虚地址到实地址的定位及 读/写/执行操作，同时判断并处理缺页中断。

## 实现LFU页面淘汰算法。

## 2.2 提高要求

1. 建立一个多级页表。

2. 实现多道程序的存储控制 。

3. 将do\_request()和do\_response()实现在不同进程中，通过进程间通信 （如FIFO）完成访存控制的模拟 。

4. 实现其它页面淘汰算法：如页面老化算法、最近最久未使用淘汰算法 （LRU）、最优算法（OPT）等。

## 2.3 完成情况

【简述实验完成过程】。完成了以下功能：

## 1.设计并实现一个虚存管理模拟程序，模拟一个单道程序的页式存储管理，用一个一维数组模拟实存空间，用一个 文本文件模拟辅存空间。

## 2.建立一个一级页表。

## 3.程序中使用一个函数do\_request()随机产生访存请求，访存操作包括读取、 写入、执行三种类型 。

## 4.实现一个函数do\_response()响应访存请求，完成虚地址到实地址的定位及 读/写/执行操作，同时判断并处理缺页中断。

## 5.实现LFU页面淘汰算法。

6. 建立一个多级页表。

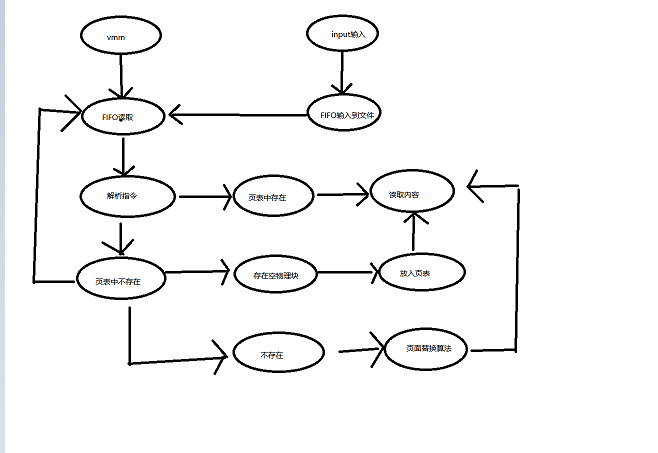
7. 实现多道程序的存储控制 。

8. 将do\_request()和do\_response()实现在不同进程中，通过进程间通信 （如FIFO）完成访存控制的模拟 。

9. 实现其它页面淘汰算法：如页面老化算法、最近最久未使用淘汰算法 （LRU）。

# 3.设计说明

## 3.1 程序流程图



## 3.2基本要求实现说明

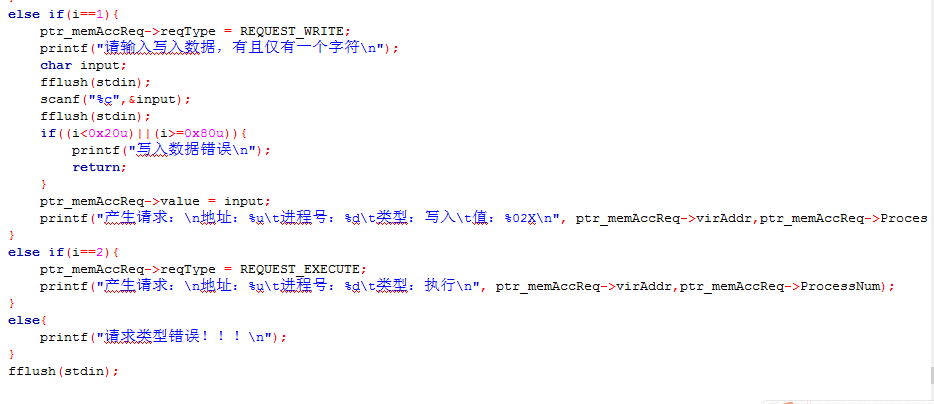
配以详细的代码和实现步骤截图。

（1）（2）（3）（4）已实现

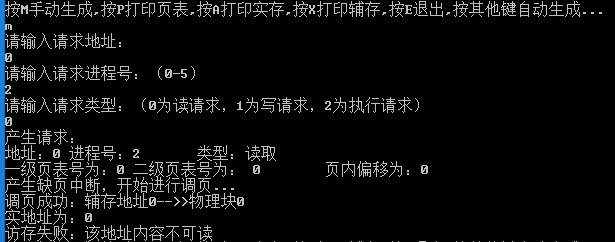
（5）手动输入指令

根据提高要求中多道程序的要求，在手动输入请求时，需要输入模拟的进程号，当请求为写入时，设置了一个判断条件保证输入的写入数据仅为一个字节



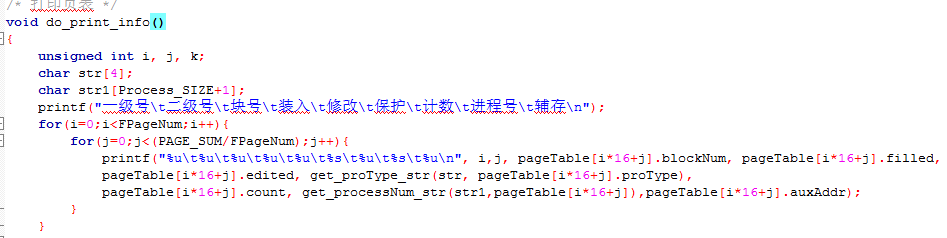


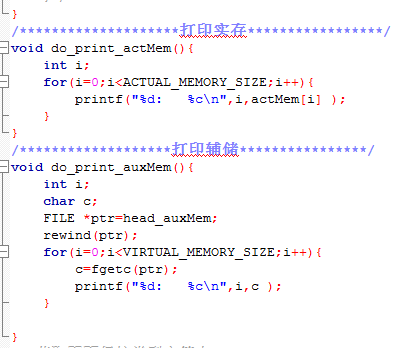
运行结果：



1. 打印

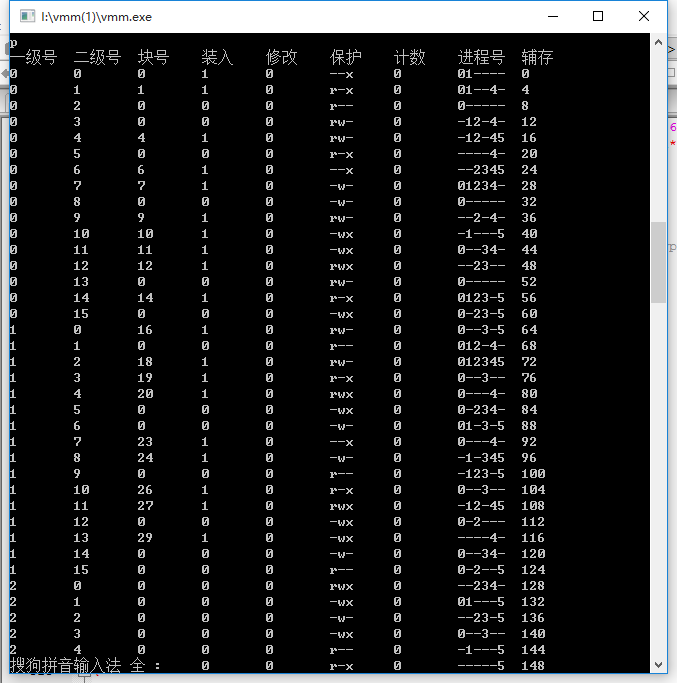
分为三个功能，打印页表，打印实存，打印辅存，后两项都是按照逐字节打印出来的，每行前一部分是地址，后一部分是地址所存储的数据，代码见下图



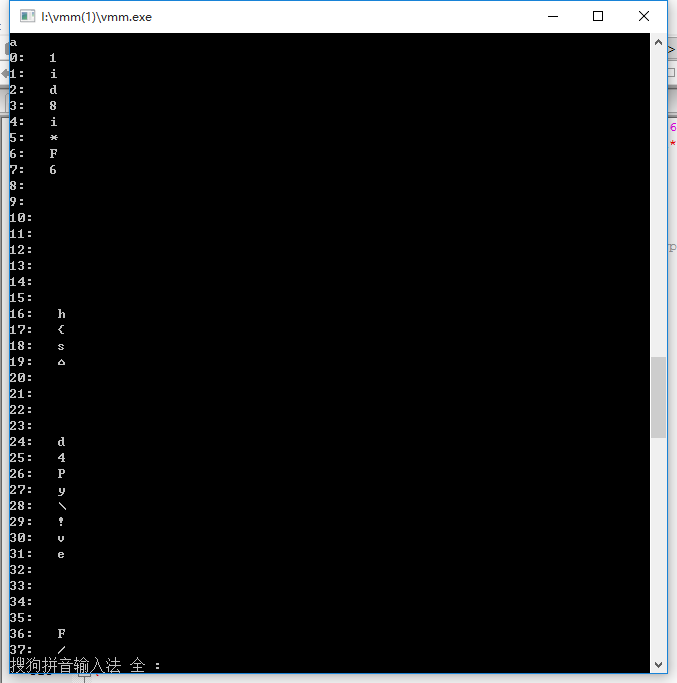


运行结果：

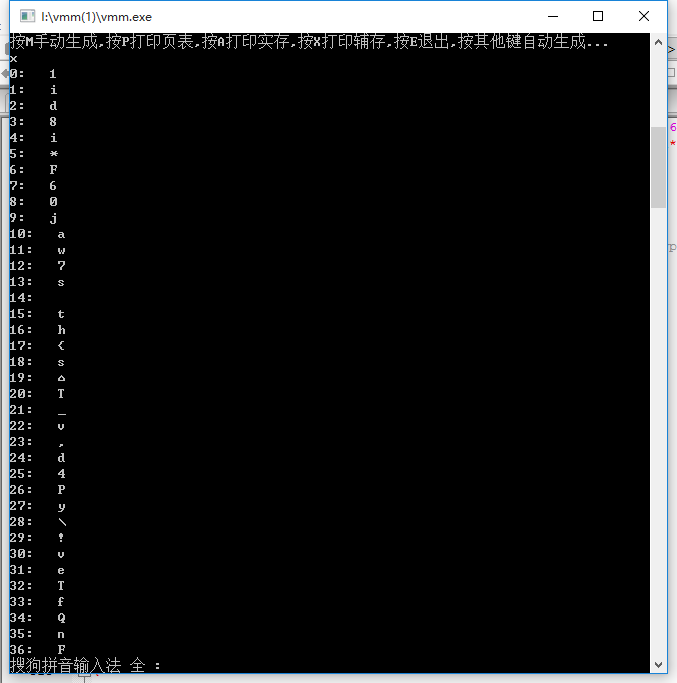
打印页表



打印实存

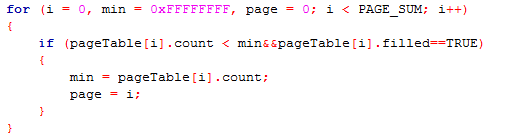


打印辅存



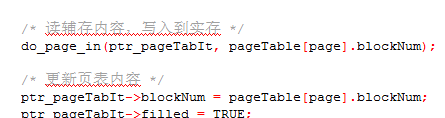
1. LFU算法：

源码中的LFU算法有一些问题，首先是该处



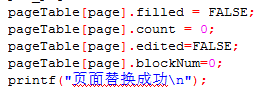
在源码中没有判断pageTable[i].filled，导致每次替换时都会替换第一次被装入的物理块，即使该物理块已经被装入其他的页表项，所以出现了同一个物理块被很多页表项同时装入的问题。

第二处问题是在这里



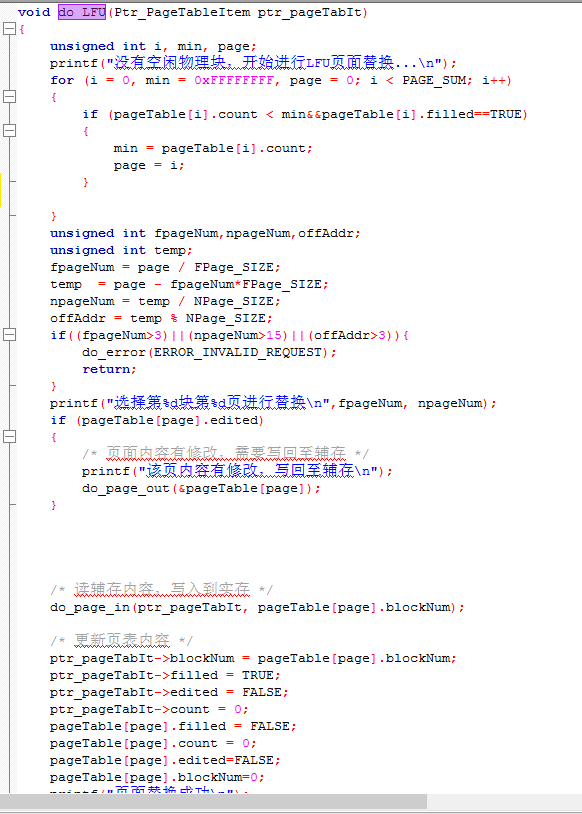
在写入时，源代码用的是page，这是页表项，而实际上这个操作是对物理块进行的操作，应该改为blockNum

第三处问题

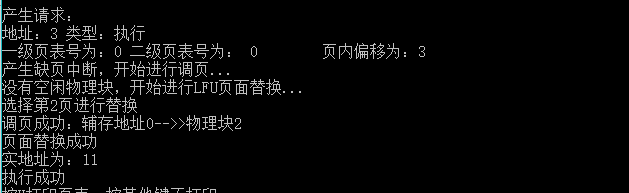


在源代码中，只有前两项，但我认为，当物理块被取出时，应该把页表项的edited和blockNum属性都置零。

源代码

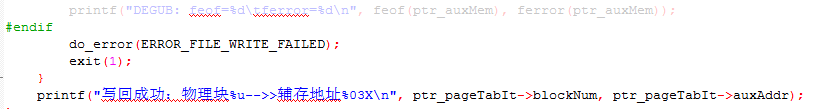


执行结果



1. 另外，在源代码中，还有另一处bug

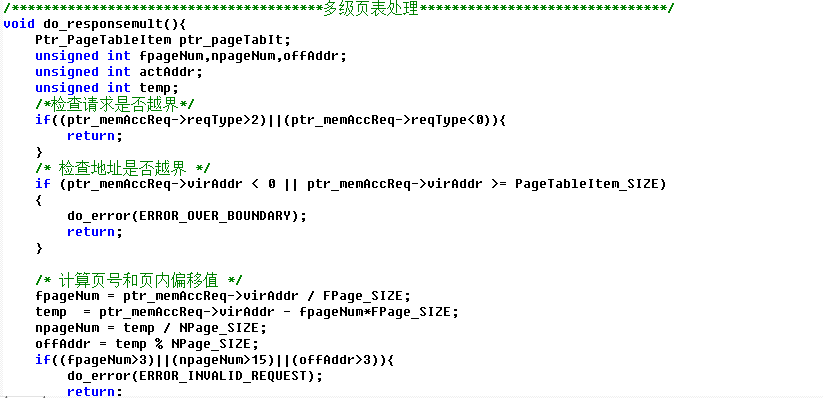
在提供的do\_page\_out函数中，最后输出结果时，把物理块号和辅存地址写反了，见图



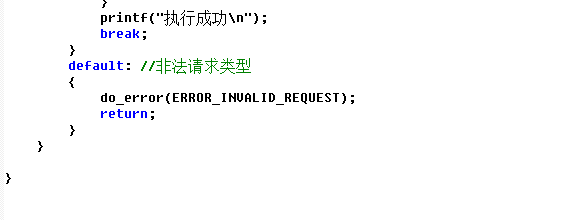
## 3.3 提高要求实现说明

配以详细的代码和实现步骤截图。

1. 多级页表







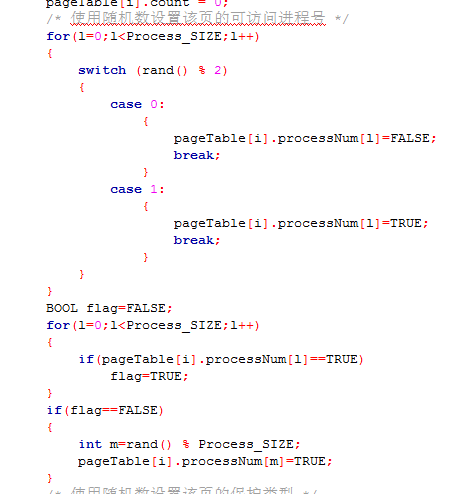
运行结果在下面提高要求中的体现。



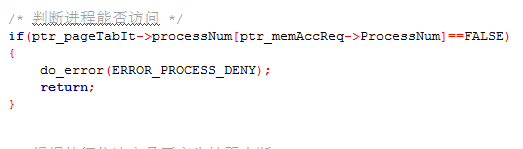
1. 多道程序的控制

设置总进程数为6，每个页表项随机设置几个进程可以访问，在访问页表项时，判断进程号与页表项的权限，如果没有权限则会输出错误信息。

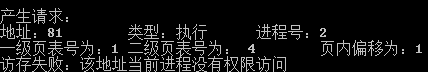
页表项初始化时的随机算法



判断权限

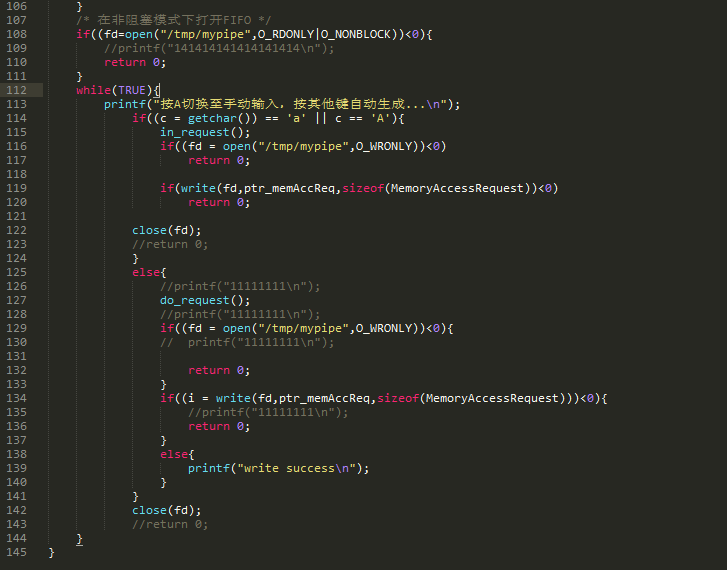
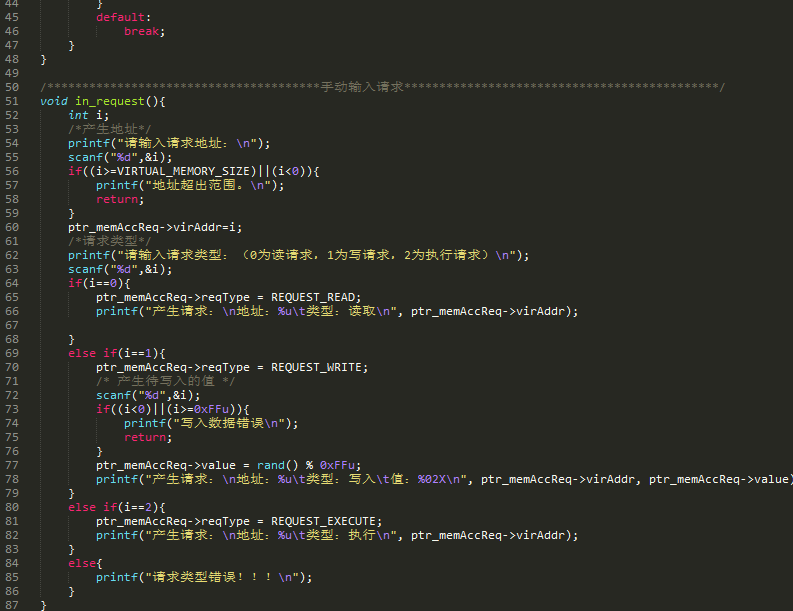
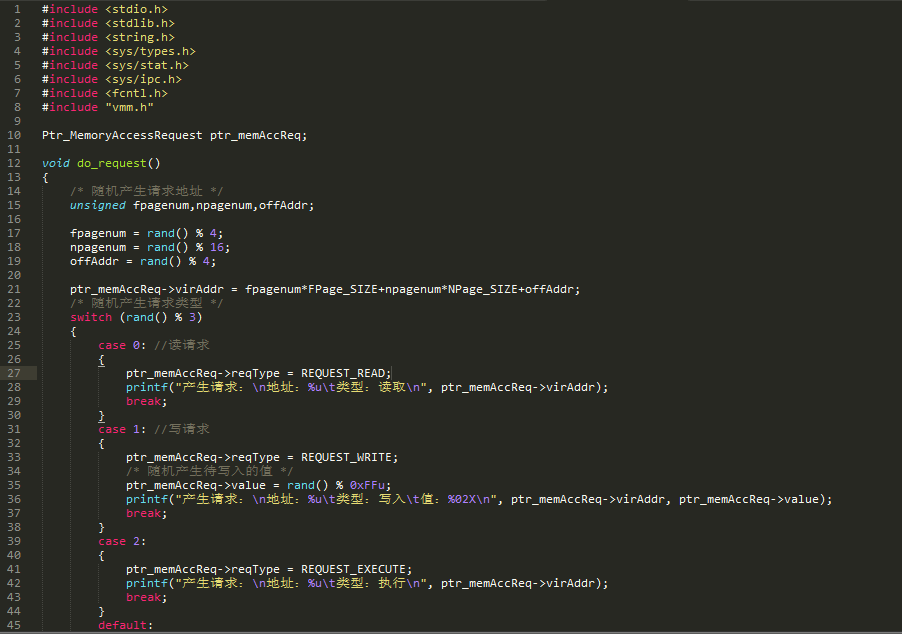


如果没有权限的结果

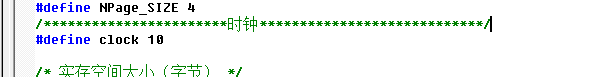


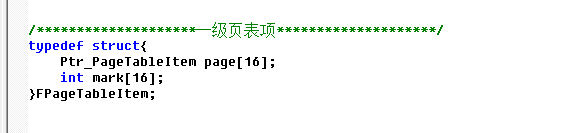
1. 实现FIFO通信

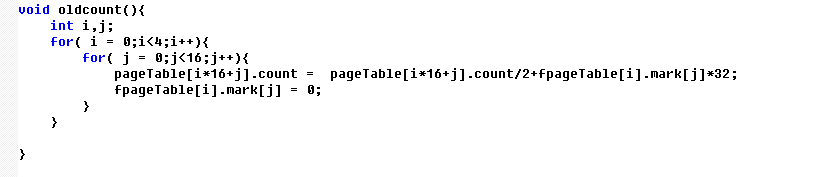
将do\_request()和do\_response()实现在不同进程中，通过进程间通信 （FIFO）完成访存控制的模拟 ：

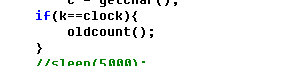


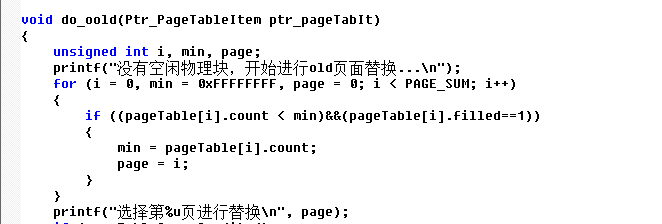
1. 页面老化算法





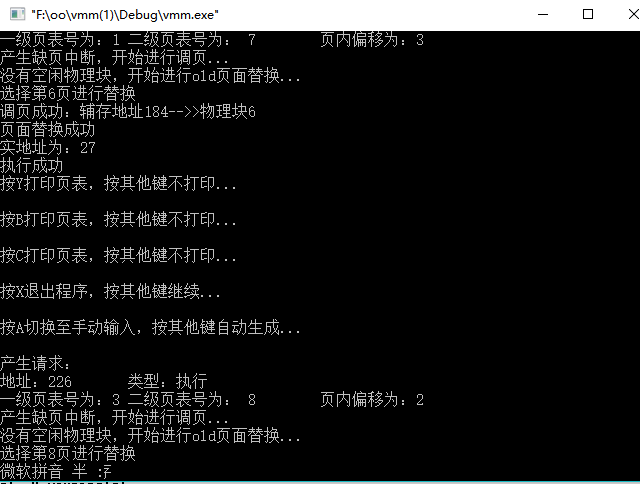




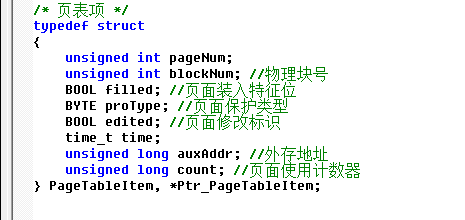


（在每个页面被装入或者替换时对以及也飙的mark置1，初始化为0）

运行结果



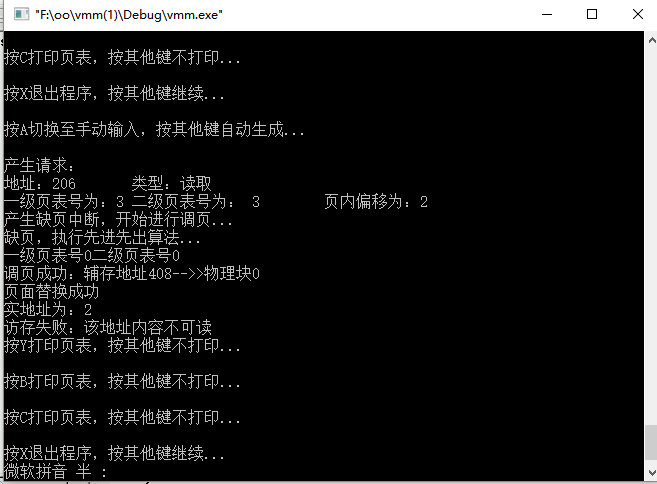
1. 先进先出算法



QQ截图20160508105438.png



运行结果



# 4.收获和感想

通过这次虚拟存储器管理的实验，我基本的了解了分页式存储管理和LFU调度算法原理，和操作的方法，还有许多方面不足，有待改进！

通过对提高要求的学习，我们还了解了多道程序对页表的访问，同时还更加加深了对于管道的理解。

而且我们还学到了在linux系统中编程的调试方法，而不是再像以前一样导出打印，这让我们以后的调试工作变得更加简单。

在这次试验中，我们小组成员协调配合。将任务分解，然后组合，经验互相分享，最后我们在加深了对于linux系统的了解。不仅学到了我嗯这次实验本应该学到的内容，也体会到了合作学习的乐趣。