**操作系统课程设计实验报告**

——实验三：虚存管理

负责人姓名：叶俊辰

学号：14061166

日期：2016.5.11

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 王燕飞 | 14061156 | 实验一 |
| 2 | 林峥 | 14061160 | 实验二 |
| 3 | 叶俊辰 | 14061166 | 实验三 |
| 4 | 龚瑞昊 | 14061192 | 实验四 |

目录

[1.实验目的 4](#_Toc446001831)

[2.需求说明 5](#_Toc446001832)

[2.1基本要求 5](#_Toc446001833)

[2.2 提高要求 5](#_Toc446001834)

[2.3 完成情况 6](#_Toc446001835)

[3.设计说明 8](#_Toc446001836)

[3.1 程序流程图 8](#_Toc446001837)

[3.2基本要求实现说明 9](#_Toc446001838)

[3.3 提高要求实现说明 1](#_Toc446001839)2

[4.收获和感想 1](#_Toc446001840)9

# 

# 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（do\_request()函数是随机生成请求）

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

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

以上4个提高要求均完成。

## 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（do\_request()函数是随机生成请求）

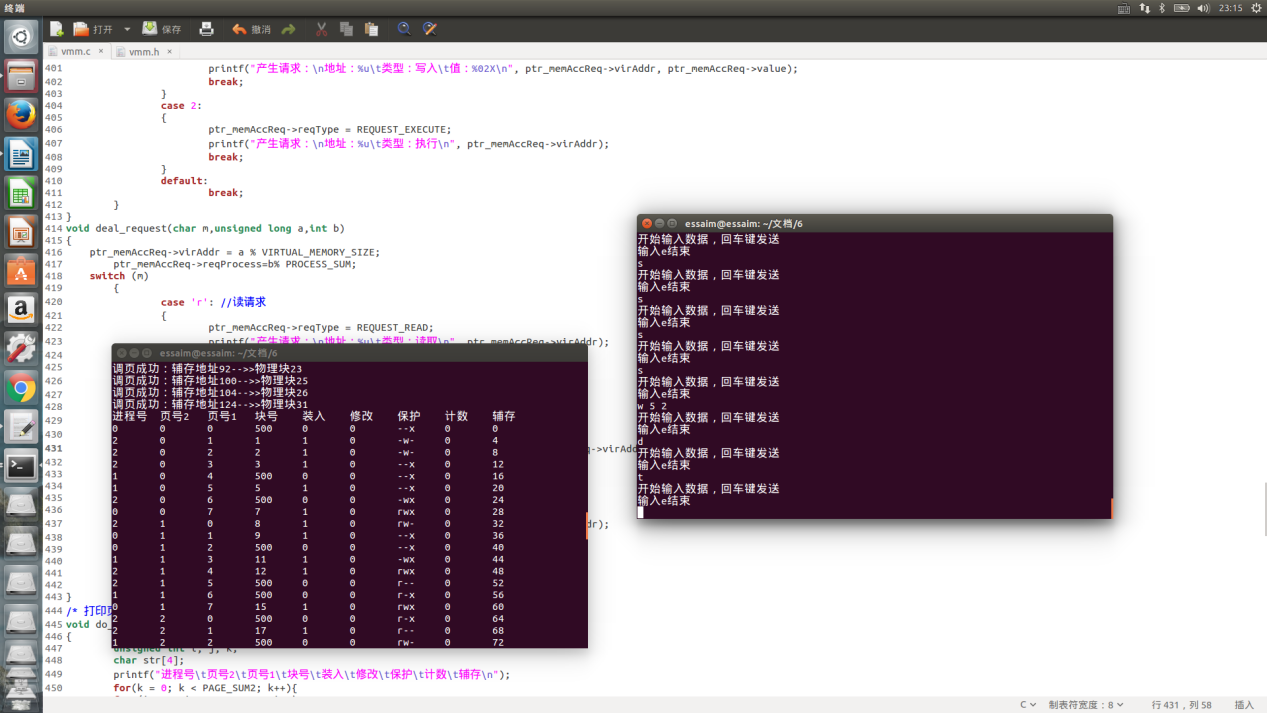
3.地址转换是否正确

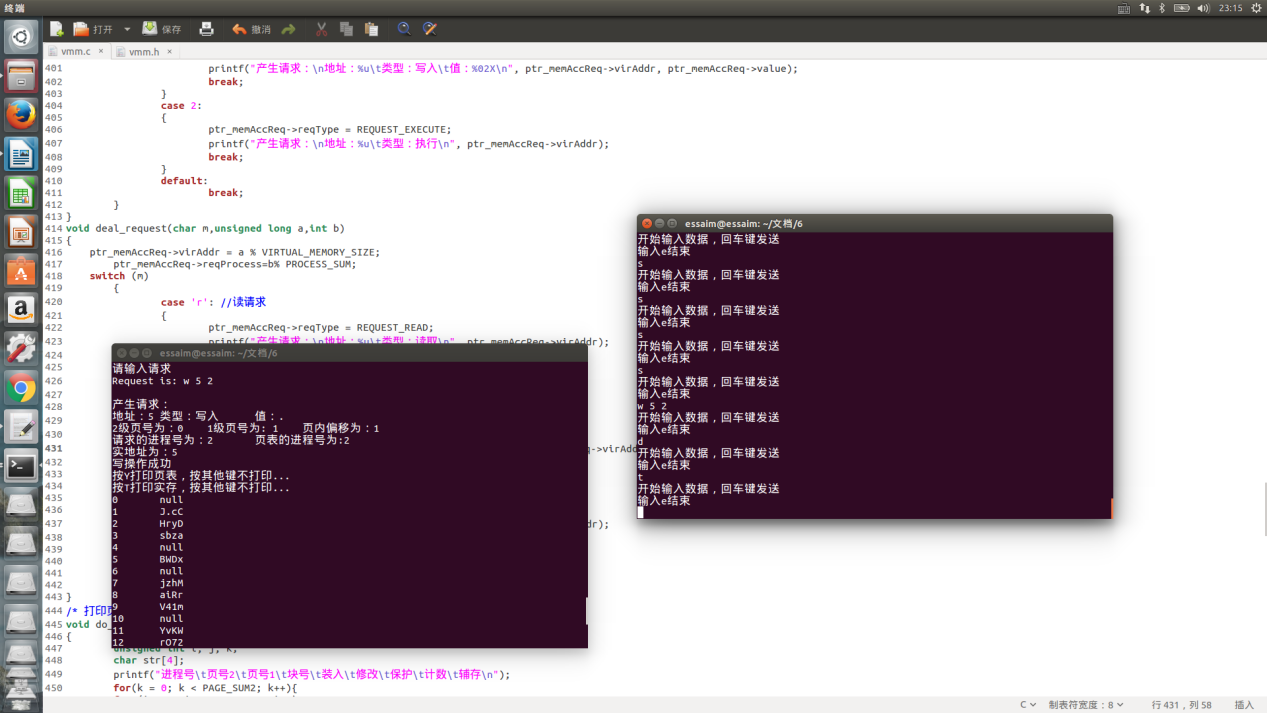
4.页面装入/页面交换是否正确

5.存取控制是否正确

6.读命令读取是否正确

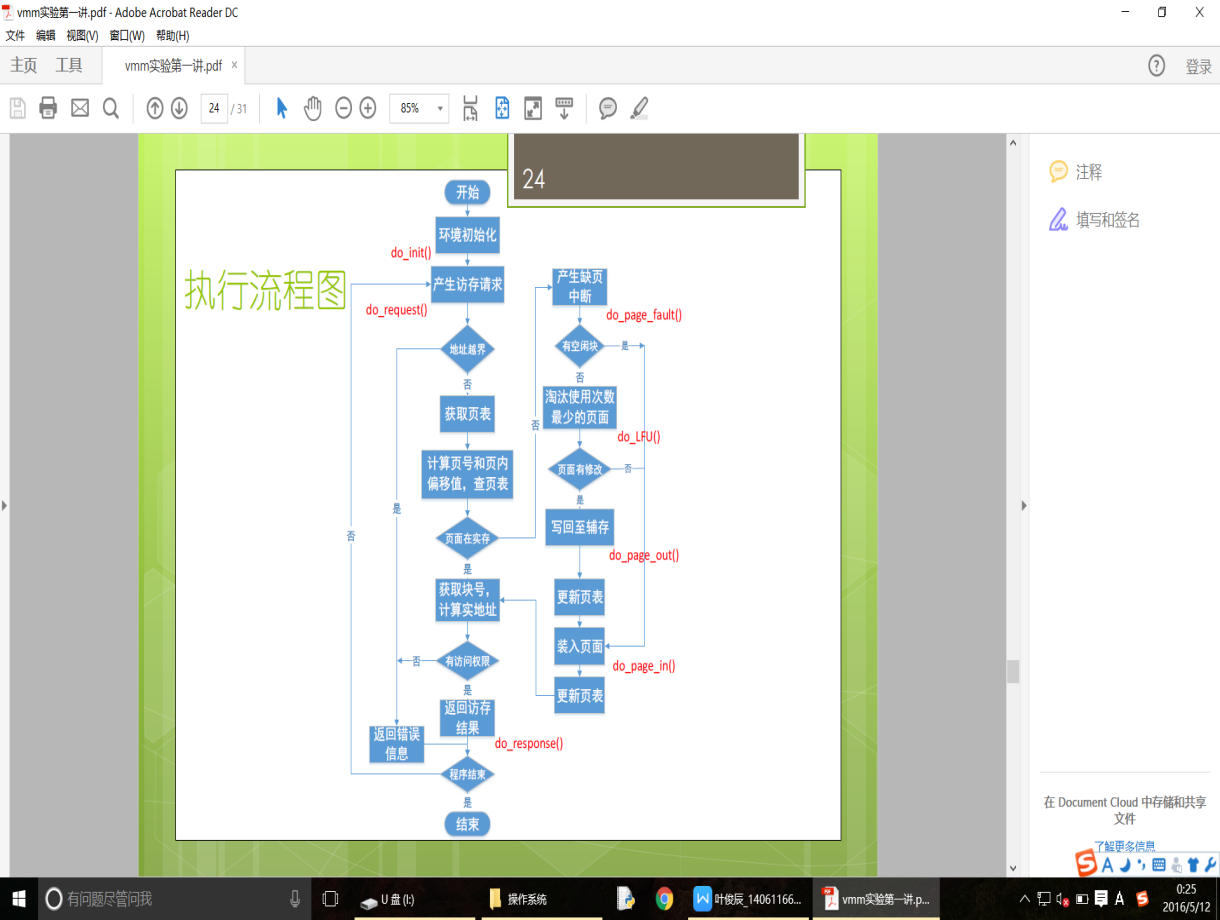
7.写命令实存内容是否正确写入

其中实存打印没有完成，因为我看漏了。但是在检查之后我补上去了，所以在提交的代码中这部分是完成了的。



# 3.设计说明

## 3.1 程序流程图



感觉学长给的这张图已经说得很完美了。

## 

## 3.2基本要求实现说明

## 2016-05-11 22:53:35屏幕截图

## 2016-05-11 22:53:52屏幕截图

## 2016-05-11 22:53:46屏幕截图

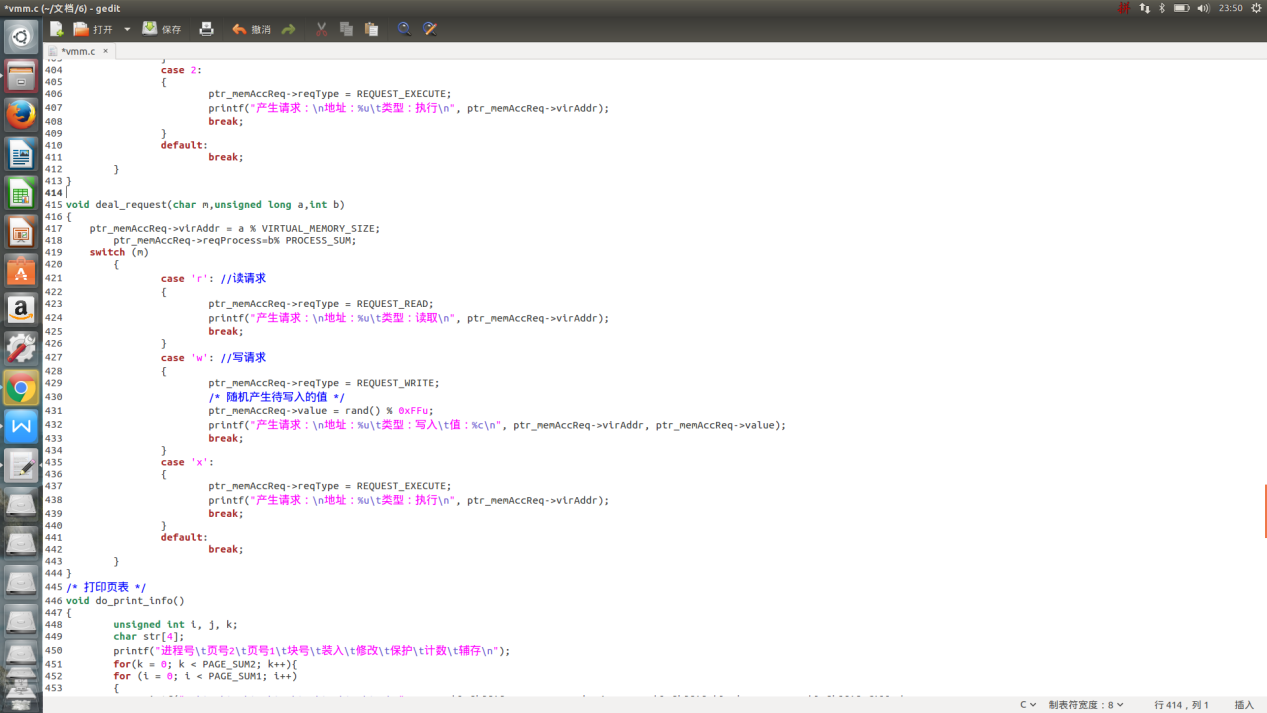
## 在完成基础要求，首先对代码进行了通读，发现了上面四个不合理之处。

修改一：对页表和虚存的对应进行了修改，如果不进行修改，会使虚存地址超出虚存的空间。

修改二：将两个输出的地方参数换了位置，因为这两个参数反了。

修改三：将没有对应实存的页表的对应实存的那一项置成了500而不是之前的0,避免和0号物理块混淆。

修改四：在LFU算法中寻找替换块的时候加上啦filled的判断。

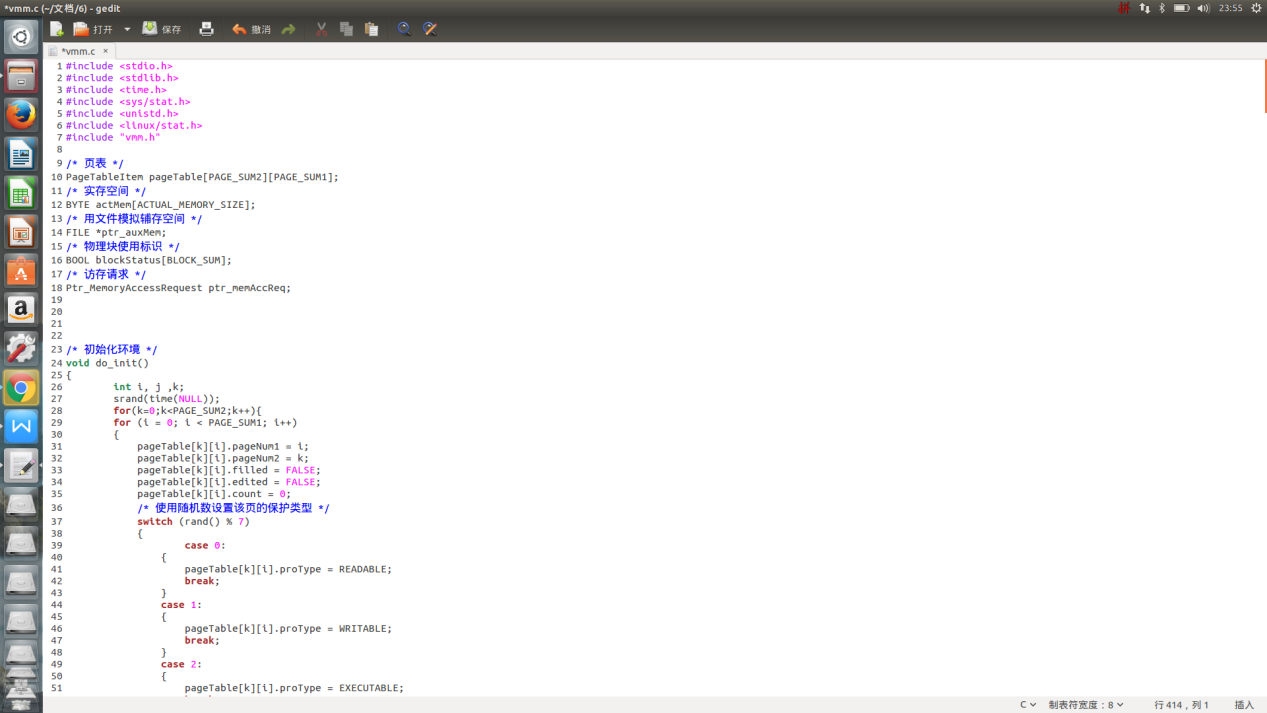
然后再开始完成基础要求。

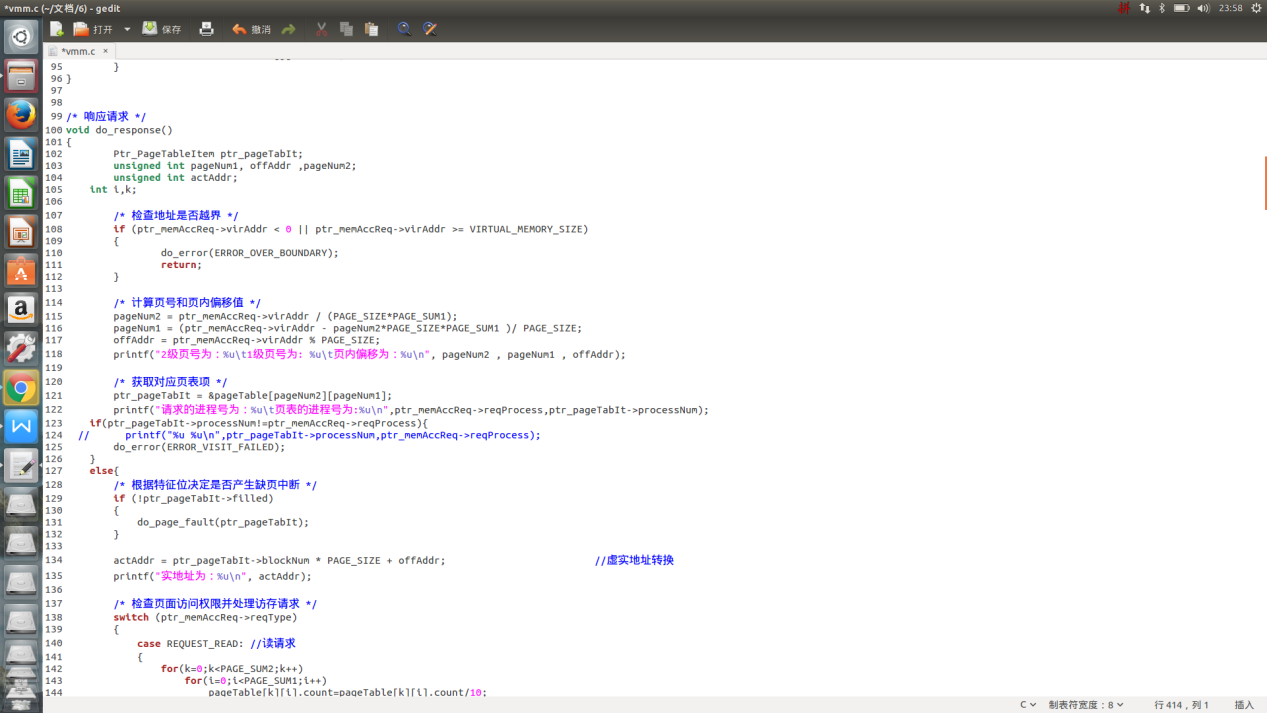
## 2016-05-11 23:51:25屏幕截图

加入了一个deal\_request的函数，对自己生成的请求进行处理。

加入了一个按T打印实存的小循环，来打印实存。

## 3.3 提高要求实现说明

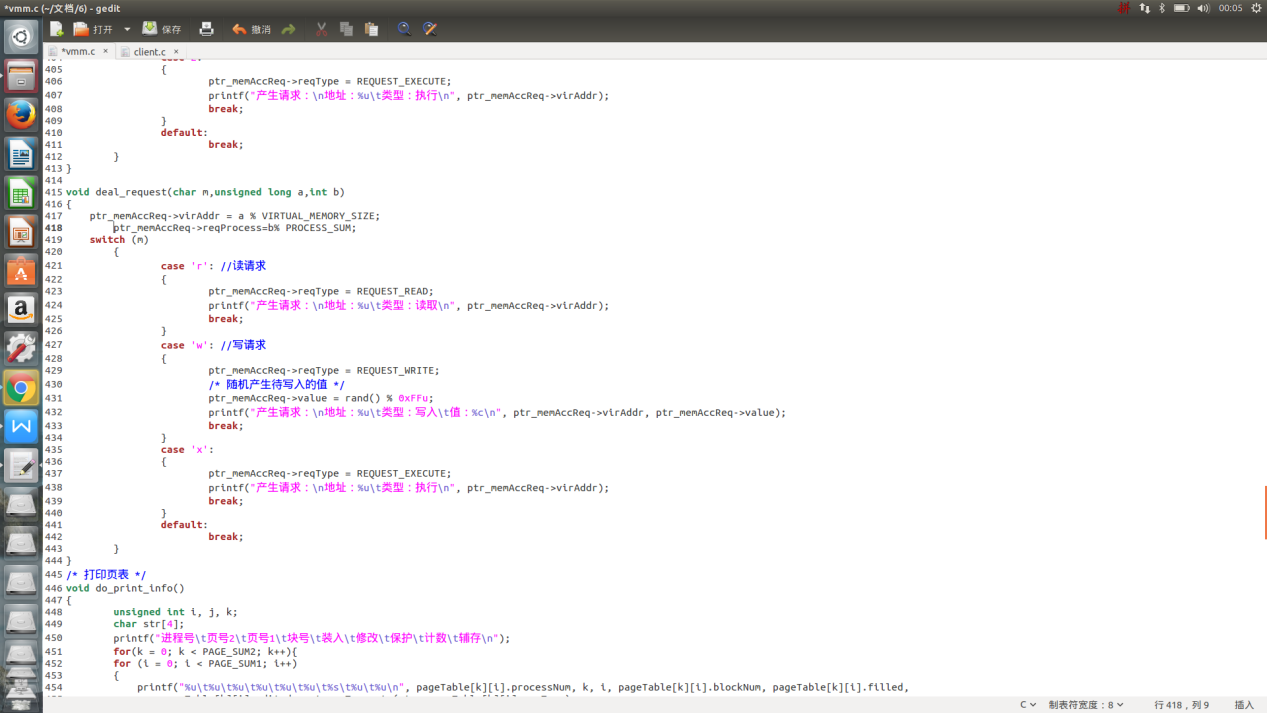
1. 建立一个多级页表将pageTable从一维数组改成了二维数组，这样表示将原来的一级页表改成啦二级页表。



同时在地址的运算上，做了些许改动。

2.实现多道程序的存储控制

在产生请求的时候，给请求加上了一个新的特征，进程号。然后页表在生成的时候也会随机分配给一个进程，看请求的进程和目标页表的进程号是否相同，如果相同则可以访问，不同则不能访问。



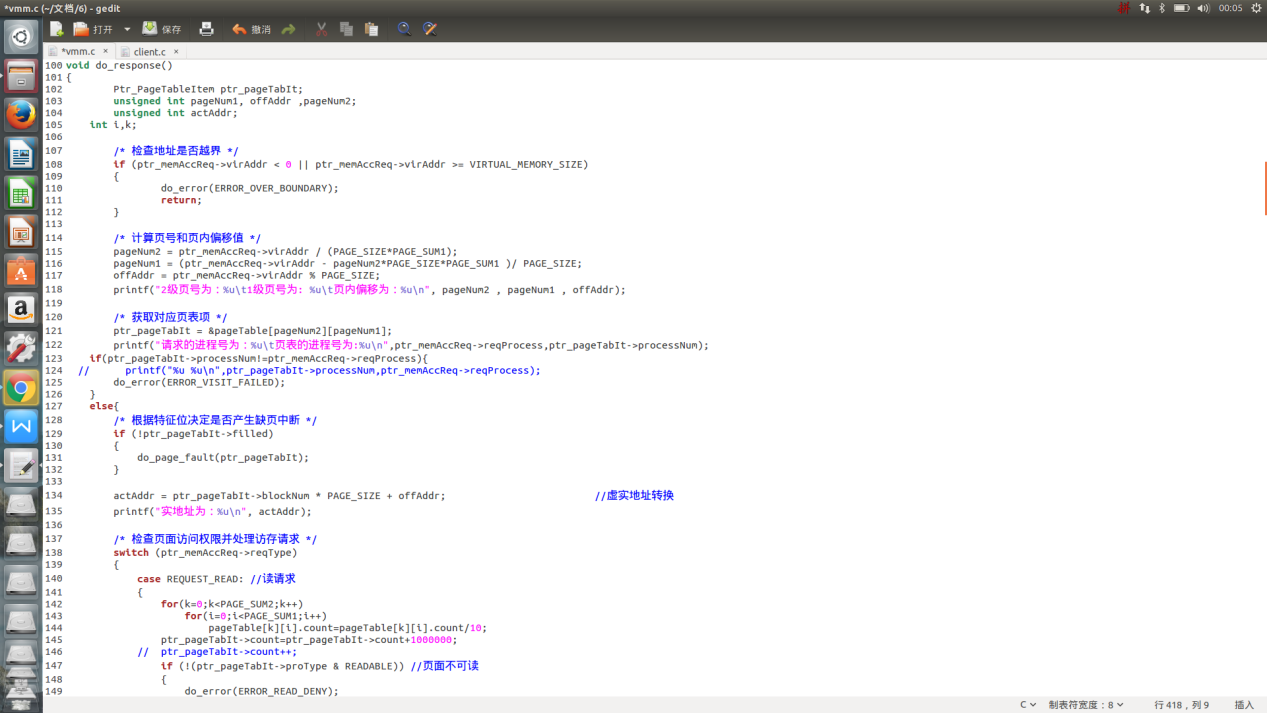
ptr\_memAccReq->reqProcess=b% PROCESS\_SUM;

这是请求的进程号初始化。



pageTable[k][i].processNum = rand() % 3;

这是页表的进程号初始化。



if(ptr\_pageTabIt->processNum!=ptr\_memAccReq->reqProcess){

// printf("%u %u\n",ptr\_pageTabIt->processNum,ptr\_memAccReq->reqProcess);

do\_error(ERROR\_VISIT\_FAILED);

}

这是对进程号是否相等的判断。

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

通过FIFO，我写了一个客户端，可以向程序中发送我在客户端中所输入的数据。

#include <stdio.h>

#include <stdlib.h>

#define FIFO "MYFIFO"

int main()

{

FILE \*fp;

char str[1000];

char c;

int i=0;

/\* 打开FIFO \*/

while(1){

if((fp=fopen(FIFO,"w"))==NULL){

perror("open");

exit(1);

}

printf("开始输入数据，回车键发送\n");

printf("输入e结束\n");

i=-1;

str[++i]=getchar();

if(str[i]=='e'||str[i]=='E'){

str[i+1]='\0';

break;

}

while(str[i]!='\n'){

str[++i]=getchar();

}

fputs(str,fp);

fclose(fp);

}

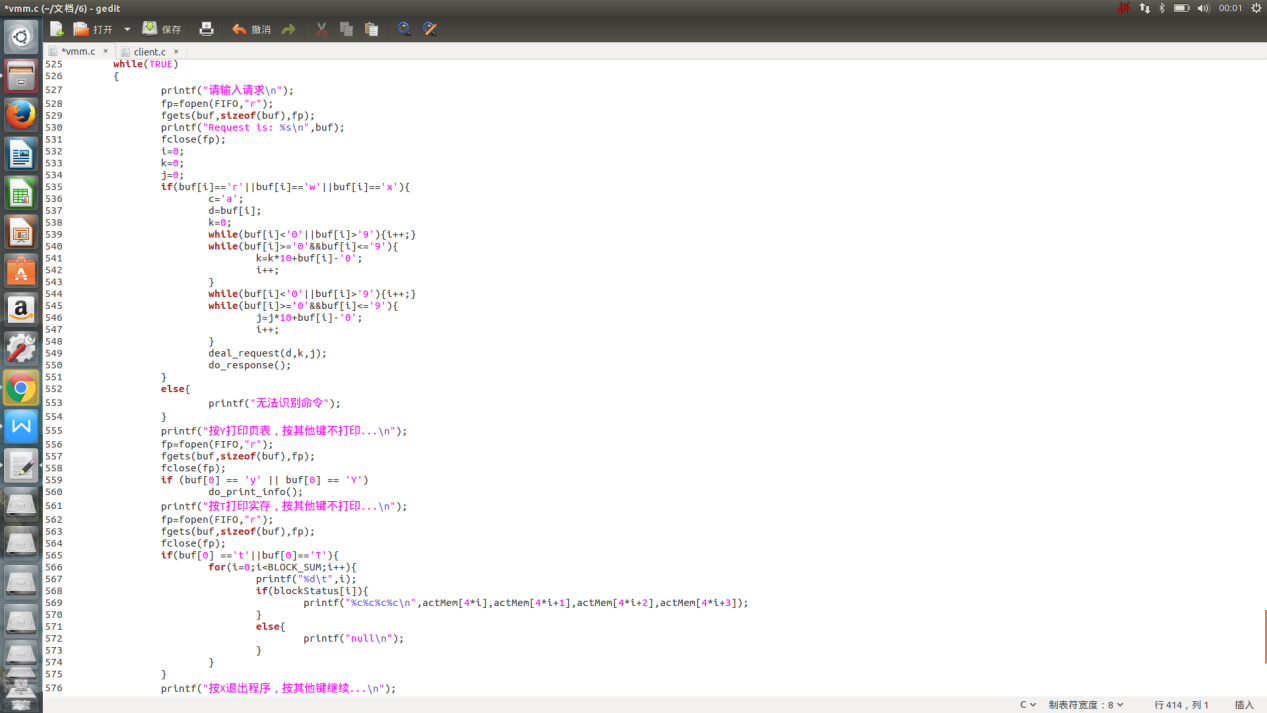
fputs(str,fp);

fclose(fp);

return 0;

}

然后在主程序中接收，处理



这就是处理的程序。

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



我实现的算法应该和页面老化算法最接近，但是又有一点差别。

for(k=0;k<PAGE\_SUM2;k++)

for(i=0;i<PAGE\_SUM1;i++)

pageTable[k][i].count=pageTable[k][i].count/10;

ptr\_pageTabIt->count=ptr\_pageTabIt->count+1000000;

上面是改动的代码，用上面这段代码代替了 //ptr\_pageTabIt->count++;

其余的地方没有改动。

如果访问了该页表，这个页表的计数就除以10后加1000000，其他的没访问到的页表就除以10。

# 

# 4.收获和感想

这次能担任这次实验的主负责人，我的收获很大，首先明白的linux内存管理的机制，页表虚存实存的关系的掌握更深了一个层次。其次，大大提高了我的读代码和分析代码的能力，因为一个人反反复复把这五六百行代码读了好几遍。