**操作系统课程设计实验报告**

——实验三：虚拟内存

负责人姓名：李康宇

学号：14061172

日期：2016.5.14

**19小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 岳明哲 | 14061173 | 实验4 |
| 2 | 李文涛 | 14061217 | 实验1 |
| 3 | 李康宇 | 14061172 | 实验3 |
| 4 | 张东鑫 | 14061153 | 实验2 |

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.深入了解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LRU，FIFO算法。

# 3.设计说明

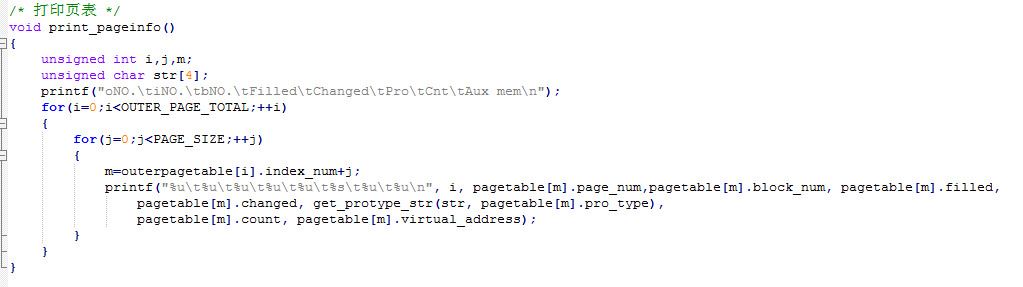
## 3.1 程序流程图

流程图

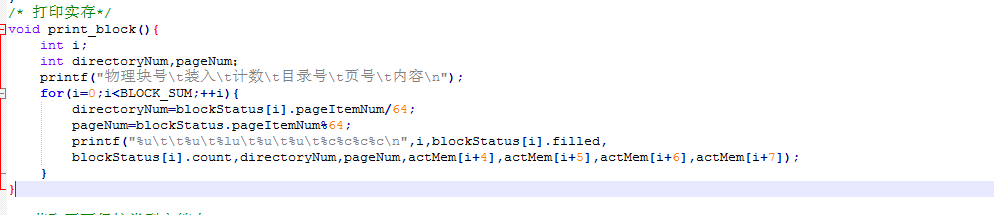


## 3.2基本要求实现说明

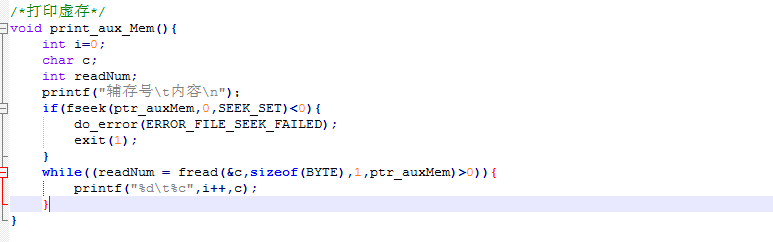
1，页表打印



2，打印实存（补）



3、打印虚存（补）



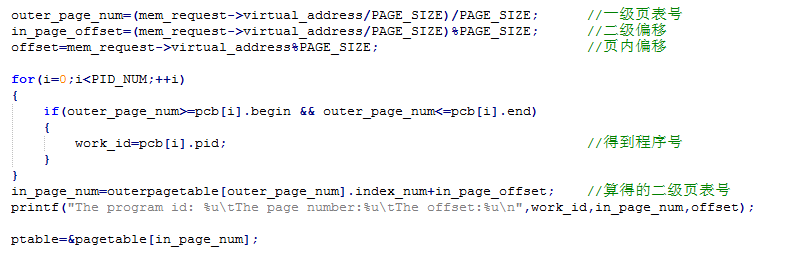
## 3.3提高要求实现说明

1. 多级页表

虚存共有64页，这是二级页表需要的大小，再将二级页表项的每四个分为一组，由一个一级页表指向，这样一级页表共有16个，各级页表都是由一结构体现，一级较为简单，只需对应到二级页表的映像，二级页表需要存放各类信息。

在得到一个逻辑地址后，将其分为三部分，分别为，一级页表号，对应的二级页表偏移，最后是页内偏移。

实现代码如下：



1. 多道程序

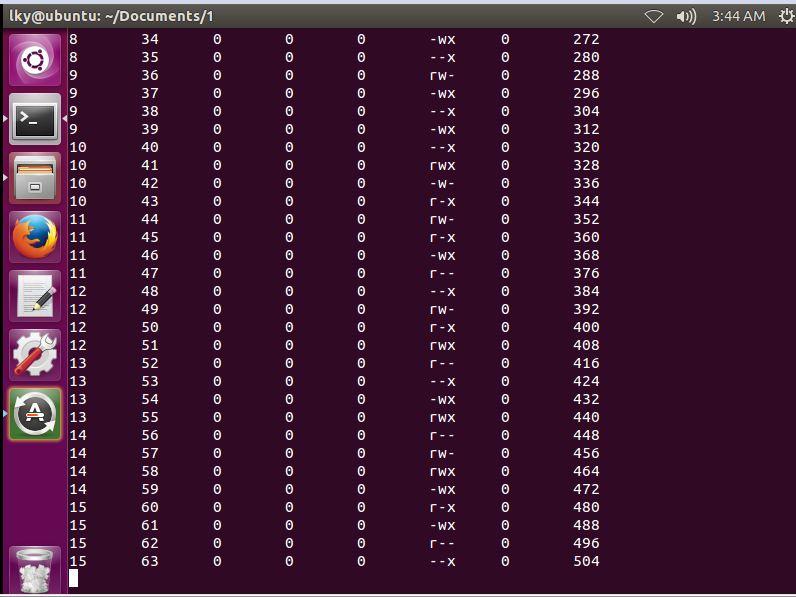
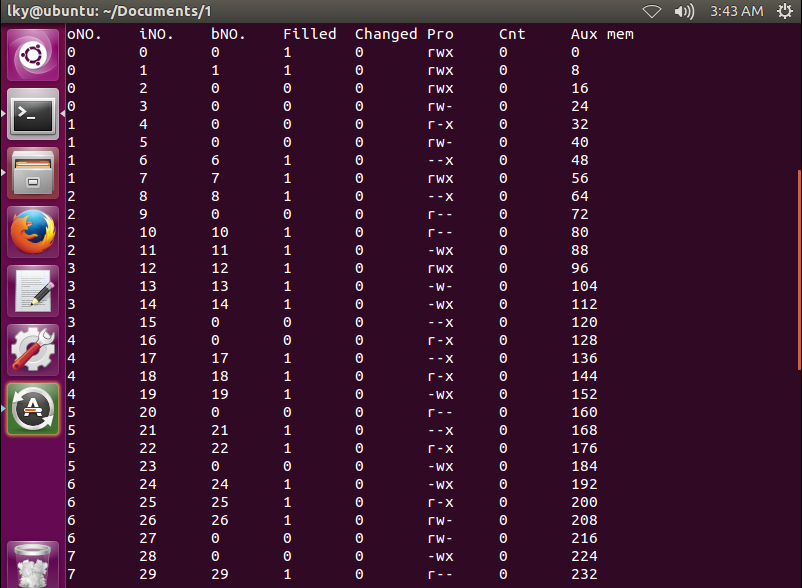
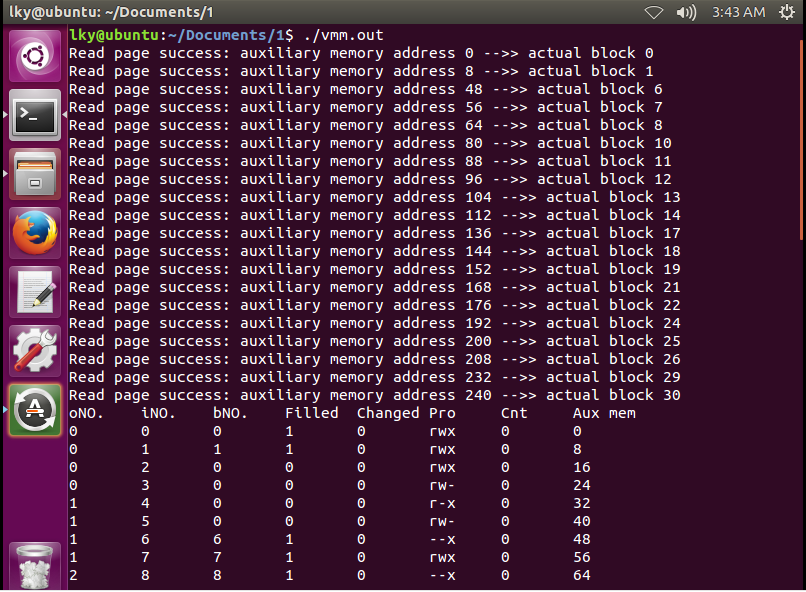
假设虚存文件里共存有两个程序，两个程序均分了虚存的大小，虚存共有64\*4，页面大小为4，续存共有64页，每个程序占了32页大小的空间。由于需要对每一个程序建立其对应的页表，我采用的是两级页表的形式，对于每个程序来说一级页表共需8个，为了简化操作，我将一级页表连在了一起，故一级页表的标识为0~15。

1. 进程间通信

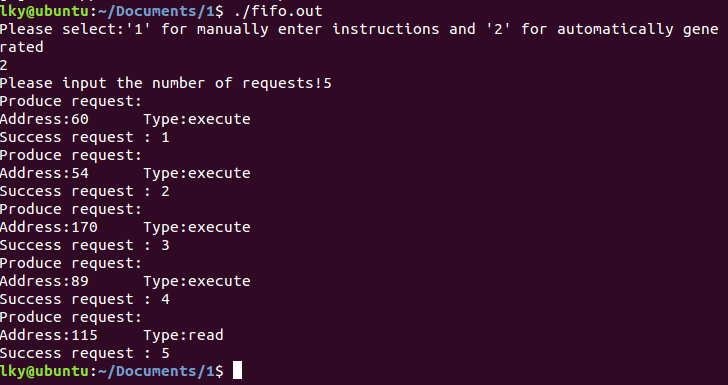
进程间通信采用的是FIFO机制，通过FIFO实现do\_request与do\_response之间的通信，将do\_request写在fifo.c，在fifo.c里打开FIFO，按照用户的需求，产生一定数量的请求，并将产生的请求写入FIFO。响应程序do\_response()在vmm.c，通过在vmm.c里面打开FIFO，读出里面的数据进行处理，再进一步响应，这样就可以实现进程间的信息交流。

实现图：

Vmm.out



Fifo.out:（选择自动生成了5条访存指令）



1. LRU和FIFO页面置换算法

LRU：全局变量exec\_times记录了当前所有页面所执行的次数，二级页表的结构中设有unsigned int型的变量no\_use，把当前的exec\_times赋给no\_use，表示这个页面在第no\_use次被执行，因此，这个值越小表明越久没有用到，因为每次用到都会刷新no\_use的值，所以，每次需要替换页面时，找no\_use最小的页面替换。

FIFO：定义了一个数组Time，用来存放页面装入的先后次序，Time[0]总是存放最先放入的页面，每次需要替换页面时，只需替换Time[0]即可，替换后，将Time中的数据整体前移一位，把替换掉的页面记录冲掉。

# 4.收获和感想

1. 我更加熟悉了LINUX内存调度的实现，明白了页面置换算法的实现，进程间的通信，以及FIFO的实现方法这些知识。
2. 通过自己动手查资料，自学相关知识，提高了我的自学能力。
3. 增强了团队凝聚力，明白了团队合作的重要性。

会议记录

第一次会议

阅读源代码，搞清楚要实现的功能，分配任务。

第二次会议

互相交流，讨论。

第三次会议

互相交流，讨论。提出合理的建议，处理bug。