操作系统课程设计实验报告

——实验三：vmm实验

负责人姓名：王裕淞

学号：14061199

日期：2016.5.7

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 干逸清 |  | 测试 |
| 2 | 刘亮 |  | 调试 |
| 3 | 王裕淞 |  | 代码 |
| 4 | 朱谨 |  | 整理报告 |

目录

1.实验目的 4

2.需求说明 4

2.1基本要求 4

2.2 提高要求 4

2.3 完成情况 5

3.设计说明 6

3.1 程序流程图 6

3.2基本要求实现说明 6

3.3 提高要求实现说明 7

4.收获和感想 11

# 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()是现在不同进程中，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

基本要求由给出的代码实现，在此之外，完成了以下功能：

1.二级页表

2.老化算法

3.页面分组调换

4.多个进程在同一实存上的存储管理

# 3.设计说明

## 3.1 程序流程图

见proc.png

## 3.2基本要求实现说明

## 

## 

配以详细的代码和实现步骤截图。

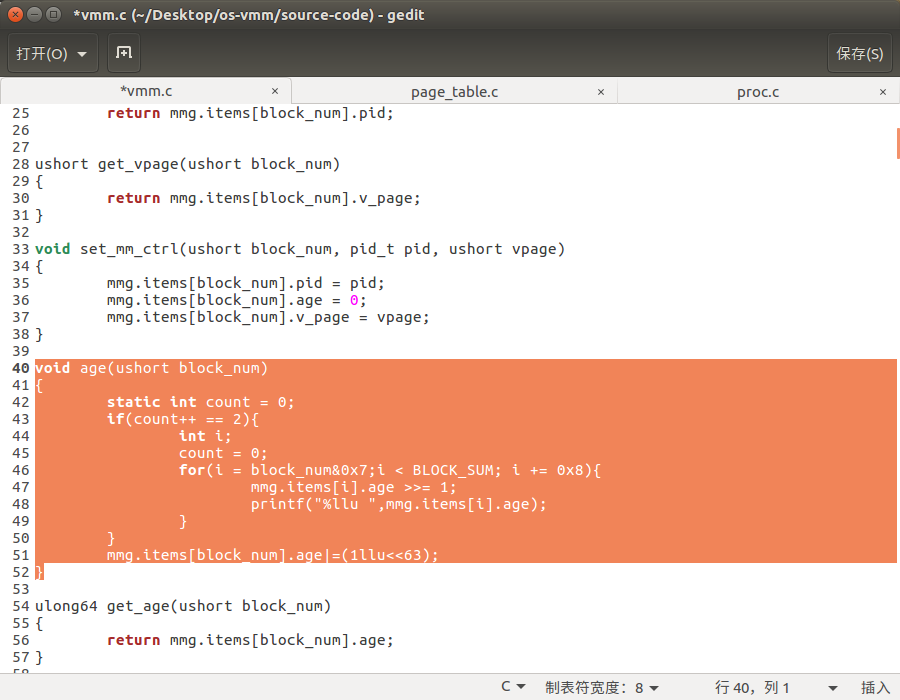
读、写、执行操作要做的事情几乎没有任何区别，如上图相关函数，获取对应进程的页表，查到目标地址所所在页所映射的表项中存放的装入情况，如果没装入则进行装入，如果需要换页则进行换页。然后判断响应权限进行操作。

然而由于原代码实现了LFU算法，后来实现了老化算法，这方面后面再说。

单道进程问题，因为实现了多进程，这里已经没办法写了。

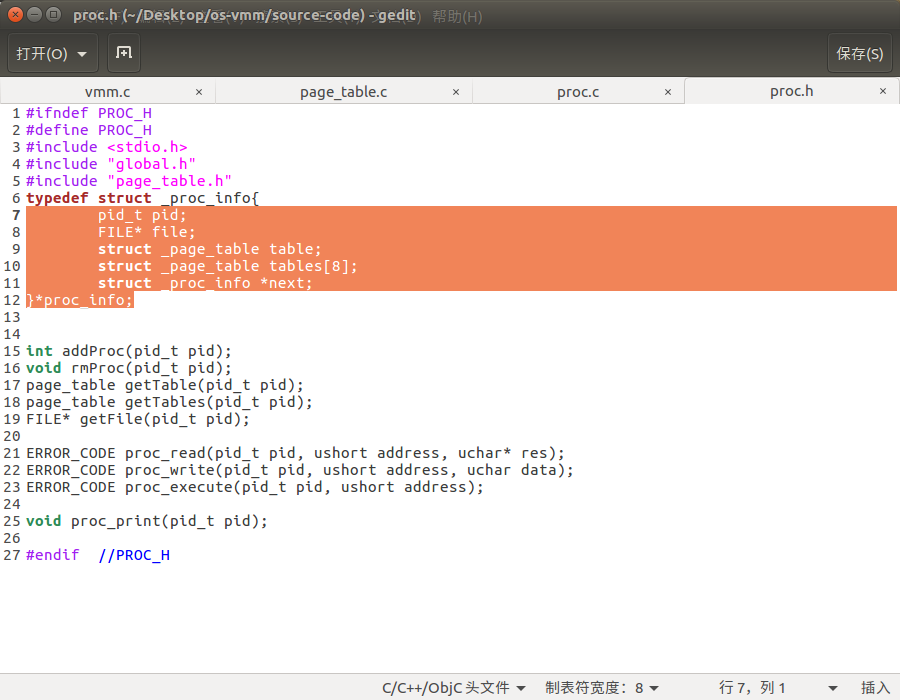
## 3.3 提高要求实现说明

配以详细的代码和实现步骤截图。

关于老化算法：

以上函数用于进行老化。老化函数进行两次，对目标组进行老化。老化函数会将指定页项刷新。

关于多道程序和多级页表：



用进程信息链表存放不同进程的进程ID、外存文件指针、一级页表、二级页表，并实现各个以ID为标记的对该数据结构的操作。只是一般的链表操作。

一级页表中有8个表项用来标记该项在二级页表中的位置，对应地址的高三位

二级页表中每个页表有8个表项用来标记该项在实存中的位置，对应地址的高4到高6位

二级页表的查找方式在基本要求说明的图中出现了（不可避免的出现过）

关于不同进程交互：



从名为“fifo”的明明管道中读取指令并送去解析，指令中包含所属进程号，因此可以作出区分。

任何进程都可以向该管道中写入信息，但只有正确格式的指令可以被处理。那么除此之外，实际上整个程序的实现是但进程方式实现的。

# 4.收获和感想

通过这次vmm实验……

通过这次实验，亲身体验了一下大致不具体的多级页表内存管理的流程，了解了一些调换算法的原理，以及有一个大的内存的重要性。

在我的这次代码实现中，包括一级页表、二级页表在内的进程信息没有放进实存，因为困难比较大。我尝试了几天，起初对原代码的修改也很大，扩大了页大小，把页表表项规范到位上，目的是为了做出8×8的二级页表出来，并把页表也全部放入实存中参加调换。就结果而言，数据结构能够工作得很好，但是调换流程比较尴尬。情况是，当产生一次调换时，由于某页（假设非页表页）需要换出，需要将其所在的二级页表中的项的装入位、修改位进行修改，如果不存在将需要调入，会产生心的调换，如此可能递归数个回合，而我在实现的过程中产生了死循环，并没有想到太好的解决办法，时间又紧张的情况下放弃了这个思路。

但修改后的数据结构保留了下来。如果用原来的32个实页进行老化算法，那么需要的老化计数的字段会比较长，产生调度的周期也比较长（必须等所有实页填满），因此对页进行摸8分组，模8同余的页之间可以相互调换，这样可以把老化计数字段的位数降下来（代码中没有体现，实际上是可以降的），并且触发老化调换会表现得更清楚，老化效率也提高了。

感触最深的是很多信息不能参与到页面调换中，比如实际上的进程相关信息应该常驻内存，一级页表也应该常驻内存，这些信息常驻内存又需要考虑到不要影响一般内存的内容，还是很麻烦的，还好这方面没有做要求。