操作系统实验报告：虚存管理

组长：陈世涉猎 13061089

组员：衣德良 13061070

姬站华 13061077

迟禄 13061084

# 1需求说明

## 1.1基本要求(1)-(5)和提高要求(6)-(9)

通过本实验，要求学生能够了解Linux系统下页式存储管理机制，并实现一个简单的虚存管理模拟程序。具体要求如下：

(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)建立一个多级页表

(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

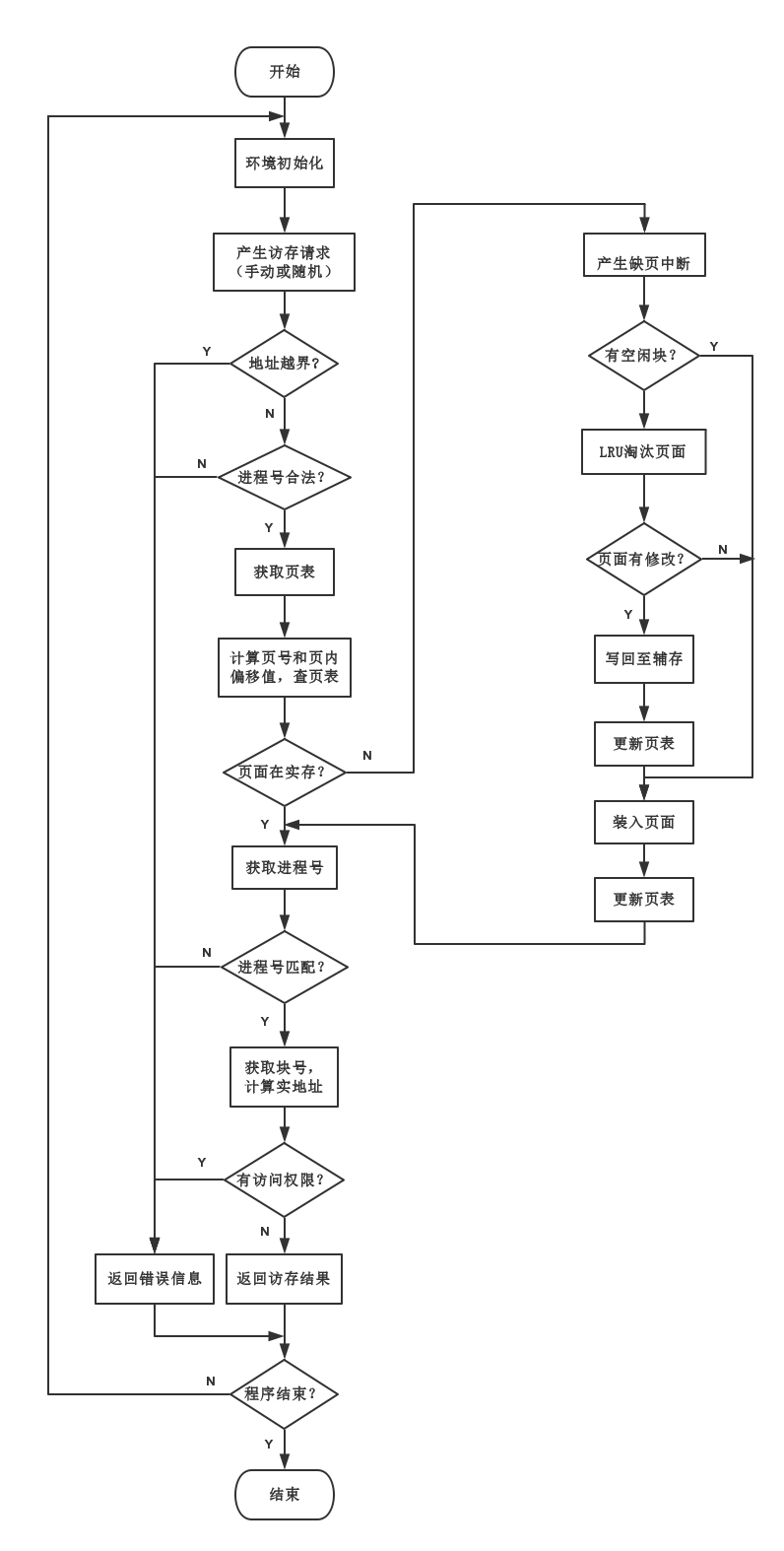
(9)实现其它页面淘汰算法：如页面老化算法、最近最久未使用淘汰算法（LRU）、最优算法（OPT）等

## 1.2完成情况

全部实现。

# 2设计说明

## 2.1流程示意图



## 2.2所使用的系统调用的列表

1. mkfifo():

创建FIFO文件，用于进程间的通信。一个终断通过运行request，创建FIFO文件并将相应指令写入，另一个终端的vmm通过读取FIFO文件调用相应的函数。

1. ftell():

移动文件流的读写位置，调用成功时返回0，失败则返回-1，错误代码保存于errno中。

1. fread():

从文件流读取数据，返回实际读取道德nmemb数。

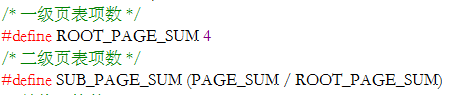
1. fwrite():

将数据写入文件流，返回实际写入的nmemb数。

## 2.3提高要求实现说明

### 1.多级页表

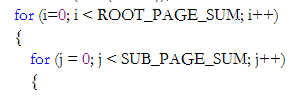
添加宏定义：



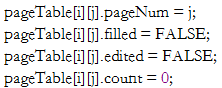
然后用二维数组模拟两级页表：



之后，将程序中所有有关页表遍历的代码改成如下框架：

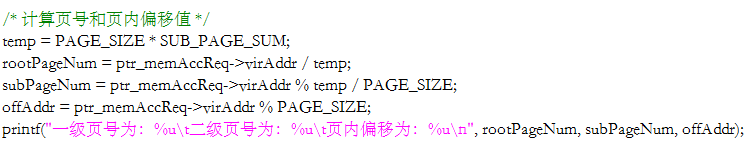


同时，所有对页表中某一项的访问都要从一维改成二维，如：





最后，也是较为重要的，就是计算页号和页内偏移值：



计算的原理为：

|  |  |  |
| --- | --- | --- |
| 一级页表 | 二级页表 | 页内偏移量 |
| 2位 | 4位 | 2位 |

只需要将相应的值从虚拟地址中提取出来即可，然后从页表中找到相应的物理地址（若未装入实存还需进行缺页中断）。

### 2.实现多道程序的存储控制

该功能的实现较为简单，主要为以下两点：

(1)在页表初始化的时候即给定每个页面是由哪个进程申请到的，以后不再改变；

(2)在处理请求时，预先判定提出申请的进程编号和被请求处理的页面对应的申请进程编号是否一致，不一致则报错

所以，在PageTableItem数据结构与MemoryAccessRequest数据结构中添加属性：unsigned int proccessNum；产生请求（随机或手动）时需要产生进程号，在访问的时候，需要核对进程号与访问地址的所属进程号是否一致，不一致则报错。PS：本程序用宏定义定义了模拟过程中进程总数



所以，可以实现任意多个进程模拟。

### 3.进程间通信

本次的进程间通信通过fifo实现。首先，vmm程序初始化时创建一个FIFO文件并打开，之后不断循环读取该文件，若有内容便响应。

if(stat("/tmp/server",&statbuf)==0)//通过文件名获取文件信息，并保存在statbuf结构体中

{

/\* 如果FIFO文件存在,删掉 \*/

if(remove("/tmp/server")<0)

{

do\_error(ERROR\_FIFO\_REMOVE\_FAILED);

exit(1);

}

}

if(mkfifo("/tmp/server",0666)<0)//mkfifo ()会依参数pathname建立特殊的FIFO文件，该文件必须不存在，而参数mode为该文件的权限

{

do\_error(ERROR\_FIFO\_MAKE\_FAILED);

exit(1);

}

/\* 在阻塞模式下打开FIFO \*/

if((fifo=open("/tmp/server",O\_RDONLY))<0)

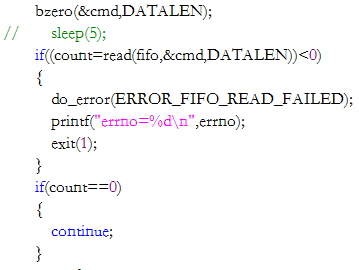
{

do\_error(ERROR\_FIFO\_OPEN\_FAILED );

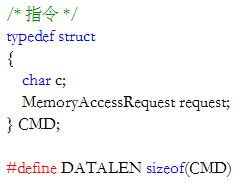
exit(1);

}

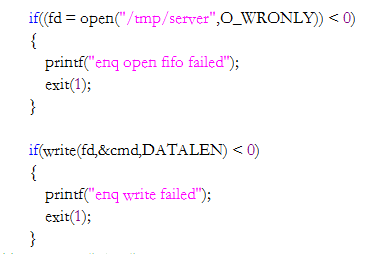
循环读取的部分代码：



另一个终端则是通过request对FIFO文件进行写入，需要创建一个新的结构体来存储指令：



c代表指令类型：打印页表、打印实存、打印虚存、随机产生请求、手动产生请求、退出等等，若是请求，还需要填充MemoryAccessRequest.



### 4.LRU

用一个链表记录使用的物理块，越是排在链表靠前位置的，表示越是最久未使用的，当链表已达到物理块的最大块数时，移除链表的最前一个块号，也就是淘汰最久未使用的。当缺页中断时，在调入新页面的同时在此链表最后位置处新添调入的物理块号；若访问的页面已经存在实存中，则将链表中的该块号移到链表最后。

# 3收获与感想

## 3.1给予你帮助的人

运助教。做进程通信的时候，遇到了FIFO文件无法读取的问题。经过与运助教的沟通、调试，发现导致错误的原因是：fifo文件的打开方式是非阻塞的，这样就会导致数据还未写入就立马读取并返回，这样便会发生读取错误，于是，将fifo文件的打开方式变成阻塞的即可。

## 3.2从实验中学到的东西

通过本实验，我们组的同学对虚存的管理有了更深一步的认识，明白了地址转换、多级页表调度、缺页中断的过程及实现方法。虽然实现过程中遇到了一些小问题，但通过各种尝试与合作，不仅解决了问题，还掌握了一些调试技巧及C语言知识（比如errono，fifo，结构及链表等）。掌握了LRU、LFU页面淘汰算法。让部分之前对实存与虚存关系认识很模糊的同学彻底明白了它们的关系及意义。