**操作系统课程设计实验报告**

——实验三：虚存管理

负责人姓名：李雨霜

学号：14061177

日期：2016.5.7

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 韩慧敏 | 14061218 | 实验一 |
| 2 | 张慧昕 | 14061117 | 实验二 |
| 3 | 李雨霜 | 14061177 | 实验三 |
| 4 | 张然殊 | 14061146 | 实验四 |

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.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.支持页表、辅存和实存内容的打印  2.支持请求命令的手动输入  3.地址转换是否正确  4.页面装入/页面交换是否正确  5.存取控制是否正确  6.读命令读取是否正确  7.写命令实存内容是否正确写入

## 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

完成了以下功能：

# 支持页表、辅存和实存内容的打印

# 支持请求命令的手动输入

# 地址转换是否正确

# 页面装入/页面交换是否正确

# 存取控制是否正确

# 读命令读取是否正确

# 写命令实存内容是否正确写入

# 建立一个多级页表

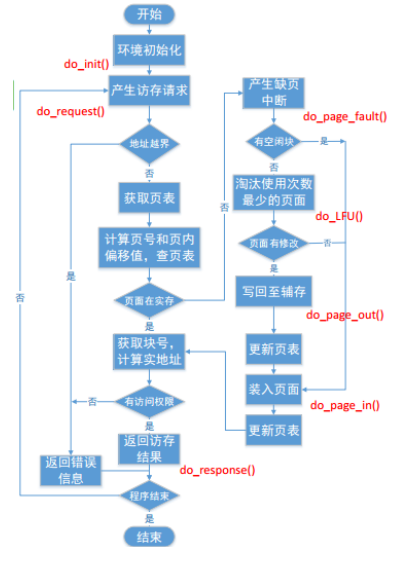
# 实现多道程序的存储控制

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

# 实现页面老化算法

# 3.设计说明

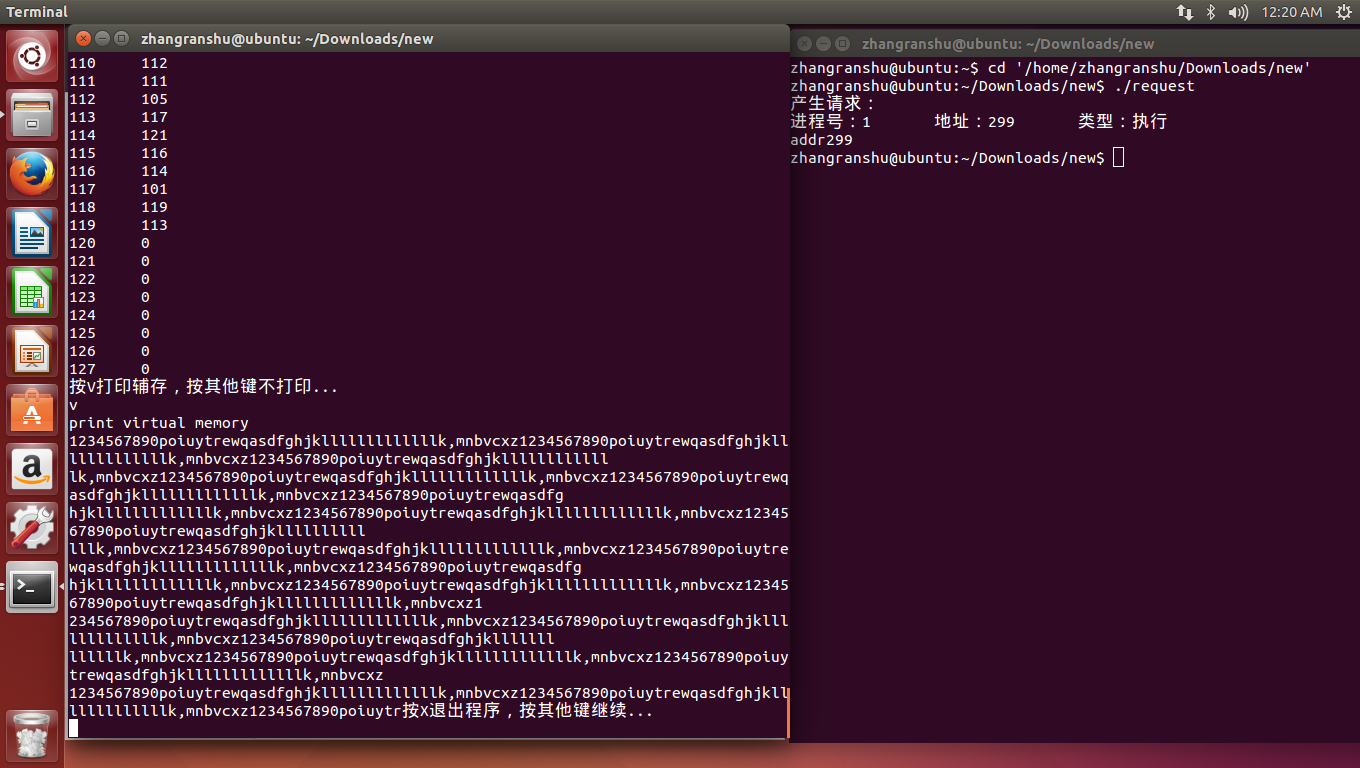
## 3.1 程序流程图

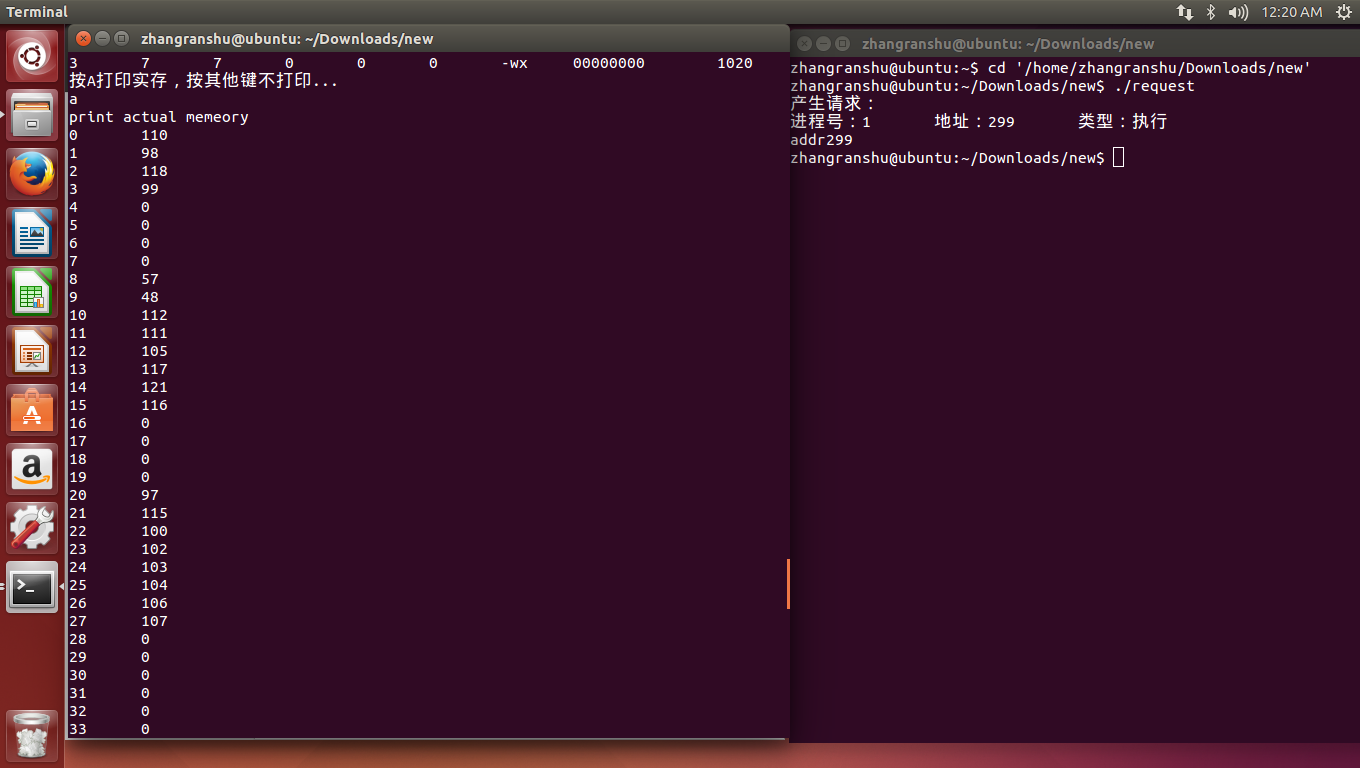


## 3.2基本要求实现说明

# 支持页表、辅存和实存内容的打印







# 支持请求命令的手动输入（见上图）

# 地址转换正确，页面装入/页面交换正确，存取控制正确，读命令读取正确

# ，写命令实存内容正确写入

## 3.3 提高要求实现说明

## 建立一个多级页表

本实验中完成了二级页表。

源码中采用一级页表，直接可查询页表项得到真实地址。修改为二级页表后，前三位表示页目录，中间三位表示二级页表，最后两位表示页内偏移（一页大小4B）。即对于每个进程64\*4B的虚拟空间，有8个页目录，每个页目录下8个页表项，每个页表项对应一个物理地址。

新增的页目录结构如下：

//页目录

typedef struct

{

unsigned int indexNum;//页目录号

unsigned int processNum;//进程号

PageTableItem index[INDEX\_PAGE];//该目录下的页表

} PageIndexItem,\*Pgd\_PageIndexItem;

在寻址时，由原本的寻址方法变为先得到页目录号，再得到页号，由此得到物理块号，最后由物理块号和页内偏移得到真实的地址，再继续进程操作。

## 2.实现多道程序的存储控制

本实验中初始将进程数设置为4（在vmm.h中可调Process\_SUM）。将虚拟空间扩大到64\*4\*4，即每个进程对应赋存axuMem中64\*4的空间。将PageIndex数组扩充成二维数组，其中第一维度用来标明页表属于哪一进程。

PageIndexItem pageIndex[Process\_SUM][INDEX\_SUM];

在生成随机命令时，将随机生成进程号，标明该命令属于哪一进程。在手动输入命令时，也需要输入进程号。

如果进程号与该进程对应的赋存地址不匹配，将返回错误。

在寻址时，加入进程号，对应进程号进行下一步寻址，得到目录号、页号、物理块号、页内偏移，最终计算出实际地址。

ProcessNum = aptr\_memAccReq->ProcessNum;

pageNum = (aptr\_memAccReq->virAddr-ProcessNum\*256) / PAGE\_SIZE % INDEX\_PAGE ;

indexNum = (aptr\_memAccReq->virAddr-ProcessNum\*256) / PAGE\_SIZE / INDEX\_PAGE;

offAddr = (aptr\_memAccReq->virAddr-ProcessNum\*256) % PAGE\_SIZE;

actAddr = ptr\_pageTabIt->blockNum \* PAGE\_SIZE + offAddr;  
 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

do\_request实现在不同进程：

利用命名管道fifo，在新终端调用./request，如无参数则随机生成，否则根据输入参数生成请求，将请求写入fifo，在vmm中创建fifo并以只读打开，每次循环中读取一条请求并加入链表中。

## edc0bbea85874788d5a001b9285bc430request.c中向fifo中写

## 5340b708089c2be4f76dcfbc20ce0b32vmm.c中创建fifo并打开

## 91bb53e78d4a604c3673809c1d584749循环中读取请求

## 关于do\_response()由于组内有不同见解，故目前上交的代码中在每次循环中响应请求。如想只在新终端下执行./response,可对循环中以下代码进行修改，去掉switch中三行注释，并把switch后的do\_response注释掉，则每次循环中根据在新终端的输入进行请求的产生和响应。

## 82c060afdb1d9bb6dd4ae49e455984ca 4.实现页面老化算法

void do\_PA(Ptr\_PageTableItem ptr\_pageTabIt)

{

unsigned int i,k,s, min, page,index,processNum, c, temp\_count;

printf("没有空闲物理块，开始进行页面老化算法页面替换...\n");

for(s=0,min=0xFFFFFFFF,page=0,index=0,processNum=0;s<Process\_SUM;s++)

for(k=0;k<INDEX\_SUM;k++)

for (i = 0; i < INDEX\_PAGE; i++)

{

for(c = 0, temp\_count = 0;c < 8;c++){

temp\_count = temp\_count \* 10 +

pageIndex[s][k].index[i].count[c] - '0';

}

if (temp\_count < min &&

pageIndex[s][k].index[i].filled==TRUE)

{

min = temp\_count;

processNum=s;

page = i;

index=k;

}

}

printf("选择第%u个目录第%u页进行替换\n",index,page);

if (pageIndex[processNum][index].index[page].edited)

{

/\* 页面内容有修改，需要写回至辅存 \*/

printf("该页内容有修改，写回至辅存\n");

do\_page\_out(&pageIndex[processNum][index].index[page]);

}

pageIndex[processNum][index].index[page].filled = FALSE;

pageIndex[processNum][index].index[page].visited = 0;

for(c = 0;c < 8;c++){

pageIndex[processNum][index].index[page].count[c] = '0';

}

/\* 读辅存内容，写入到实存 \*/

do\_page\_in(ptr\_pageTabIt,pageIndex[processNum][index].index[page].blockNum);

/\* 更新页表内容 \*/

ptr\_pageTabIt->blockNum =pageIndex[processNum][index].index[page].blockNum;

ptr\_pageTabIt->filled = TRUE;

ptr\_pageTabIt->edited = FALSE;

ptr\_pageTabIt->visited = 0;

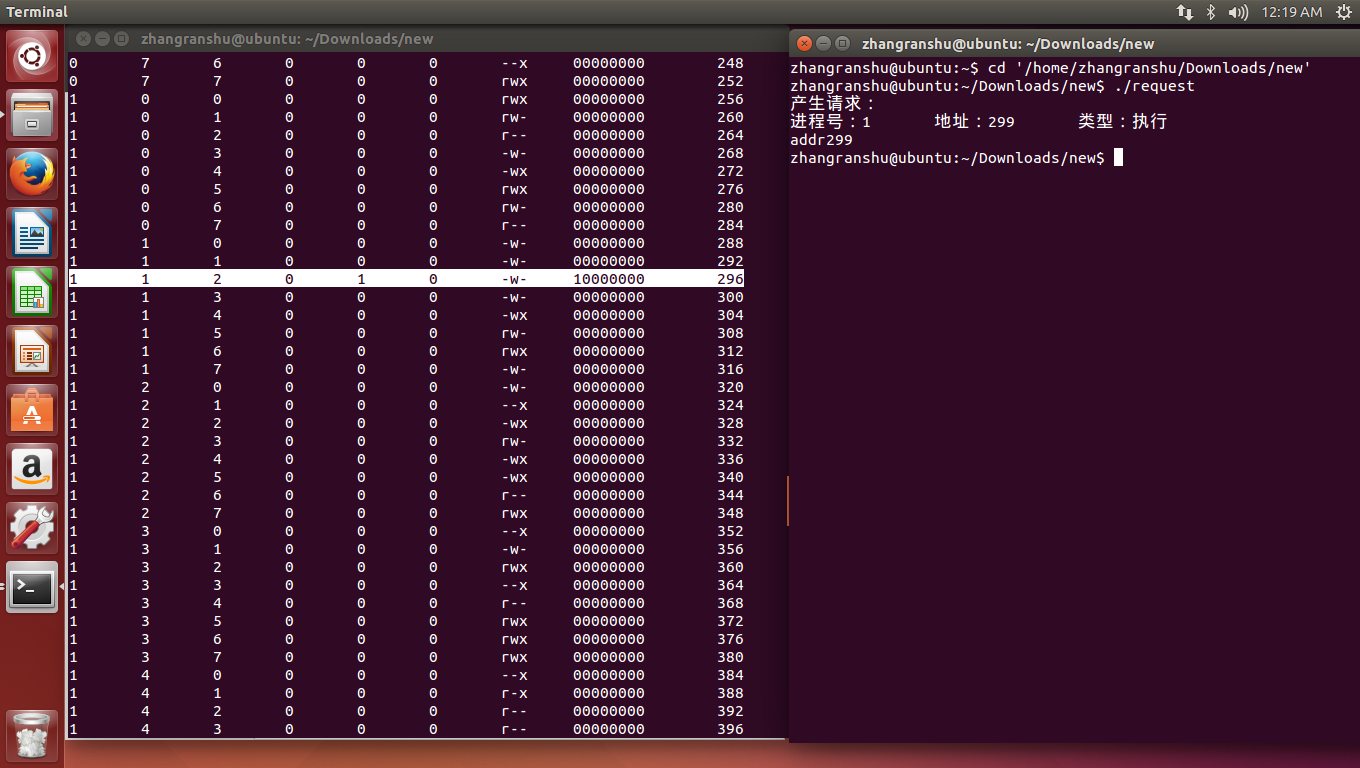
for(c = 0;c < 8;c++){

ptr\_pageTabIt->count[c] = '0';

}

printf("页面替换成功\n");

}





# 4.收获和感想

李雨霜：本次实验的难点在于概念的理解，对于多级页表机制的理解占比较多的时间，在理解之后修改代码的工作虽然涉及较多细节，比较琐碎，不过思路流畅，并不很难。多道程序的模拟，我们实现得较为简单，可能有些细节未能考虑到。

张慧昕：通过这次实验，对于页表对于虚存有了更深刻的理解，也对fifo有了更加熟练的应用，在实验的过程中更加体会到合理任务安排和时间分配的重要性。

张然殊：这次实验我负责的部分主要是基本功能的检查和实现以及实现页面老化算法。由于对页表这一部分的内容不是很熟练，刚开始还是有些棘手的，但是后来在大家的讨论以及不断学习中有了稍好的感悟。我深深体会到了团队协作的重要性，非常感谢我的队友们。

韩慧敏：此次的os实验又结束了，我一定要反思一下，自己平时对于os理论课的态度。如果说之前还没有看出os理论课和os实验课的关系，那么本次实验的内容和理论课的联系可谓十分密切，关于多级页表和淘汰算法都是理论课上老师反复强调的重点。但是因为一直以来对os理论课都没有给予足够的重视，上课的时候不能做到百分百的专心，课下也没能及时的复习，对知识的掌握不牢固，以致于做os实验的时候遇到了概念不清，逻辑混乱的问题，大大降低了写作业的效率。我们平时总会拿学习压力大作为借口，但是大多数时候，都是因为贪玩，偷懒导致对知识掌握不牢，导致效率低下，形成了恶性循环，想要课下轻松，就一定要抓紧上课的时间，认真，专注。