**计算机与电子信息学院/人工智能学院**

《计算机操作系统实验》

课程实验报告

**实验名称： 操作系统大作业**

**专 业： 计算机科学与技术**

**姓 名： 成文涛**

**学 号： 19200312**

# 一、实验问题描述

详情见 “存储管理大作业.pdf”

作业要求写的较为复杂，理解难度较大。

因作者能力有限，无法完成其全部要求，设计的出的部分算法与结果将在下面展示。

心得处将写出不足与可以改进之处。

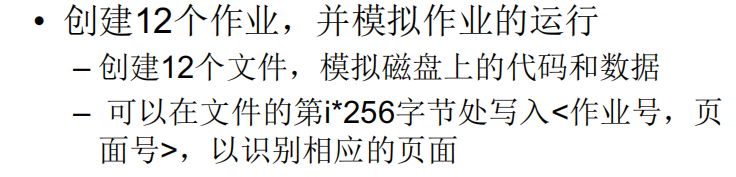
# 二、实验原理

1. //内存   页面大小 1<<8 字节    页面数: 1<<6 个  0号页框为页表
2. unsigned **char** \*mem=(unsigned **char**\*)malloc(1<<14);
3. //位图
4. **bool**  bitmap[1<<6]={0};
5. queue<**int**> Q\_FIFO;
6. queue<**int**> Q\_LRU;
7. //页框计数器
8. **int**   empty=1<<6;

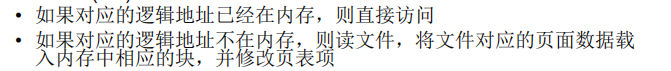
页面号占6位，作业号有12个，至少得占4位，不知道这是什么意思

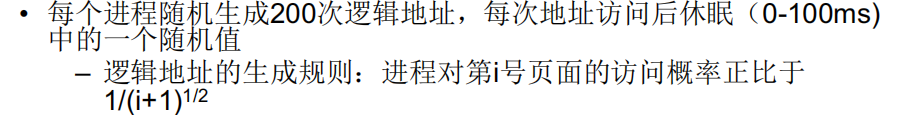
我打算：在i\*256字节处写入作业号，分配的内存是连续的，页面号可直接计算

第i个物理页面地址为： 256\*i ----256\*i+255



不清楚文件的意思，硬盘的大小设置为 (1<<10)\*（1<<6）\*(1<<8)，即为内存的1024倍。





=1

1. **double** ans=0;
2. **for**(**int** i=1;i<=(1<<16);i++)
3. {
4. ans+=(1/sqrt(i+1));
5. }

得 正比例系数k约为0.002

本想以这种算法来返回逻辑地址，但事实上总是因为数量过大，每个的概率过小而得不出结果（因为正比列系数约为0.002）：

1. **for**(**int** i=1;i<=(1<<16);i++)
2. {
3. **if**((rand)<=(0.2/sqrt(i+1)))
4. **return** i;
5. }

想到一种类似于“**素数筛**”的方法，最小的概率约为：0.00390622

占据内存大小为：Ceil(P\*1000)，得出总共的内存大小为：543517

测试了一下，效果还不错（**但是结果是近似的，且依靠随机数取值并不稳定**）

1. **void** addr\_init()
2. {
3. **int** pos=0;
4. **int** area=0;
5. **for**(**int** i=1;i<=(1<<16);i++)
6. {
7. area=ceil(1000/sqrt(i+1));
8. **while**(area--)
9. {
10. addr[pos++]=i;
11. }
12. }
13. }

200次的随机测验结果如下：

5 ,61 ,6 ,82 ,3 ,64 ,174 ,4 ,3 ,25 ,118 ,36 ,185 ,48 ,2 ,1 ,77 ,7 ,3 ,189 ,13 ,269 ,62 ,20 ,142 ,158 ,17 ,2 ,178 ,254 ,222 ,25 ,115 ,304 ,146 ,120 ,82 ,198 ,109 ,241 ,29 ,251 ,250 ,252 ,30 ,63 ,106 ,262 ,39 ,129 ,285 ,55 ,104 ,4 ,1 ,116 ,134 ,148 ,179 ,30 ,5 ,36 ,284 ,132 ,22 ,65 ,10 ,266 ,175 ,38 ,10 ,306 ,3 ,48 ,33 ,258 ,111 ,305 ,179 ,15 ,18 ,246 ,17 ,102 ,243 ,132 ,121 ,5 ,259 ,244 ,15 ,20 ,73 ,8 ,15 ,153 ,3 ,162 ,133 ,17 ,265 ,244 ,263 ,18 ,53 ,288 ,238 ,20 ,36 ,87 ,143 ,208 ,18 ,234 ,2 ,103 ,95 ,177 ,6 ,29 ,288 ,14 ,22 ,204 ,27 ,69 ,69 ,195 ,17 ,195 ,144 ,25 ,154 ,227 ,22 ,236 ,23 ,2 ,5 ,213 ,118 ,168 ,12 ,106 ,68 ,68 ,9 ,224 ,56 ,269 ,12 ,57 ,37 ,26 ,274 ,15 ,58 ,199 ,82 ,283 ,187 ,82 ,277 ,8 ,19 ,270 ,5 ,9 ,127 ,168 ,114 ,276 ,11 ,236 ,215 ,192 ,5 ,56 ,37 ,2 ,257 ,11 ,19 ,52 ,74 ,185 ,143 ,111 ,42 ,5 ,221 ,256 ,282 ,103 ,92 ,300 ,14 ,31 ,43 ,143

**页表的分配：**

如何在已经分配的内存里创建一个页表：

创建一个简化的页表：

逻辑地址（16位）--->物理地址（6位）的映射，不考虑页内偏移量等。

页表的基址：0--255

每一项占4字节，共64项：

|  |  |
| --- | --- |
| 前16位：逻辑地址 | 后16位：物理地址 |

基本思想见下：

1. 找页表
2. 1.在内存:
3. 根据逻辑地址返回页框地址
4. 2.不在内存：
5. 1.没满 找个空的页面，返回页框地址
6. 2.满了，根据替换算法替换，返回页面地址

修改页表是要注意位运算（截断与拼接）。

1. mem[(64-empty)\*4+0]=(logic\_addr>>8);
2. mem[(64-empty)\*4+0]=(logic\_addr);
3. mem[(64-empty)\*4+2]=(i>>8);
4. mem[(64-empty)\*4+3]=i;
5. **int** add(**int** logic\_addr，int replaced)
6. {
7. If(没有替换) 找个空闲页表项填入
8. Else 插入被替换的页表项中
9. }

FIFO算法：

一开始是想在 **int[1<<6]** 这样的数据结构来模拟**队列**式的结构，具体算法在FIFO.cpp中，算法较为简陋且效率低下，故不采用。

采用 **queue**队列的数据结构，可以同时实现 FIFO和LRU算法。

FIFO：如果命中：不做任何操作；否则：满了：push并pop，；未满：push

LRU：如果命中：push并删除中间的重复值；否则：满了push并pop 未满push

可见FIFO和LRU算法只在命中时候的操作不同

有push和pop操作的都要在页表的对应位置添加和删除（相关操作存于add函数中）

1. /\*命中了 修改LRU的位图\*/
2. Q\_LRU.push(logoc\_addr);
3. queue<**int**> tmp;
4. **while**(!Q\_LRU.empty())
5. {
6. **if**(Q\_LRU.front()==logic\_addr)
7. **continue**;
8. tmp.push(Q\_LRU.front());
9. Q\_LRU.pop();
10. }
11. Q\_LRU=tmp;
12. Q\_LRU.push(logic\_addr);

# 心得体会

作业要求所涉及的“文件”概念用 磁盘概念取代，即 内存是字节（16KB）,磁盘是字节（16MB）。要求中的每个作业的虚拟空间地址为14位，为方便表示，统一为16位（2字节）。

占用的内存、外存等没有实际的数据，仅仅在用位图和页表等方式表示占用。所以在页面替换和分配时，没有涉及到实质性的页面的数据操作。

因本人水平和时间等因素限制，在本次作业中没有实现多线程和并发的操作，这是比较羞愧的一点。但我猜测本次实验的重点应该为：FIFO和LRU页面替换算法，这两个我还是在代码中有所实现的。

啊啊啊啊啊，不想写这个了，太折磨了。明天还有网络编程的考试...