操作系统实验报告：实验三

姓名 沈明飞 学号 13061067

王喆 13061074 赵乐 13061076

郭奕鑫 13061088 余旭 13061090

# 1需求说明

## 1.1基本要求和提高要求

**基本要求**

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：

* 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
* 建立一个一级页表
* 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
* 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
* 实现LFU页面淘汰算法

**提高要求**

* 建立一个多级页表
* 实现多道程序的存储控制
* 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
* 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

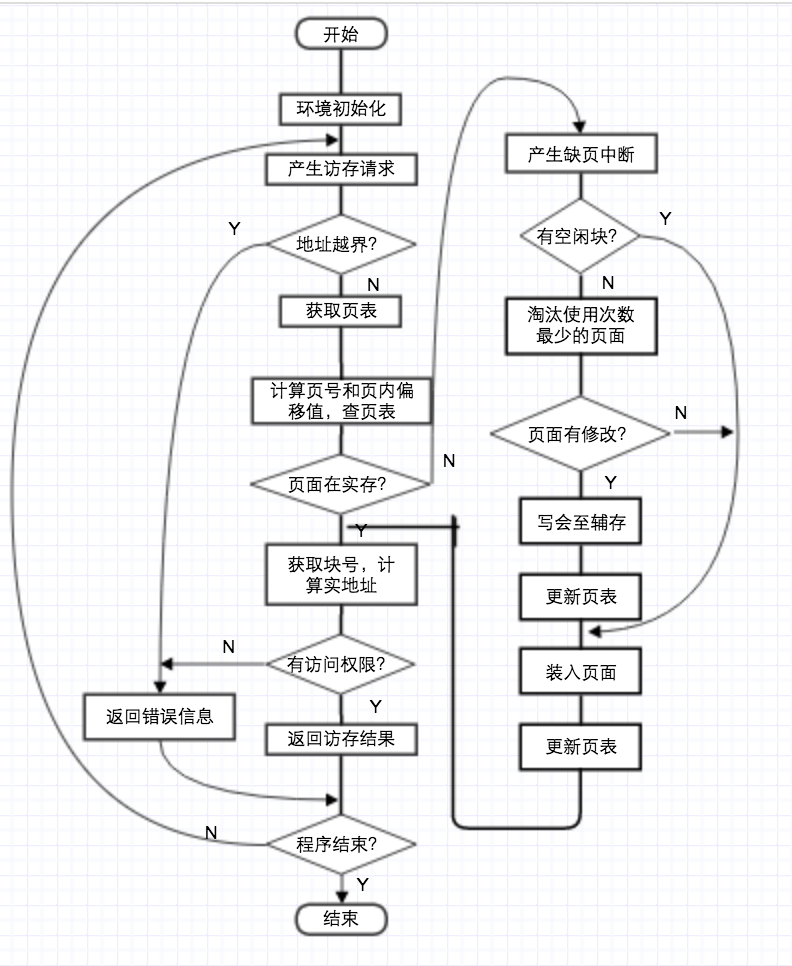
## 1.2完成情况

我们组本次完成了所有基本要求以及提高要求。

# 2设计说明

## 2.1流程示意图

程序流程图：



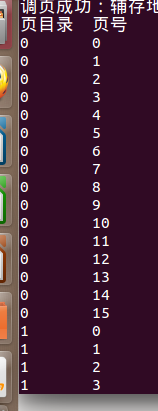
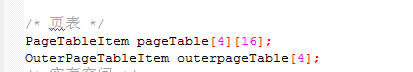
## 2.2所使用的系统调用的列表

|  |  |  |
| --- | --- | --- |
| 名称 | 作用 | 返回值 |
| bzero（void \*s, int n） | 置字节字符串s的前n个字节为零且包括‘\0’。 |  |
| read(int fd,void \* buf ,size\_t count) | read()会把参数fd 所指的文件传送count个字节到buf指针所指的内存中。文件读写位置会随读取到的字节移动。 | 若参数count为0，则read为实际读取到的字节数，如果返回0，表示已到达文件尾或是无可读取的数据 |
| stat(const char \*file\_name, struct stat \*buf) | 通过文件名filename获取文件信息，并保存在buf所指的结构体stat中 | 执行成功则返回0，失败返回-1，错误代码存于errno |
| remove(const char \* pathname) | remove()会删除参数pathname指定的文件。 | 成功则返回0，失败则返回-1，错误原因存于errno |
| mkfifo(const char \* pathname,mode\_t mode) | 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（mode%~umask），因此 umask值也会影响到FIFO文件的权限。 | 若成功则返回0，否则返回-1，错误原因存于errno中 |

## 2.3提高要求实现说明

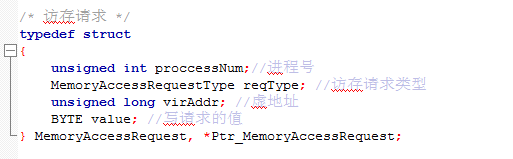
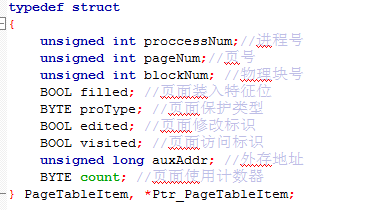
* 建立一个多级页表

将存储页表的长度为64的一维数组pageTable[64]改为4\*16的二维数组pageTable[4][16]；其中二维数组的行数即为页目录（共有4项），每一行各有16个页表项，即页号为16项，页内偏移为4项。打印页表时将页号0-63改为页目录0-3，其中每个页目录均含页号0-15。



* 实现多道程序的存储控制

在页表项中添加属性进程号，手动输入请求或自动生成请求时都要求生成一个进程号（0或1），为每个页表分配一个进程号（两个进程号均分所有页表），只有当请求的进程号与页表所分配的进程号相匹配时，请求才会被执行，否则将返回错误信息。



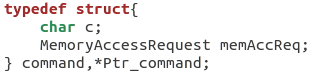
* 将do\_request()和do\_response()实现在不同进程中，通过进程间通信（如FIFO）完成访存控制的模拟

实现了在不同进程间通过管道文件FIFO进行通信。

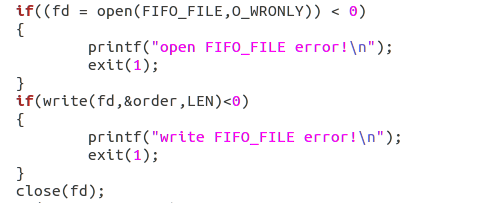
即将源代码拆成两部分，一个文件专管交互式输入，将用户的命令读入，并写入FIFO文件，而另一部分专管从FIFO文件读取命令，并执行。

以下为具体实现过程：

1.首先为每个命令添加标识符属性，以便读命令执行时能区分不同的指令并加以执行

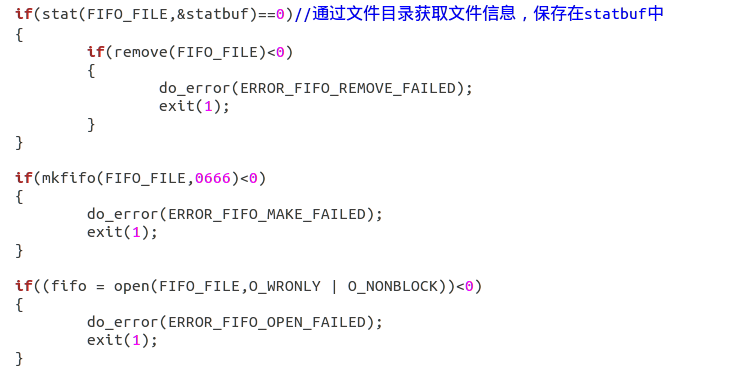


2.分为两部分，处理输入部分，不停的读入命令，并将命令加以区分，为标识符赋值，写入FIFO

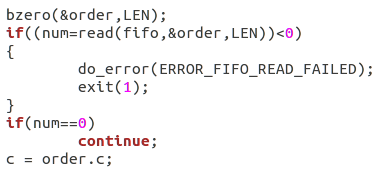


3.执行命令部分，不停的从FIFO中读取定长的数据(恰好为一个命令大小)，并加以区分执行

创建FIFO文件



读取定长指令指令，获取标识符，准备执行

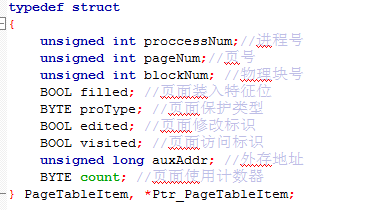


开始执行



* 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

实现了页面老化算法：在页表项中添加visited页面访问标识，每次页面被访问时（读、写、执行）访问标识置为1。取每次响应指令为一个时间点，对所有页面的计数器count进行更新，并重置visited为FALSE。更新方法为将count右移1位，同时如果visited标识为TRUE，则将置最高位为1，否则置为0。当发生缺页中断时，取count值最小的一页进行替换。



# 3收获与感想

## 3.1给予你帮助的人

郭奕鑫，赵乐，王喆，余旭

## 3.2从实验中学到的东西

通过这次实验，我了解了Linux的虚存管理机制，并且熟悉了各种页面淘汰算法，对进程的通信也有了更加深刻的认识。

在实验过程中，涉及到一些C语言中关于文件的函数，这些函数以前不是很了解，或者仅停留在能够正确使用的水平。这次实验中，为了保证实验的正确性，我对这些函数做了更加深入的了解，收获了许多。

此外，这次实验也让我认识到了团队的重要性，大家共同合作，看似庞大的工程就可以在规定时间内高效的完成。

## 3.3任务分配

13061067沈明飞 负责基础要求以及提高要求中的二级页表的实现

13061074 王喆 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

13061076 赵乐 修改了源码中实存规模为512的错误（实际应为256），修正了基础要求完成后存在的BUG，完成了实验报告的撰写。

13061088 郭奕鑫 实现页面老化算法

13061090 余旭 实现多道程序存储控制