**操作系统课程设计实验报告**

——实验三：内存管理

负责人姓名：叶旭诚

学号：14061215

日期：2016.5.6

**小组成员及分工**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 叶旭诚 | 14061215 | 整体分工与策划，缺页调度算法的添加 |
| 2 | 刘润泽 | 14061120 | do\_response()与do\_request()的多进程实现 |
| 3 | 陈鸿超 | 14061216 | 基本要求的实现，二级页表以及多程序控制的添加 |
| 4 | 杨佳琦 | 14061124 | 基本要求的实现与测试 |

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 5](#_Toc446001836)

[3.1 程序流程图 6](#_Toc446001837)

[3.2基本要求实现说明 7](#_Toc446001838)

[3.3 提高要求实现说明 10](#_Toc446001839)

**一、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.

4.熟悉并掌握二级页表的原理与使用

5.探索研究其他页面淘汰算法

**二、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、执行三种类型并支持请求命令的手动输入

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

1：完成了所有的基本要求，并将请求命令改成手动输入

2：建立了一个二级页表

3：完成多道程序控制页表

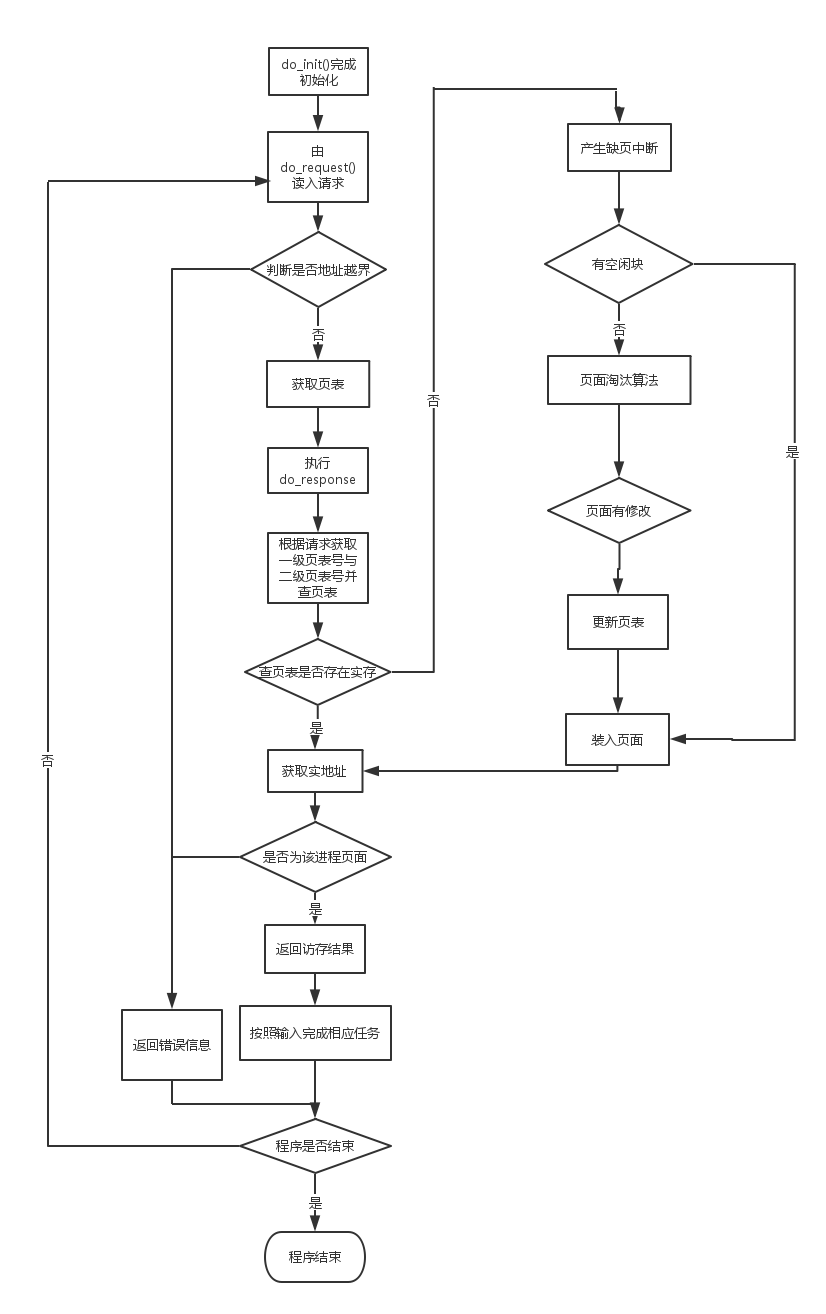
4：利用FIFO将do\_request()和do\_response()在不同的进程完成

5：实现了LRU算法以及页面老化算法

**三、设计说明**

**3.1 程序流程图**

（1）程序总体流程图

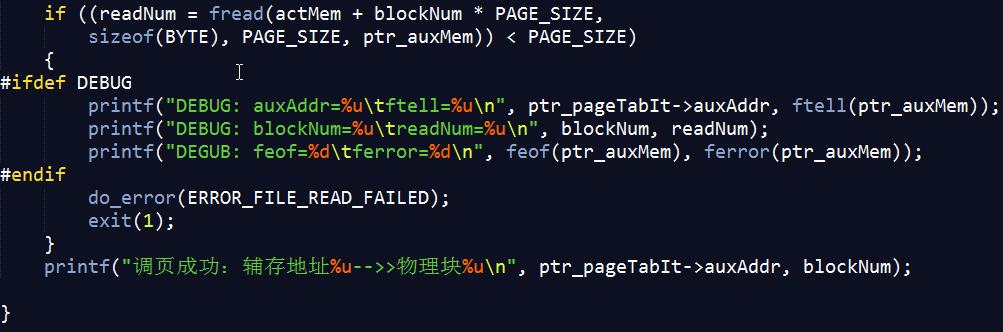


## 3.2基本要求实现说明

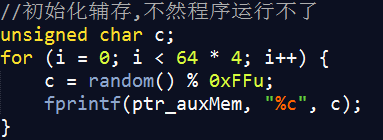
1:原代码BUG的修改

(1):do\_init()时未初始化辅存

影响:因为没有初始化辅存,所以辅存文件为空,在初始化实存时调用do\_page\_in函数,这里面freed函数读取空文件返回值是0而不是PAGE\_SIZE,所以会退出程序。



代码修改:



(2):初始化页表辅存地址错误

影响:会造成输入的地址与期盼的辅存不符

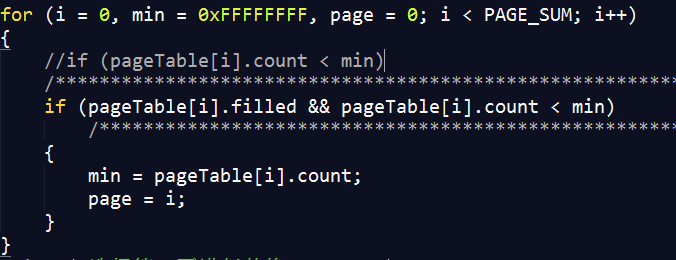
代码修改:



(3):FLU算法选择切换页表有误:

影响:源代码中是直接选择所有页表项中使用次数最少的，这样很容易 就选到那些并没有分配实存块的页表，这明显错了。所以应该选择被分配 实存块的使用次数最少的页表项。

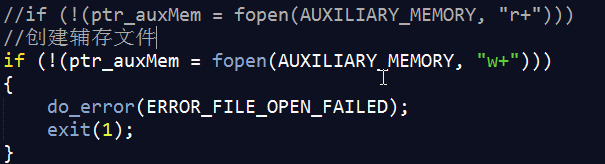
代码修改:



(4):程序一开始调用的fopen()函数参数有误

影响:使用参数”r+”是打开一个已存在的文件,而程序一开始并没有辅存文件存在，打开失败，程序无法运行。所以要改成”w+”，创建一个文件，如果文件已存在就删掉旧的再创新的。

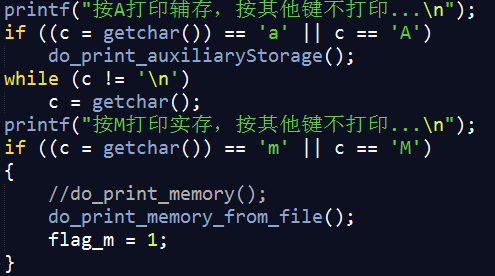
代码修改:



2:其他基本要求的实现

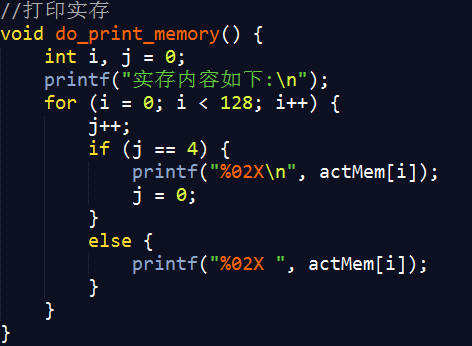
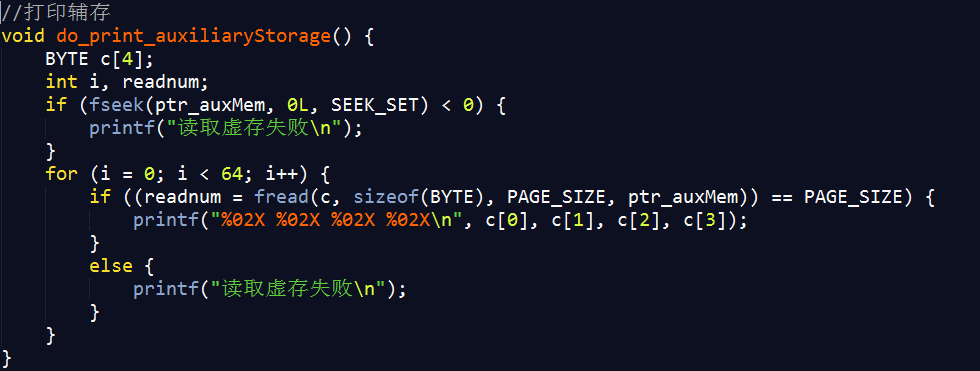
(1):辅存和实存的打印

首先和打印页表一样，由测试者决定要不要打印



根据测试者的输入决定是否调用打印函数,这里的函数是第一次写的一

进程时的打印函数，与后面修改后的多进程打印有区别



(2):手动输入请求

在处理请求函数里面读取输入而不是随机产生请求



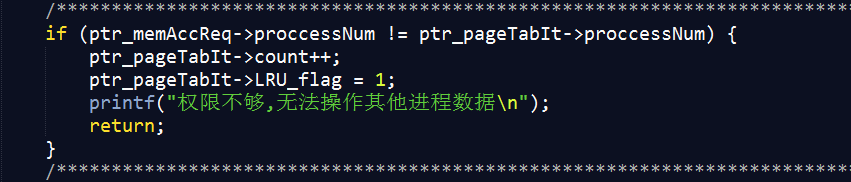
## 3.3 提高要求实现说明

1.实现多道程序控制：

首先为页表加一个参数，用于指定改页表所属的程序

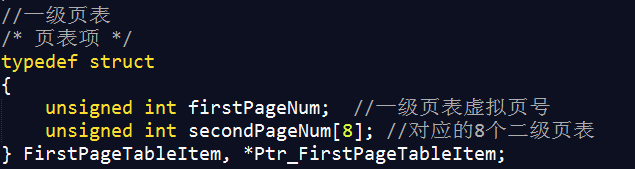


在处理请求时，都要先判断产生请求的程序是否拥有该页表的使用权，然后决定要不要执行相应操作。

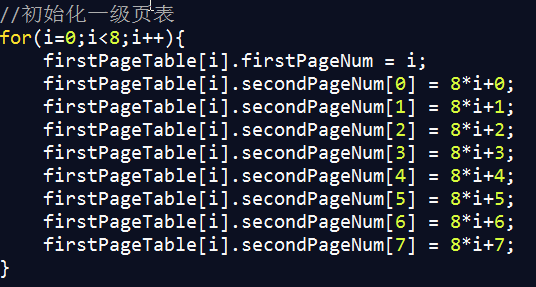


2.多级页表

先在头文件里创建一级页表的结构，原来的页表变为了二级页表。有8个一级页表项，每个一级页表对应8个二级页表。



初始化时初始化一级页表所对应的8个二级页表地址

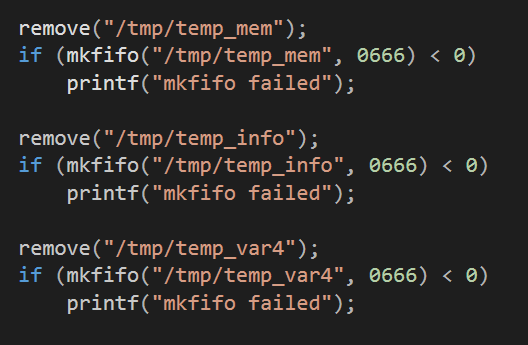


处理请求地址时，先将其转化成一级页表的页号和偏移量在找到对应的二级页表

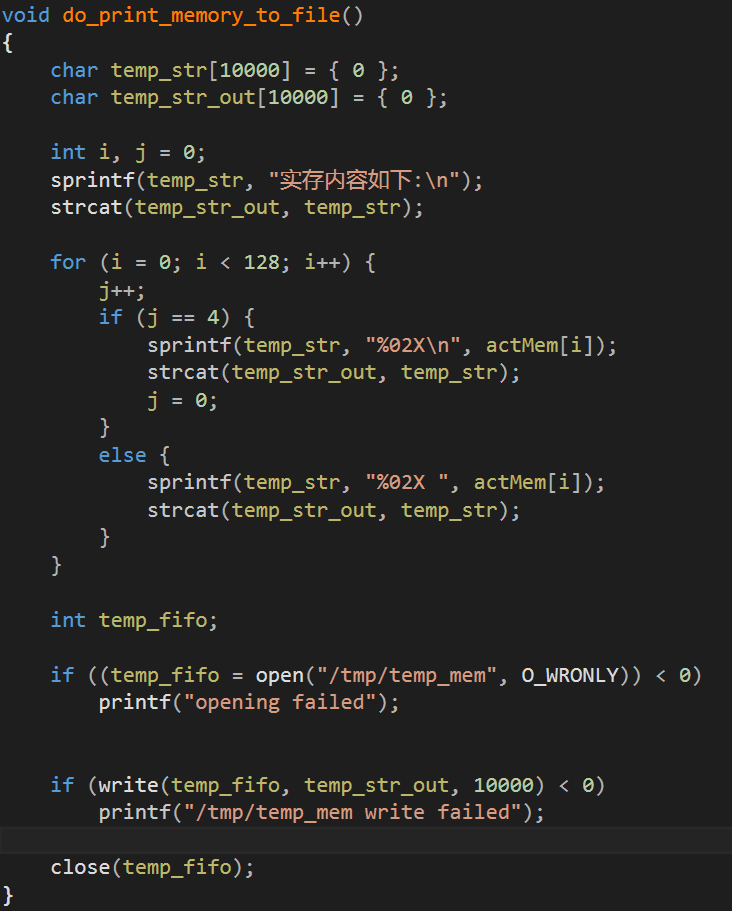
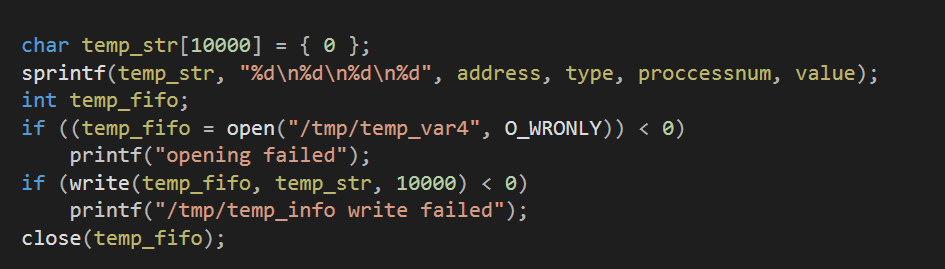
3：通过fifo通信将do\_request()和do\_response()实现在不同进程中

实现步骤如下：首先将do\_request()和do\_response()分别放在父进程与子进程中，父进程完成do\_request的输入功能，子进程完do\_response

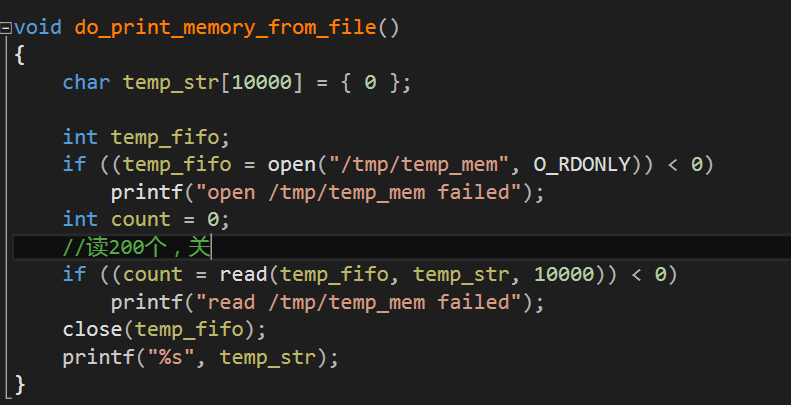
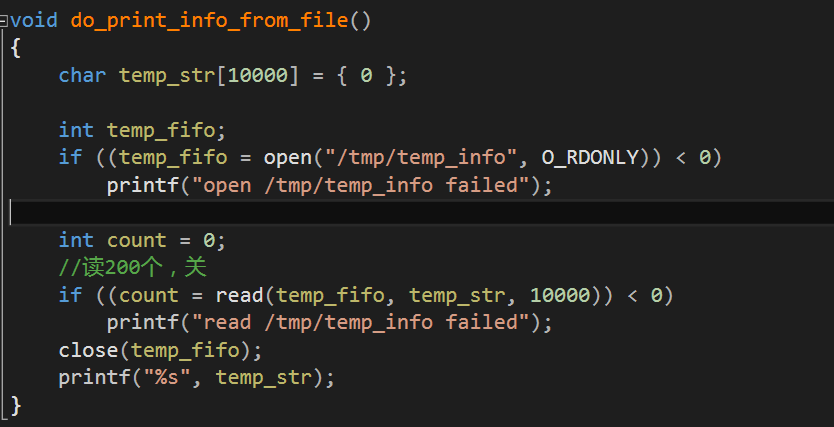
完成进程间通信：(1)创建命名管道：



(2)写：



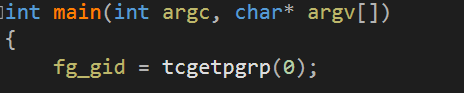
(3)读：



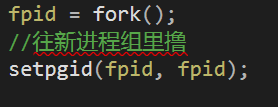
注释里写的读200请个不要在意，没有改

## 然后切换前台进程组，将子进程中的do\_response()输出的提示信息打印出来

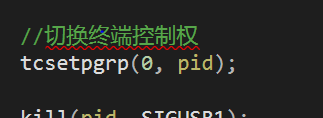
开始时，记录父进程所在进程组号



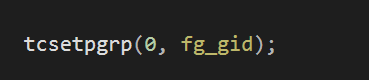
创建子进程时将其弄到新的进程组里，进程组号同其pid



更改前台进程组



子进程输出完毕，再次切换前台进程组

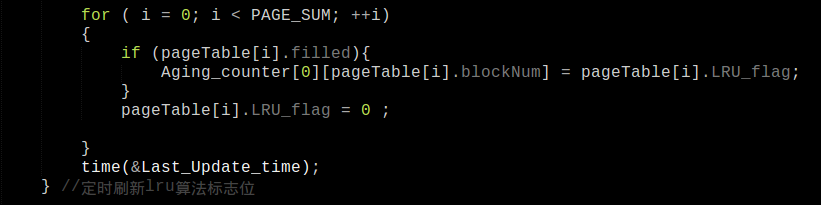


4：其他页面替换算法的实现

1. :LRU算法

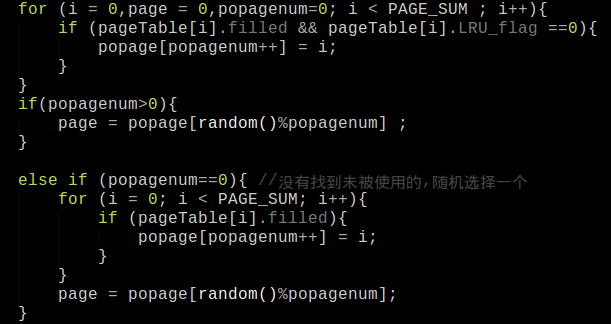
实现思路：LRU（Least recently used，最近最少使用）[算法](http://lib.csdn.net/base/31)根据数据的历史访问记录来进行淘汰数据，其核心思想是“如果数据最近被访问过，那么将来被访问的几率也更高”。

具体实现方法：首先在子进程中设置一个LRU的标志位刷新函数，每经过一段时间将所有的页面的LRU标志位全部置0



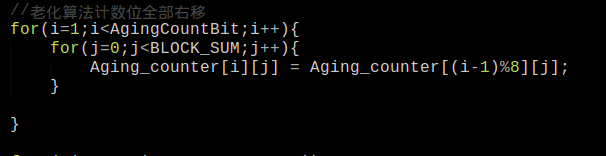
然后每次在发生缺页终端并调度时，首先查找有没有标志位为0的；如果有则加入可能采用页面的队列中，对于多个这种页面，随机选择一个页面进行替换

如果所有的标志位都为1，则在所有的标志位中随机选择一个符合要求的



1. :老化算法

老化算法没有找到详细的资料，在某个资料中找到了大致的的思路，老化算法的基本原理与LRU相似，有一点的拓展的地方是为每一个实存页面添加一个相应位数的计数器（程序中完成的是8位的计数器），每次刷新LRU标志位时将该页面的标志位填在计数器的第一位上，并将原来其他7位的标志位向右移一位。



在发生缺页中断时，首先检查对应页表的标志位，如果为0，再记录提取其对应的老化算法计数器中的所有位的和，选取所有标志位为0且计数器和最小的那一个页面进行替换。如果存在多个符合以上要求的，随机选取一个页面进行替换

如果所有页面的LRU标志位都为1，那么选取计数器和最小的那块页面。同样，若有多个符合该要求的页面，随机选择一块

