**操作系统课程设计实验报告**

——实验三：虚拟内存

负责人姓名：周子杨

学号：14061126

日期：2016.5.6

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 王震 | 14061131 | 实验一 |
| 2 | 刘铮 | 14061139 | 实验二 |
| 3 | 周子杨 | 14061126 | 实验三 |
| 4 | 曾祥楷 | 14061138 | 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 4](#_Toc446001835)

[3.设计说明 5](#_Toc446001836)

[3.1 程序流程图 5](#_Toc446001837)

[3.2基本要求实现说明 5](#_Toc446001838)

[3.3 提高要求实现说明 5](#_Toc446001839)

[4.收获和感想 5](#_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

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

## 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

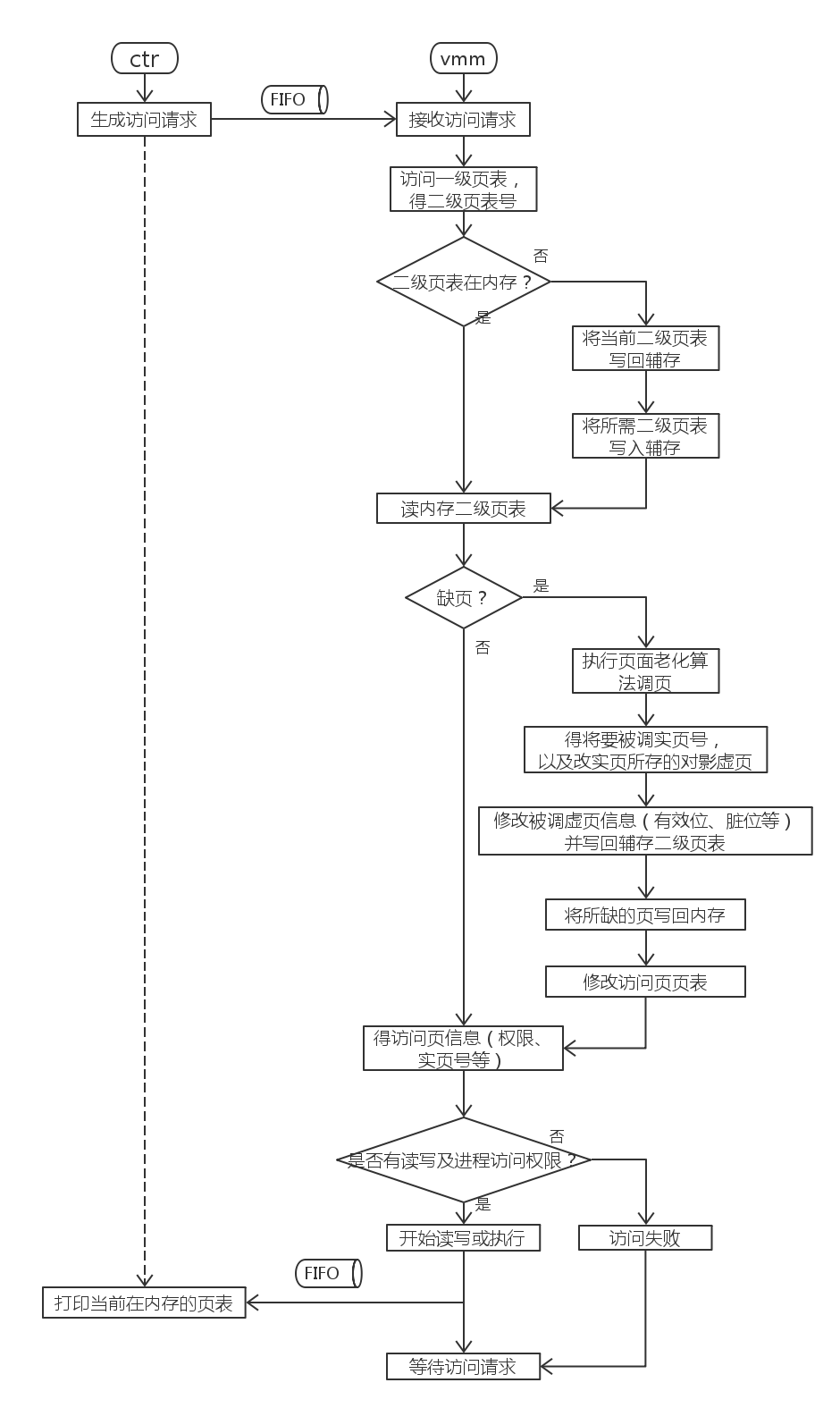
【简述实验完成过程】。完成了以下功能：

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. 建立了一个二级页表
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. 实现了页面老化算法

# 3.设计说明

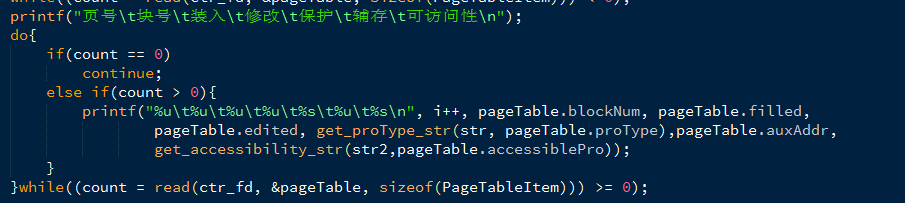
## 3.1 程序流程图

流程图

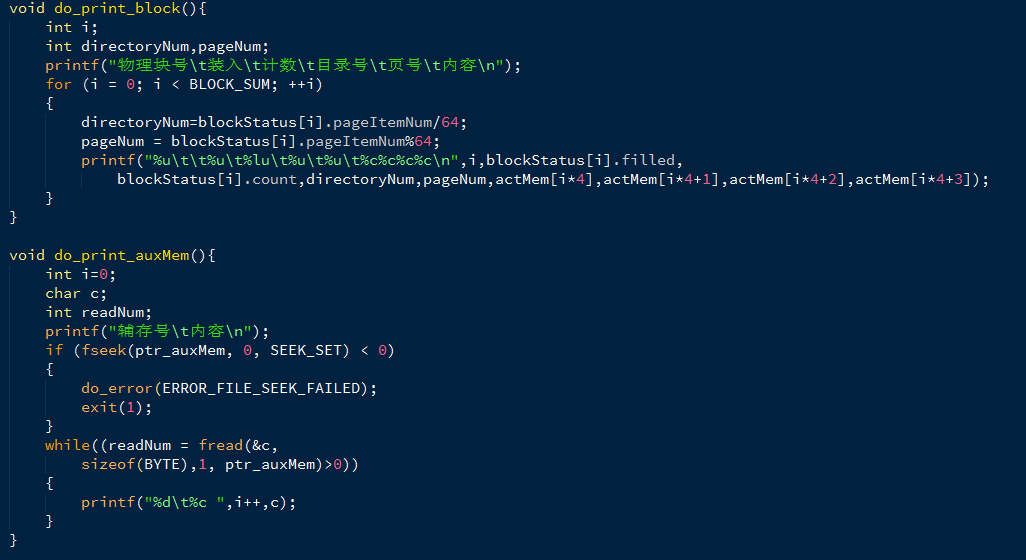


## 3.2基本要求实现说明

1，页表打印



2，实存虚存打印

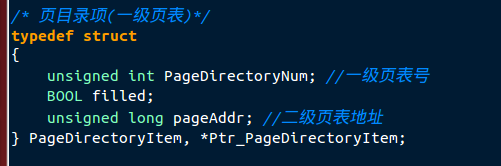


## 3.3提高要求实现说明

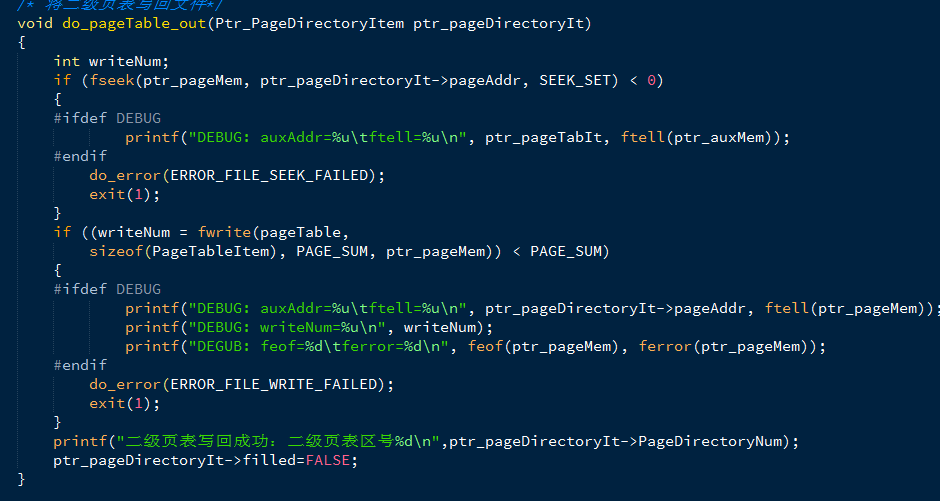
1. 多级页表

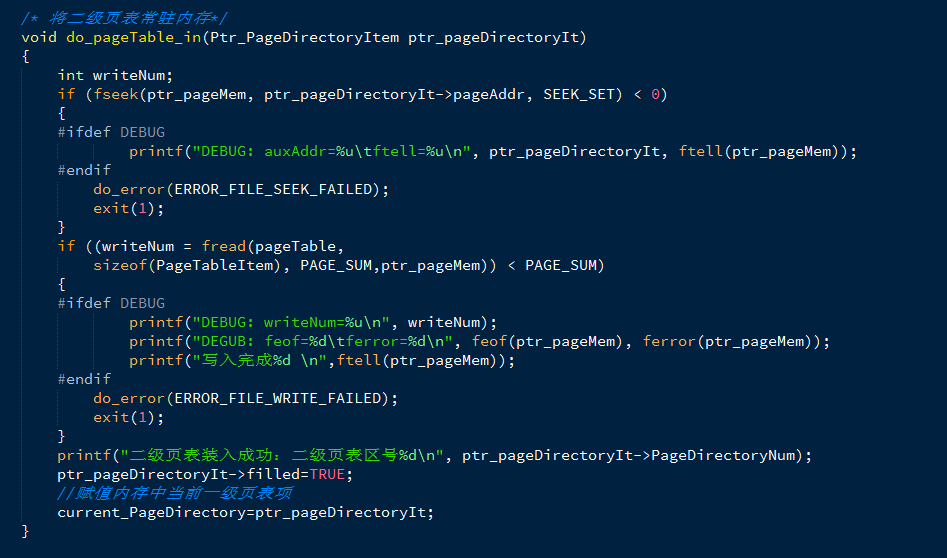
将磁盘分为不同的二级页表组，每组二级页表由一个一级页表指出。本次实验中，使用文件来模拟所有的二级页表项。一级页表结构中存有指出的二级页表组在文件中的首地址，一级页表号，是否被装入。

如下图：

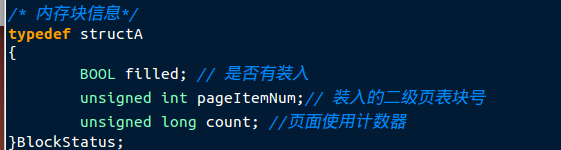


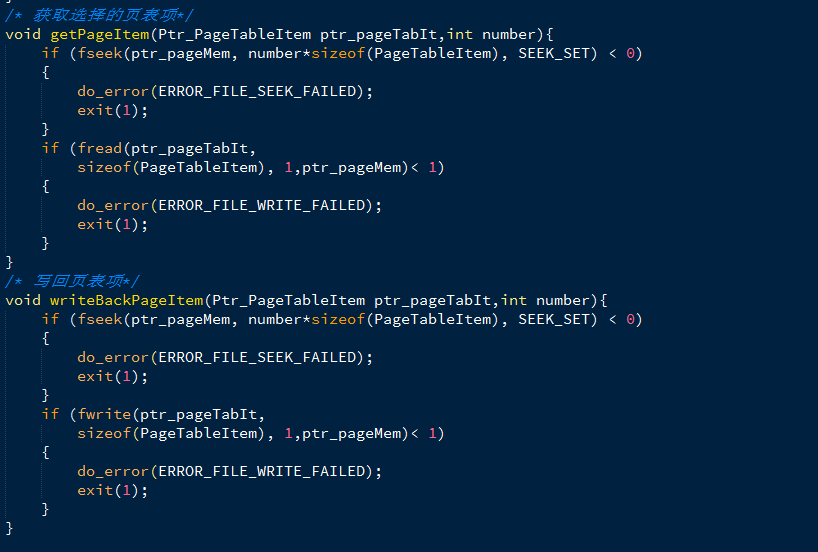
在使用过程中，要访问哪个二级页表项，先将改组二级页表装入内存中的pageTable 里，然后再进行访问。如果当前要访问的二级页表不再内存中，先将当前常驻内存的二级页表组写回文件。这两个步骤的方法封装为函数如下，





为了方便缺页中断时的处理，将blockStatus修改增加count，使得当需要访问存放在内存中的二级页表项的内容时，可以更简单。并提供写入写出函数如下。



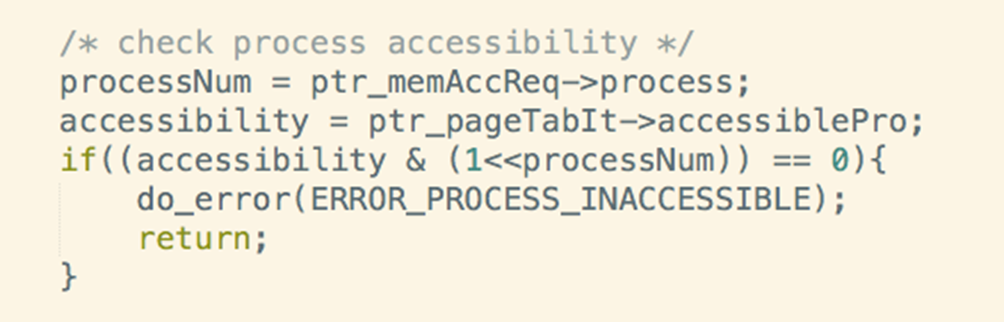


1. 多道程序

多道程序

正常情况下，要实现多个程序访问内存，每个程序都需要有自己一份独立的页表。在本次实验中，在页表项中加入一个标志位来代表此虚地址能否被这个程序访问，来模拟多个独立页表。

本实验中，在页表项结构中加入了accessiblePro来表示此页表项能被哪些程序访问。accessiblePro 是一个BYTE(char)，有8位，本实验代码中只支持模拟5个程序，因此只用了低5位。低5位每一位表示这个程序能否访问此页表项，1表示课访问，0表示不课访问。初始化时，给这个标志位随机赋一个0-31的值，表示给这个页表项随机赋予可访问的程序。在生成访问请求时，访问请求结构中添加一个process来表示是哪个程序访问，在do\_response中添加判读产生此请求的程序是否有权限访问此页表，有权限则继续执行，无权限

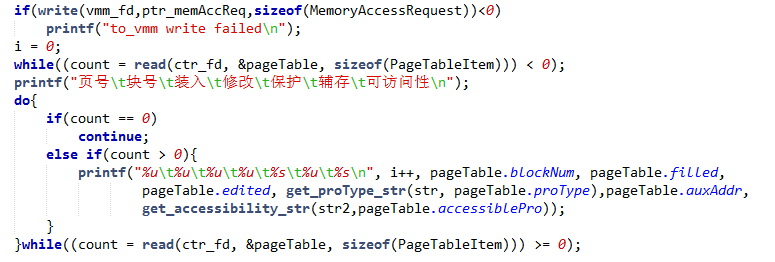


则返回。代码如下：

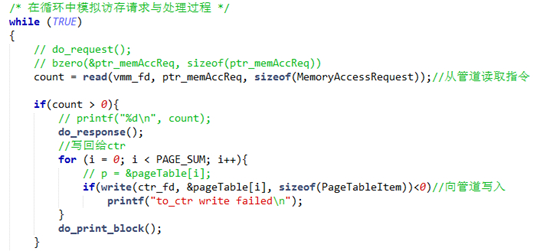
1. FIFO



在vmm.c中打开两个管道，分别用于从ctr向vmm传指令以及vmm向ctr传递处理之后的页表。



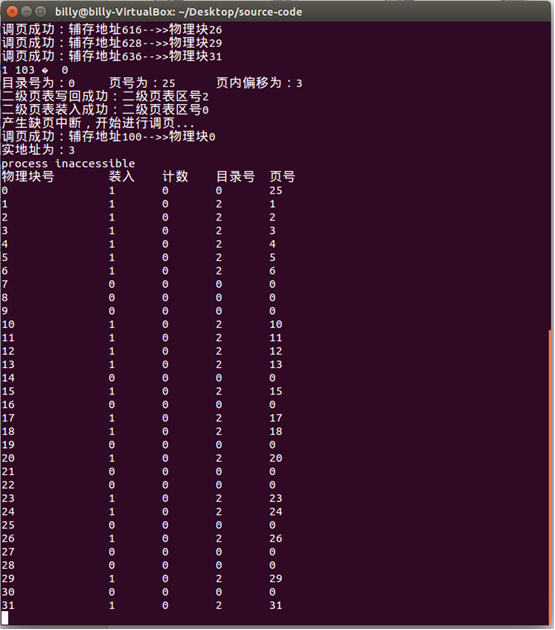
在ctr里用write函数向管道里写入指令。



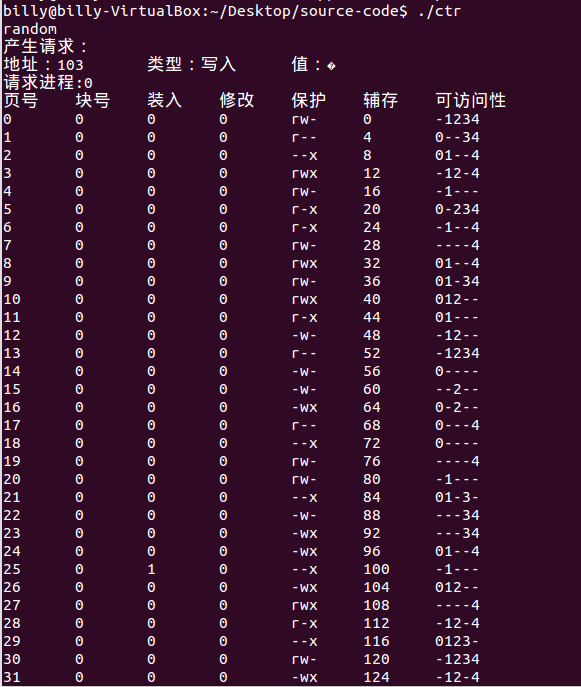
在ctr中用read函数从管道中读取指令，并且响应，并且向另一个管道中写入处理过之后的页表项，在ctr中用read函数从管道中读取页表项的信息并打印出来。

最终效果：

vmm

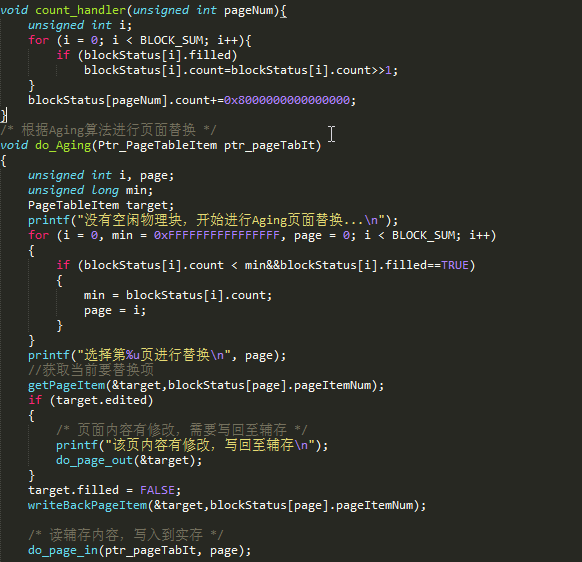


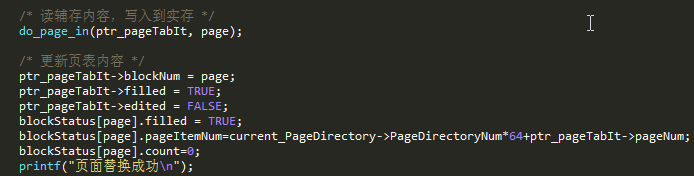
ctr



1. 页面老化算法

算法的基本原理也是利用count变量，但不是想LFU那样简单的每次访问一遍加一，而是在一个时钟周期内，每次将count变量右移一位，同时访问到的页表项最高位变为1。这样能够考虑到时间因素，因此更加客观。具体到我们这个实验，我们的方法是每访问一次页表，所有的页表项的count变量右移一位，然后对应的访问项最高位变为1，其他的变为0。在替换时，替换的是count值最小的页表项。代码具体截图如下：





其中count\_handler函数是处理count变量，而do\_Aging函数就是老化算法的实现。

# 4.收获和感想

收获：

通过这次虚拟内存实验，我们明白了内存调度是怎么实现的，明白了老化算法如何实现，以及进程之间是如何通信更加熟悉了FIFO的实现方法。我们小组的第三次合作，各个组员的协同工作更加熟练。通过相互之间的合作学习，互相帮助，最终完成了任务，提高了我们的teamwork。同时，这次的实验用到了好多老师上课的时候没有教过的知识，我们通过自己上网搜索答案，自己学习来解决问题，很好地锻炼了独立自主的学习能力。

感想：

自己体验到了linux系统内部究竟是如何调度内存块的，结合上个学期的计组实验的知识，更加体会到了操作系统的作用，以及初期系统设计人员为了性能优化所做出的努力，和前人的智慧。

会议记录

我们小组每周开一次会议

第一次会议

阅读源代码，确定每个人的工作，确定这次实验的难点重点进行重点学习。

第二次会议

交流完成的基本功能，确立提高功能中的工作分工，以及对于其中页面淘汰算法，多级页表的大致实现方向。

第三次会议

交流完成的提高要求，讨论一下具体实现了哪些功能，遇到了哪些调不出来的bug，进行集中处理，合并代码进行测试，讨论并完成实验报告。