编号：_____39061416____
版本：_____1.0_____

北京航空航天大學
BEIHANG UNIVERSITY

Project：Linux 虚存管理

编写	黄建宇	2012. 04. 14
复查	黄建宇	2012. 04. 14
批准	黄建宇	2012. 04. 14

目录

1 小组成员

每个人的贡献大小以此排名为依据，靠前的为贡献较大者
39061416 黄建宇

2 实验目的与总体结构

了解实验的目的，对于作业的整体设计说明，要求思路清晰，表达明确

2.1 实验目的

- 了解 Linux 的内存管理机制。
- 掌握页式虚拟存储技术，理解虚地址到实地址的定位过程。
- 掌握“最不频繁使用淘汰算法”，即 LFU 页面淘汰算法

2.2 需求说明

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.基本要求：

- 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。

2.扩展要求：

- 实现多道程序的存储控制。
- 建立一个多级页表或快表。
- 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。

2.3 设计说明

2.3.1 结构设计

➢ 程序逻辑设计

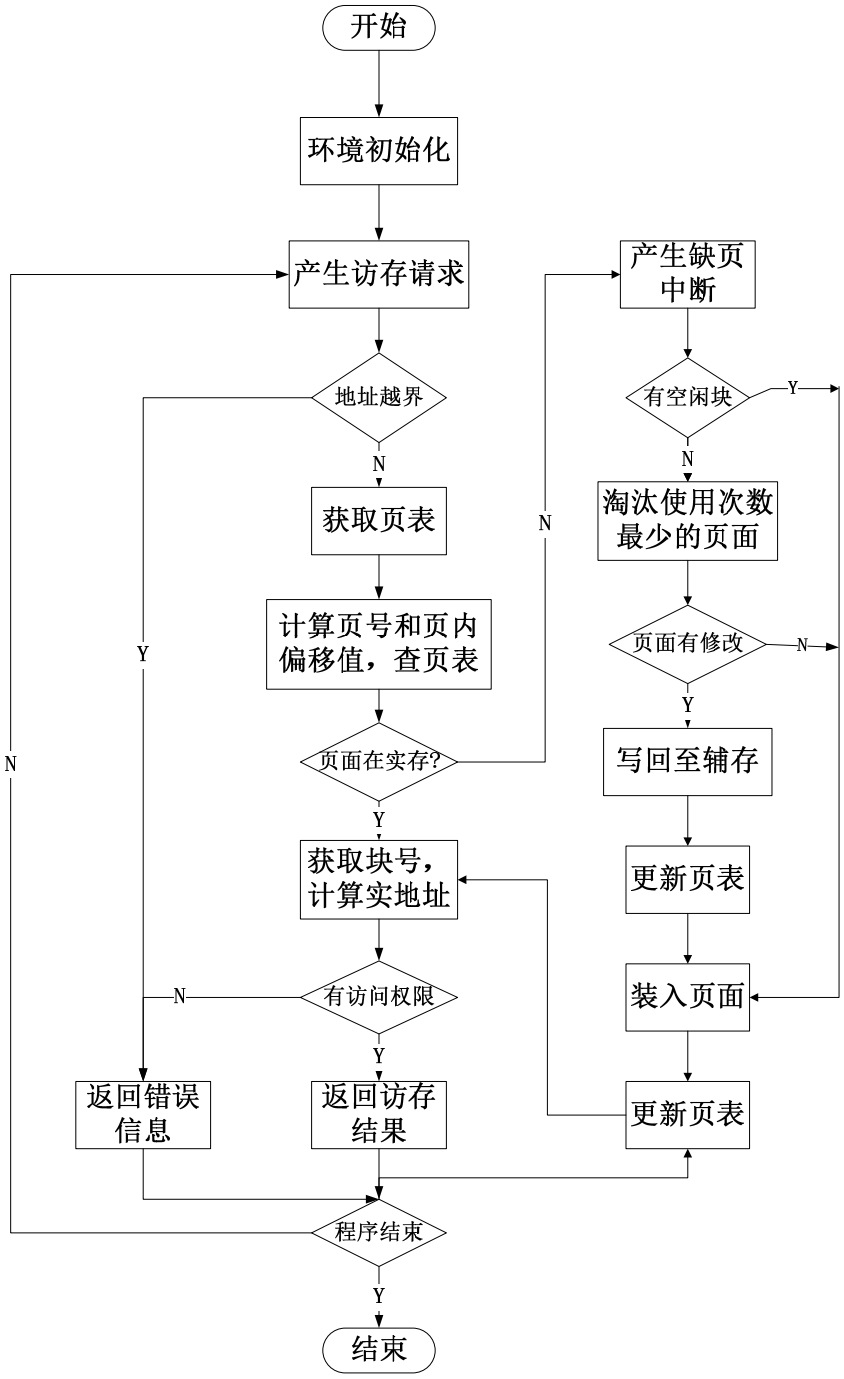

图2. 1 程序逻辑流程图

2.3.2 功能设计

1、数据结构：

表2. 1 数据结构

结构名称	结构标识名称	字段信息	字段说明	结构功能描述
一级页表	OuterPageTableItem OuterPageTablePtr	int page_num	此页表的页号	多级页表中的第一级，建立到二级页表的映射
		int page_index	存储二级页表项的首地址	
二级页表	PageTableItem *PageTablePtr	int page_num	此页表的页号	多级页表中的第二级，建立到辅存的映射
		int block_num	存储的块号	
		BOOL filled	特征位	
		BOOL changed	页面是否改变	
		uchar pro_type	保护类型	
		ulong virtual_Addr	对应虚存地址	
		ulong count	页面使用计数	
		uint no_used	是否久未使用	
访存请求	MemoryAccessRequest *MemoryAccessRequestPtr	RequestType request_type	访存请求类型	存储产生的访存请求信息
		ulong virtual_Addr	访问的虚存地址	
		uchar value	请求的值	

2.主要函数与接口说明

全局函数：

```
//欢迎函数
void welcomeVMMRequest();
void welcomeVMMResponse();
//初始化环境
void init();
//内存访问
void do_request(MemoryAccessRequestPtr);      //produce the memory access request randomly
void do_response();                            //response to the memory access request
//处理页面
void do_page_in(PageTablePtr,unsigned int);    //move the contents in the disk into the actual memory
void do_page_out(PageTablePtr);        //move the contents in the actual memory out to the disk
void do_page_fault(PageTablePtr);      //deal with the page fault
//页面替换算法
void do_LFU(PageTablePtr);                     //least frequency use algorithm
```

```
void do_FIFO(PageTablePtr);                    //first in, first out
void do_LRU(PageTablePtr);                     //least recently use
//打印页表信息
void print_pageinfo();
//错误处理函数
void handle_error(ErrorType);
//获取保护类型字符串
char *get_protype_str(char *str,unsigned char type);
```

3.接口与调用关系设计

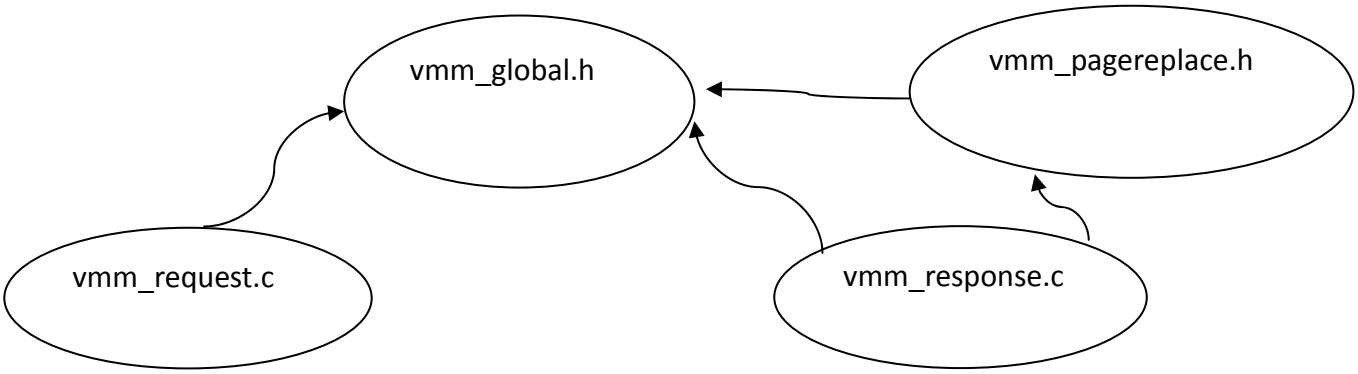

图2. 2 接口与调用关系图

4. 核心逻辑或功能代码

请见 3 实验内容中的 3.1 实现方法以及详细注释。

3 实验内容

详细说明实验的过程，实现方法及遇到的问题

3. 1实现方法

- 进程间通信

进程间通信采用的是 FIFO 机制，通过 FIFO 实现 do_request 与 do_response 之间的通信，将 do_request 写在一简单程序 vm_request.c，在 vm_request.c 里打开 FIFO，按照用户的需求，产生一定数量的请求，并将产生的请求写入 FIFO。响应程序 do_response()在 vmm_response.c，通过在 vmm_response.c 里面打开 FIFO，读出里面的数据进行解析，再进一步响应，这样就可以实现进程间的信息交流。

- 页面替换算法
  - FIFO 算法
  - LRU 算法

✧ LFU 算法

具体请见 vmm_pagereplace.h

➢ 多级页表

在得到一个逻辑地址后，将其分为三部分，分别为：一级页表号，对应的二级页表偏移，最后是页内偏移。如下图所示：

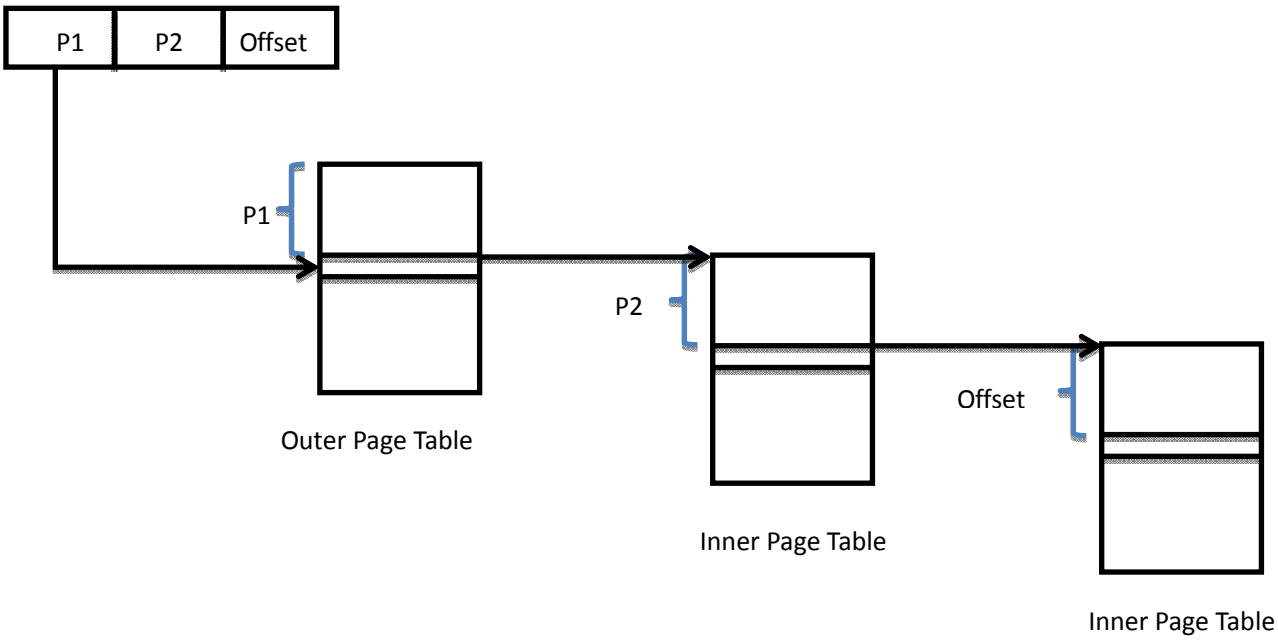

图3. 1 多级页表示意图

➢ 多道程序存取控制

假设虚存文件里共存有 4 个程序，两个程序均分了虚存的大小，我们的虚存共有 64*4，页面大小为 4，续存共有 64 页，每个程序占了 32 页大小的空间。因而需要对每一个程序建立其对应的页表。这里采用的是两级页表的形式，对于每个程序来说一级页表共需 4 个，为了简化操作，将一级页表连在了一起， 这 4 个程序分别占用相应的虚拟内存空间。

3. 2测试和使用说明

3.2.1 程序开发环境：

gcc、gdb

3.2.2 运行环境

unbuntu 9.04，linux 内核 2.6 以上版本

3.2.3 安装说明

1. 打开源码所在文件夹，make 编译。
2. 打开两个或者多个终端，切换到源码所在路径后，其中一个（或多个）终端输入./request，输入所需生成的访存请求个数，生成访存请求，即在另一进程./response 里产生主存的访问请求，另一个终端输入./response，初始化页表后，等待访存请求的到来，处理访存请求。
3. 进程./response 通过进程间通信（普通的 file 或者 FIFO(命名管道)，分别对应源码中的 vmm(file)和 vmm(FIFO)）获得访存请求，同时用 FIFO 的方式处理它们，直到所有 request 都处理完后，./response 继续等待下一次访存请求。此时再运行一个或多个./request，输入所需生成的访存请求个数，生成访存请求，即可继续运行处理程序。

特别说明：
在源码 vmm(FIFO)中，需要先运行./response，处于等待接收另一终端运行./request 进程发送访存请求的状态。此时，在另一终端运行./request 后，./response 才能做出响应。（先运行./request，等运行完后再运行./response 无效。必须两个终端同时运行./request 和./response）
在源码 vmm(file)中，可以先运行./request 再运行./response。
这是由于进程间通信通过的 file 与 FIFO（命名管道）两者的不同特性造成的。

3.2.4 测试用例和运行结果分析:

- **输入描述**
  在两个终端中，分别运行./request 和./response
- **输出描述**

./request:输出访存请求
./response:初始化时打印页表，每处理一次访存请求，可以打印当前页表信息。

- **运行结果**

1 makefile 开始编译程序，运行./request 进程生成访存请求。

```
james@james-desktop:~/Desktop/vmm(file)$ make
cc -o request vmm_request.c
cc -o response vmm_response.c
james@james-desktop:~/Desktop/vmm(file)$ ./request

////////////////////////////////////////////
        Welcome to VMM 1.0 Request Part
        Author:39061416 Huang Jianyu
        Website:www.huangjy.info
        Mobile Phone:15210965935
        Email:hjyahead@gmail.com
////////////////////////////////////////////


Please input the number of the requests:
2
Generate request:
Address:119     Type:read
Success request : 1
Generate request:
Address:158     Type:write      value:CC
Success request : 2
james@james-desktop:~/Desktop/vmm(file)$
```

图3. 2 ./request运行截图

2 ./response 运行后初始化页表

```
james@james-desktop:~/Desktop/vmm(file)$ ./response

////////////////////////////////////////////
        Welcome to VMM 1.0 Response Part
        Author:39061416 Huang Jianyu
        Website:www.huangjy.info
        Mobile Phone:15210965935
        Email:hjyahead@gmail.com
////////////////////////////////////////////


Read page success: auxiliary memory address 8 -->> actual block 1
Read page success: auxiliary memory address 24 -->> actual block 3
Read page success: auxiliary memory address 32 -->> actual block 4
Read page success: auxiliary memory address 40 -->> actual block 5
Read page success: auxiliary memory address 56 -->> actual block 7
Read page success: auxiliary memory address 72 -->> actual block 9
Read page success: auxiliary memory address 80 -->> actual block 10
Read page success: auxiliary memory address 88 -->> actual block 11
Read page success: auxiliary memory address 96 -->> actual block 12
Read page success: auxiliary memory address 112 -->> actual block 14
Read page success: auxiliary memory address 208 -->> actual block 26
Read page success: auxiliary memory address 216 -->> actual block 27
Read page success: auxiliary memory address 240 -->> actual block 30
```

图3. 3 页表初始化截图

```
Read page success: auxiliary memory address 216 -->> actual block 27
Read page success: auxiliary memory address 240 -->> actual block 30
Read page success: auxiliary memory address 248 -->> actual block 31
--------------------------------------------------------------------------
| oPageNO | iPageNO | blockNO | Filled | Changed | ProType | Cnt | Aux mem |
--------------------------------------------------------------------------
|       0 |       0 |       0 |      0 |       0 |     -wx |   0 |       0 |
|       0 |       1 |       1 |      1 |       0 |     -w- |   0 |       8 |
|       0 |       2 |       0 |      0 |       0 |     rw- |   0 |      16 |
|       0 |       3 |       3 |      1 |       0 |     r-x |   0 |      24 |
|       1 |       4 |       4 |      1 |       0 |     --x |   0 |      32 |
|       1 |       5 |       5 |      1 |       0 |     -w- |   0 |      40 |
|       1 |       6 |       0 |      0 |       0 |     --x |   0 |      48 |
|       1 |       7 |       7 |      1 |       0 |     rw- |   0 |      56 |
|       2 |       8 |       0 |      0 |       0 |     --x |   0 |      64 |
|       2 |       9 |       9 |      1 |       0 |     r-- |   0 |      72 |
|       2 |      10 |      10 |      1 |       0 |     rwx |   0 |      80 |
|       2 |      11 |      11 |      1 |       0 |     r-- |   0 |      88 |
|       3 |      12 |      12 |      1 |       0 |     --x |   0 |      96 |
|       3 |      13 |       0 |      0 |       0 |     rw- |   0 |     104 |
|       3 |      14 |      14 |      1 |       0 |     r-- |   0 |     112 |
|       3 |      15 |       0 |      0 |       0 |     -wx |   0 |     120 |
|       4 |      16 |       0 |      0 |       0 |     rwx |   0 |     128 |
|       4 |      17 |       0 |      0 |       0 |     r-x |   0 |     136 |
```

图3. 4 页表初始化截图

4	17	0	0	0	r-x	0	136
4	18	0	0	0	rwx	0	144
4	19	0	0	0	-w-	0	152
5	20	0	0	0	r-x	0	160
5	21	0	0	0	-wx	0	168
5	22	0	0	0	-w-	0	176
5	23	0	0	0	-w-	0	184
6	24	0	0	0	rwx	0	192
6	25	0	0	0	r-x	0	200
6	26	26	1	0	r--	0	208
6	27	27	1	0	rw-	0	216
7	28	0	0	0	--x	0	224
7	29	0	0	0	r--	0	232
7	30	30	1	0	r-x	0	240
7	31	31	1	0	r--	0	248
8	32	0	0	0	r--	0	256
8	33	0	0	0	r--	0	264
8	34	0	0	0	-wx	0	272
8	35	0	0	0	--x	0	280
9	36	0	0	0	rwx	0	288
9	37	0	0	0	-wx	0	296
9	38	0	0	0	rw-	0	304
9	39	0	0	0	--x	0	312
10	40	0	0	0	rwx	0	320
10	41	0	0	0	--x	0	328
10	42	0	0	0	r--	0	336
10	43	0	0	0	-w-	0	344
11	44	0	0	0	-wx	0	352
11	45	0	0	0	r--	0	360
11	46	0	0	0	-wx	0	368
11	47	0	0	0	rw-	0	376
12	48	0	0	0	r-x	0	384
12	49	0	0	0	r-x	0	392
12	50	0	0	0	--x	0	400
12	51	0	0	0	--x	0	408
13	52	0	0	0	-w-	0	416
13	53	0	0	0	rw-	0	424
13	54	0	0	0	-w-	0	432
13	55	0	0	0	-wx	0	440

图3. 5 页表初始化截图

3 通过进程间通信./response 进程相应访存请求。

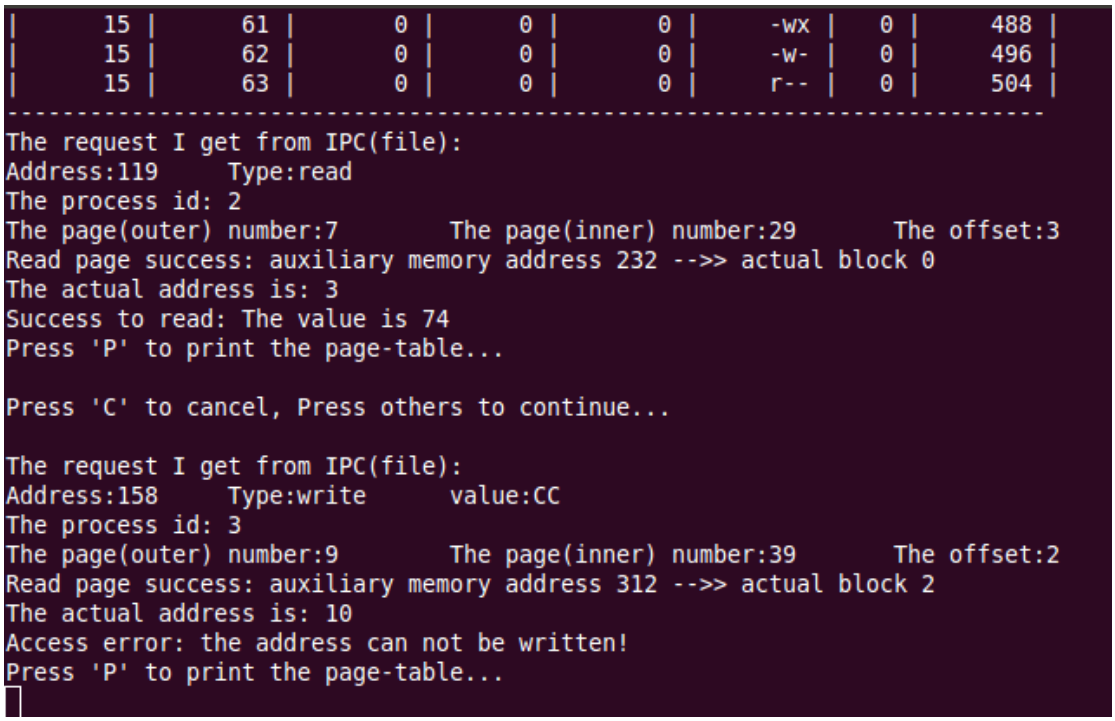

图3. 6 通过进程间通信获得另一进程发出的访存请求并处理

4 输入’P’，打印更新后的页表。

4	17	0	0	0	r-x	0	136
4	18	0	0	0	rwx	0	144
4	19	0	0	0	-w-	0	152
5	20	0	0	0	r-x	0	160
5	21	0	0	0	-wx	0	168
5	22	0	0	0	-w-	0	176
5	23	0	0	0	-w-	0	184
6	24	0	0	0	rwx	0	192
6	25	0	0	0	r-x	0	200
6	26	26	1	0	r--	0	208
6	27	27	1	0	rw-	0	216
7	28	0	0	0	--x	0	224
7	29	0	1	0	r--	1	232
7	30	30	1	0	r-x	0	240
7	31	31	1	0	r--	0	248
8	32	0	0	0	r--	0	256
8	33	0	0	0	r--	0	264
8	34	0	0	0	-wx	0	272
8	35	0	0	0	--x	0	280
9	36	0	0	0	rwx	0	288
9	37	0	0	0	-wx	0	296
9	38	0	0	0	rw-	0	304
9	39	2	1	0	--x	1	312
10	40	0	0	0	rwx	0	320
10	41	0	0	0	--x	0	328
10	42	0	0	0	r--	0	336
10	43	0	0	0	-w-	0	344
11	44	0	0	0	-wx	0	352
11	45	0	0	0	r--	0	360
11	46	0	0	0	-wx	0	368
11	47	0	0	0	rw-	0	376
12	48	0	0	0	r-x	0	384
12	49	0	0	0	r-x	0	392
12	50	0	0	0	--x	0	400
12	51	0	0	0	--x	0	408
13	52	0	0	0	-w-	0	416
13	53	0	0	0	rw-	0	424
13	54	0	0	0	-w-	0	432
13	55	0	0	0	-wx	0	440

图 3.7 打印更新后的页表截图

运行一定次数后，实存已满，需要进行页面替换。

```
|      14 |      58 |       0 |       0 |       0 |     rwx |   0 |     464 |
|      14 |      59 |       0 |       0 |       0 |     rw- |   0 |     472 |
|      15 |      60 |       0 |       0 |       0 |     r-- |   0 |     480 |
|      15 |      61 |       0 |       0 |       0 |     r-- |   0 |     488 |
|      15 |      62 |       0 |       0 |       0 |     -w- |   0 |     496 |
|      15 |      63 |       0 |       0 |       0 |     --x |   0 |     504 |
--------------------------------------------------------------------------
The request I get from IPC(file):
Address:119     Type:read
The process id: 2
The page(outer) number:7        The page(inner) number:29       The offset:3
The actual address is: 119
Success to read: The value is 74
Press 'P' to print the page-table...

Press 'C' to cancel, Press others to continue...

The request I get from IPC(file):
Address:158     Type:write      value:CC
The process id: 3
The page(outer) number:9        The page(inner) number:39       The offset:2
Please choose a method to do the page exchange algorithm,and press '1' for FIFO,
 '2' for LFU, '3' for LRU...
1
```

图3. 8 页面替换算法

输入 1，即使用 FIFO 算法替换页面。

```
The process id: 2
The page(outer) number:7        The page(inner) number:29       The offset:3
The actual address is: 119
Success to read: The value is 74
Press 'P' to print the page-table...

Press 'C' to cancel, Press others to continue...

The request I get from IPC(file):
Address:158     Type:write      value:CC
The process id: 3
The page(outer) number:9        The page(inner) number:39       The offset:2
Please choose a method to do the page exchange algorithm,and press '1' for FIFO,
 '2' for LFU, '3' for LRU...
1
There is no idle block.Do FIFO:
 Replace the 0th page.
Read page success: auxiliary memory address 312 -->> actual block 0
Page exchange Success!
The actual address is: 2
Access error: the address can not be written!
Press 'P' to print the page-table...
Press 'C' to cancel, Press others to continue...
```

图3. 9 页面替换算法

- 结果分析

1. 从图 3. 6 可以看出，./response 程序能打印出 process id，已经实现对“实现多道程序的存储控制”。在程序中，假定访问的进程数为 4，设定每个进程的访问虚存范围，从而根据访问的内存请求判断相应的进程 ID。

2. 从图 3.6 可以看出，./response 程序能打印出 The page(outer) number，实现了“建立一个多级页表或快表”。

从安装过程可以看出，已经实现“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”。其中 do_request 在./request 里面，do_response 在./response 里面。

3. 对比图 3.5 和图 3.7 可以看出，二级页表号为 29 与 39 的两项发生变化（由于访存请求地址是 119 与 158，对应虚存二级页表是 29、39），说明本程序运行结果是正确的。
4. 从图 3.8 和图 3.9 可以看出，已经实现“实现其它页面淘汰算法：先进先出（FIFO）、最近最久未使用淘汰算法（LRU）、最不频繁使用淘汰算法（LFU）”。

4 每个人的工作与会议记录

每次实验至少要开 4 次小组会议,每次会议都要记录以下内容:

a 说明每一位组员前一阶段完成的具体工作,是否按时按量完成任务。

b 说明每一位组员下一阶段需要完成的工作。

c 其他会议内容(如讨论的问题以及解决方案等)

4.1 会议时间表与会议记录

2012 年 3 月 26 日	虚存管理设计	黄建宇
2012 年 4 月 1 日	虚存管理文档书写	黄建宇
2012 年 4 月 3 日	虚存管理文档审查，修改	黄建宇
2012 年 4 月 6 日	虚存管理程序编写	黄建宇
2012 年 4 月 12-14 日	调试，debug，完善文档	黄建宇

每次会议都基本完成上次会议所计划内容，并规划下次完成进度。

5 其他说明

5.1 组内成员任务分工说明

(注明各个组员的工作量比例)

由于本组只有 1 人，工作量 100%。

5.2 实验完成部分说明

基本都已完成。

实现所有扩展要求：

- 实现多道程序的存储控制。
- 建立一个多级页表或快表。

- 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：最近最久未使用淘汰算法（LRU）、先进先出（FIFO）、最不频繁使用淘汰算法（LFU）。

6 程序清单

vmm_global.h	全局常量、变量、函数与接口
vmm_pagereplace.h	页面替换算法
vmm_request.c	访存请求进程对应的程序(do_request)
vmm_response.c	访存处理进程对应的程序(do_response)
aux_space	用文件模拟辅存
makefile	makefile 编译工程
vmm_fifo	建立的 ipc 机制文件：普通 file 或者 FIFO（命名管道）

7 实验心得

7.1 心得：

我感受到理论与实践之间的差距，在课堂上我感觉分页机制很简单，但到了 OS 课程设计实际模拟时，却遇到一个个困难。通过解决这一个个困难，我感觉我对课堂上的理论部分有了更深刻的理解。

特别感谢王雷老师精彩的授课，感谢王欢助教辛苦的讲解。最终我高效率高质量的完成的实验的要求。

7.2 建议：

建议老师多给一些相关的资料，比如初步介绍一下本实验要用到的相关函数。我看了《Unix 高级环境编程》才知道一些 FIFO（命名管道）的相关知识。