操作系统实验报告

实验三 虚存管理

学号	姓名
10231016	童浩（组长）
10231008	解佳琦
10231024	陈宇宁
38230213	邱伟豪

1 需求说明

1.1 基本需求

1.1.1 虚存管理模拟程序

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。

1.1.2 页表

页式存储管理技术用固定大小的页(Page)来描述逻辑地址空间，用相同大小的页框(Frame)来描述物理内存空间，由操作系统实现从逻辑页到物理页框的页面映射，同时负责对所有页的管理和进程运行的控制。

其中页表实现从页号到物理块号的地址映射。

逻辑地址转换成物理地址的过程是：用页号 p 去检索页表，从页表中得到该页的物理块号，把它装入物理地址寄存器中。同时，将页内地址 d 直接送入物理地址寄存器的块内地址字段中。这样，物理地址寄存器中的内容就是由二者拼接成的实际访问内存的地址，从而完成了从逻辑地址到物理地址的转换。

本程序要求实现一个一级页表。

1.1.3 访存请求

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。

1.1.4 响应请求

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。

1.1.5 页面淘汰算法

当没有空余的物理块之后，需要用页面淘汰算法来淘汰一些不经常访问的页，装入需要访问的页。

本程序要求实现 LFU 页面淘汰算法。

1.2 进阶需求

1.2.1 多道程序的存储控制

为实现多道程序的存储控制，我们使用程序的进程号来标志访存请求，用链表为每一道程序建立一个页表，当产生请求时，查询请求来自哪一个程序，然后进行相应的访存操作。

1.2.2 实现快表

建立快表结构，每次访存先查询快表，若找到则直接进行操作，更新快表的计数器。否则查询页表，并把查询到的放入到也表中。其中快表淘汰使用简单的淘汰计数器值最小的项算法。

1.2.3 将访存请求和响应请求实现到不同进程中

利用 FIFO 来在不同进城中实现访存请求的产生和响应。响应请求与虚存管理主程序在一起，首先建立一个 FIFO 文件，并通过循环从文件读取命令。

访存请求产生程序在程序开始打开 FIFO 文件，向其中写入请求。由此来进行进程间通讯。

1.2.4 实现 LRU 的页面淘汰算法

该算法是根据一个作业在执行过程中过去的页面踪迹来推测未来的情况。过去一段时间里不曾被访问过的页，在最近的将来可能也不再会被访问。所以当需要淘汰一页时，应选取在最近一段时间内最久未使用过的页面予以淘汰。

LRU 算法的实现方式是为每一页设定一个标志位，每当访问某一页时，将该页的标志位的值从 0 改变成 1，周期地检查每一页的标志位，看看哪些页被访问过，并把这些记到一个表格中，然后把各页的标志位清 0。当要淘汰某一页时，检查登记表格，淘汰醉酒不曾使用的页面。本程序中使用的是近似 LRU 算法，周期性地检查每一页的标志位，然后将页面使用次数计数器右移一位，将该页标志位放于计数器最高位，淘汰页面时，通过比较计数器的大小就可以决定淘汰哪一个页面。

1.3 自行改进

1.3.1 实现访存请求进程基本的命令行交互功能

在执行访存请求产生程序时，可以输入参数，选择是自动产生请求还是手动。如果是自动，需要输入进程个数，产生请求个数（默认为 10）；如果是手动，只能是一个进程产生请求，每次产生一个访存请求后都提示是否继续产生请求。

1.3.2 实现虚存管理模拟程序基本的命令行交互功能

在每收到一条访存请求并响应之后，程序会暂停询问用户选择

(1) 查看当前有多少程序的页表；

(2) 查看某程序的页表；

(3) 打印快表；

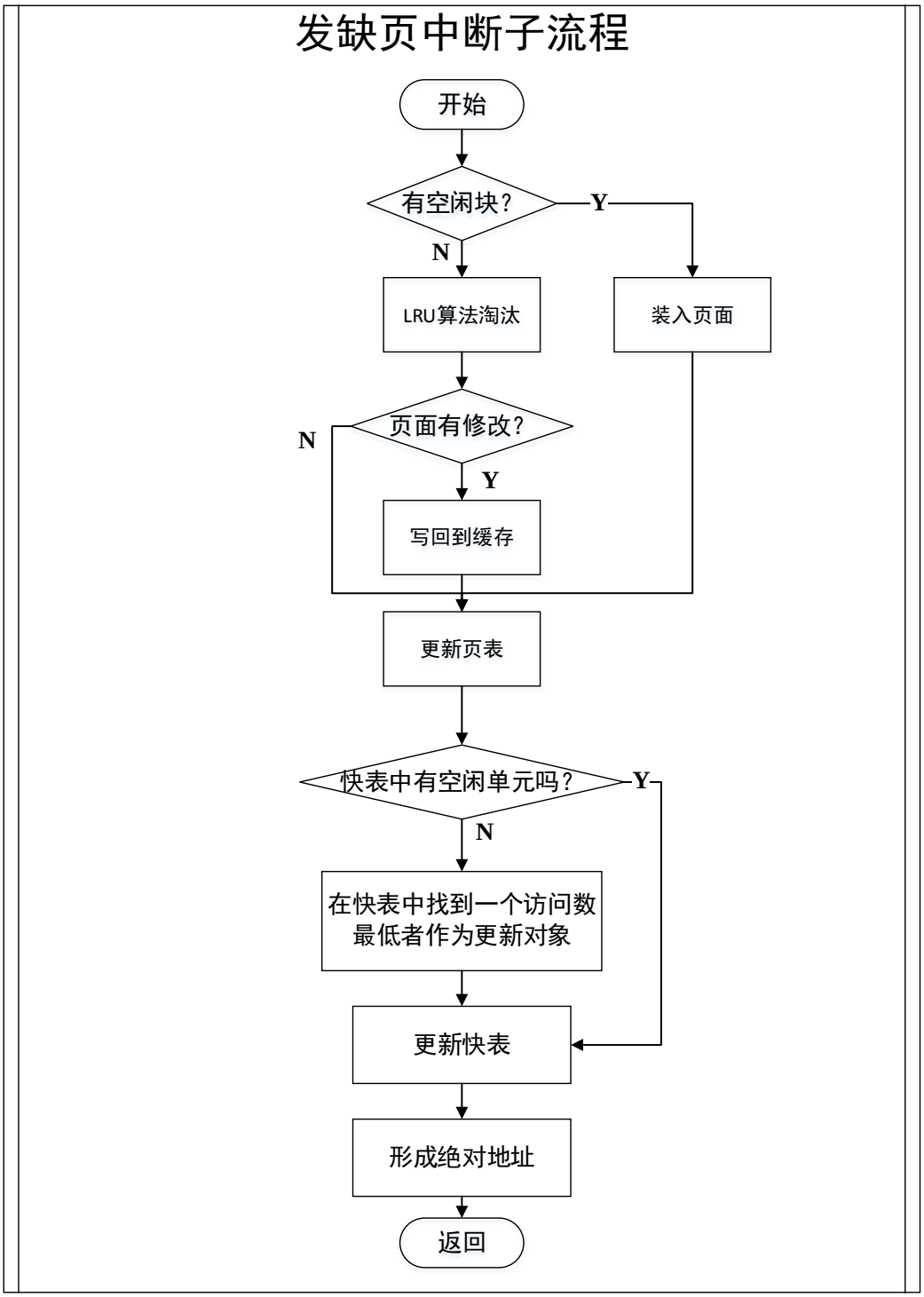
(4) 继续处理访存请求；

(5) 退出。

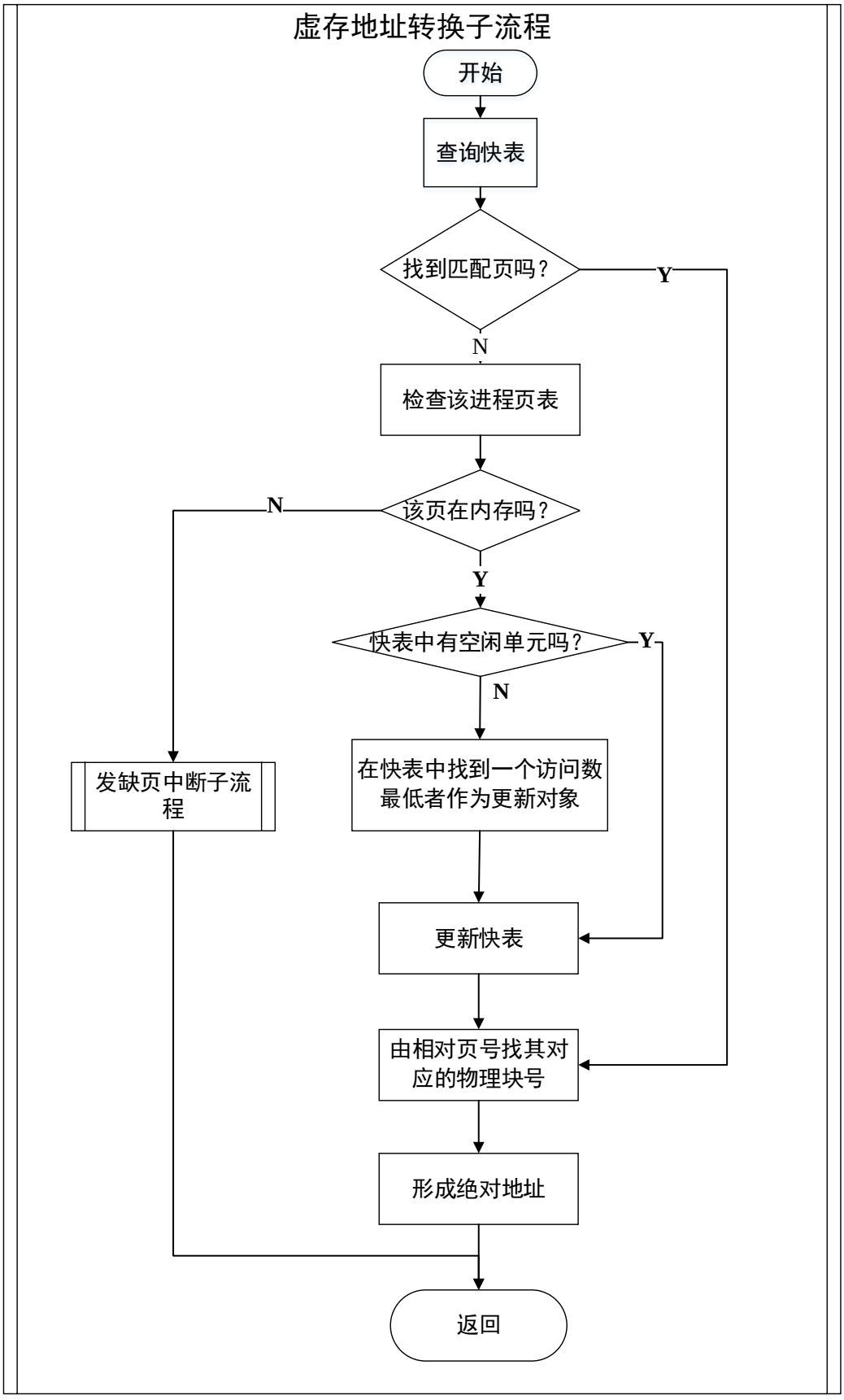
2 设计说明

2.1 结构设计

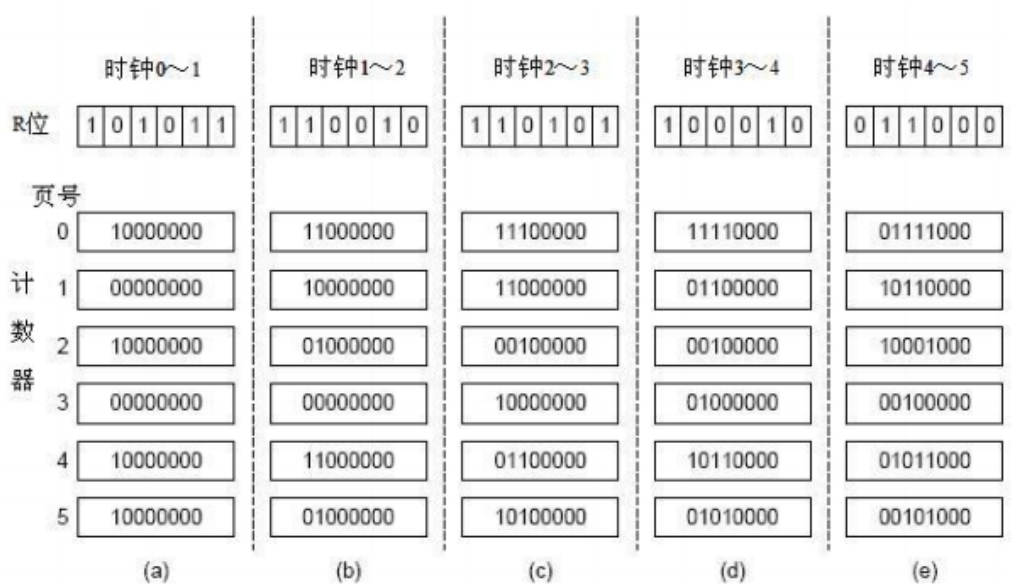
2.1.1 发缺页中断子流程



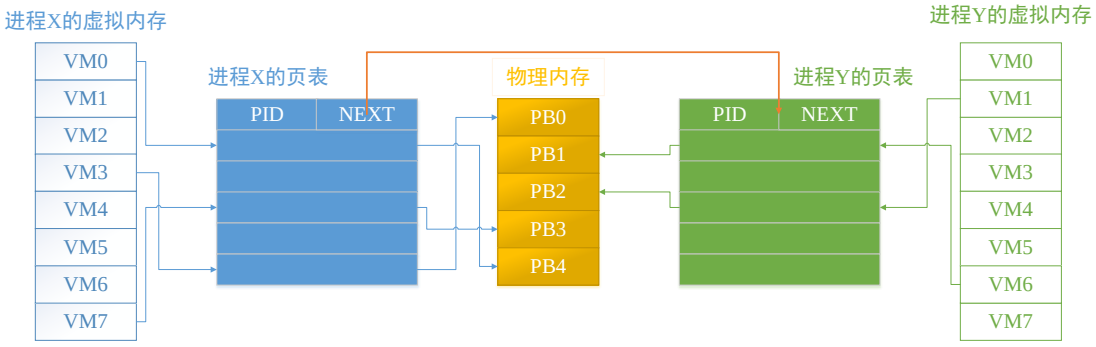
2.1.2 虚存快表地址转换子流程



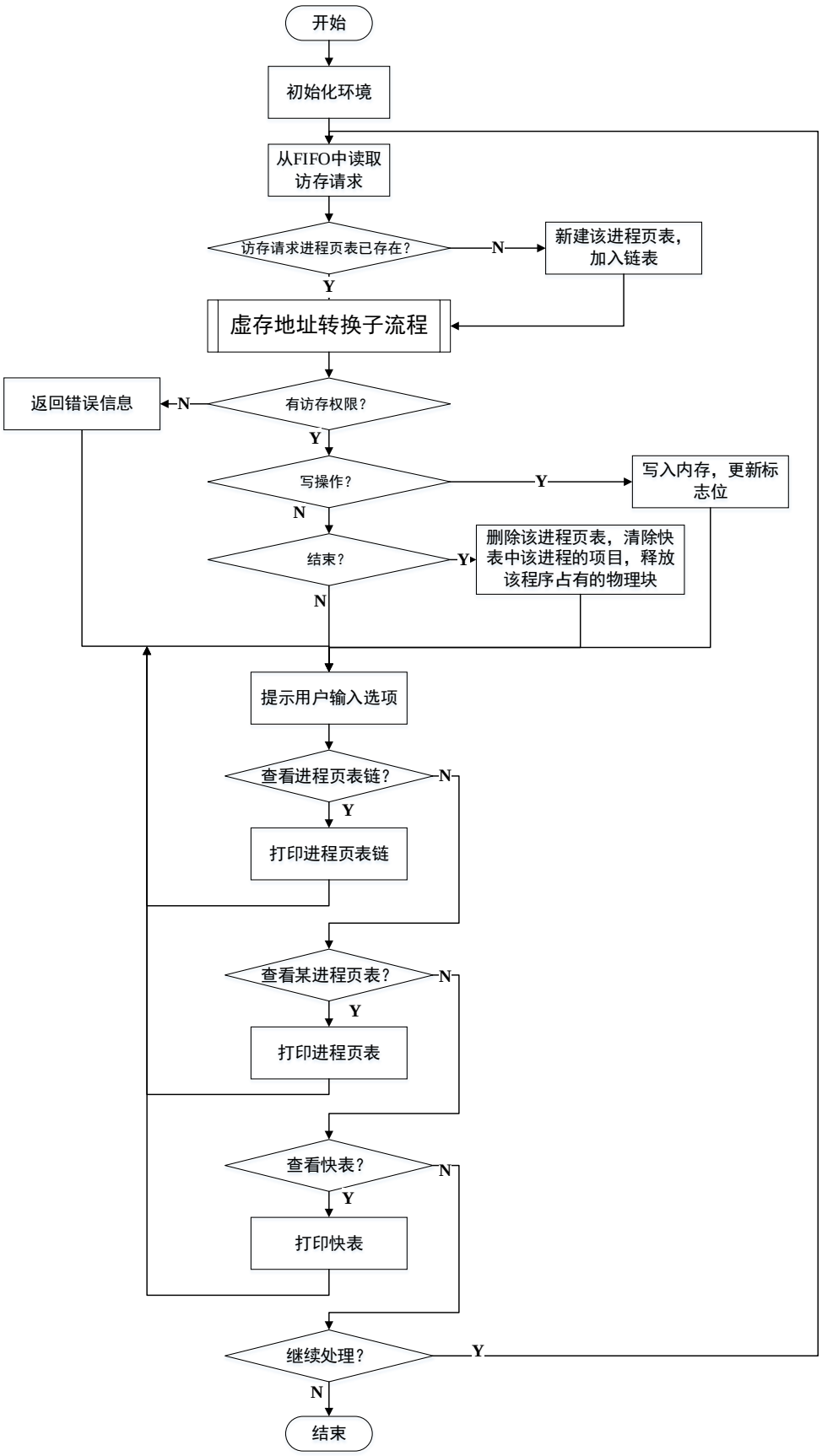
2.1.3 Linux 系统页面老化算法示意图



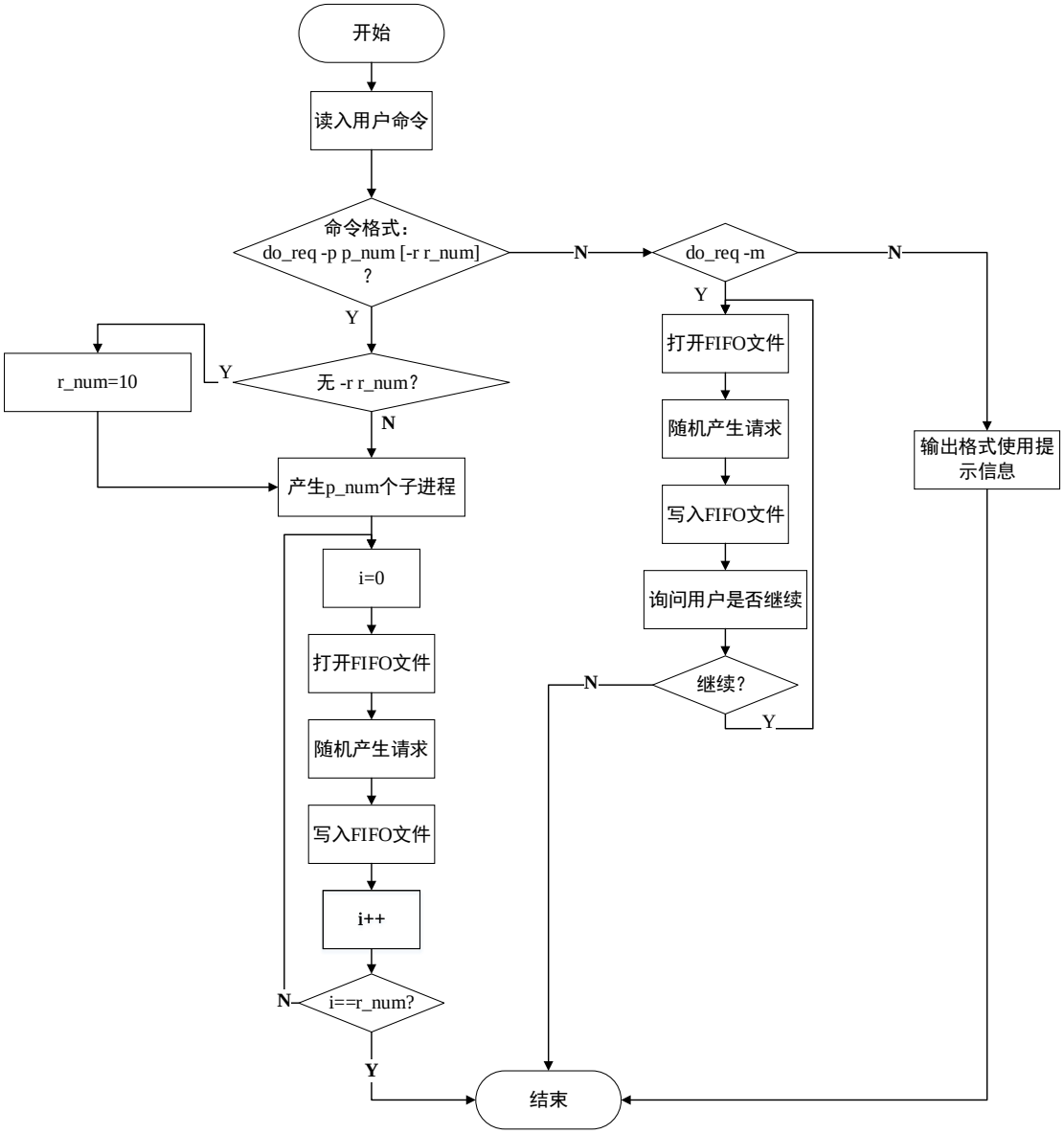
2.1.4 多道程序虚存管理示意图



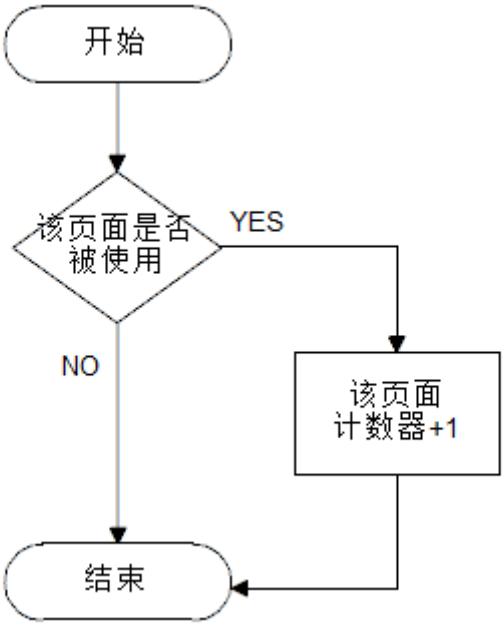
2.1.5 虚存管理模拟程序流程



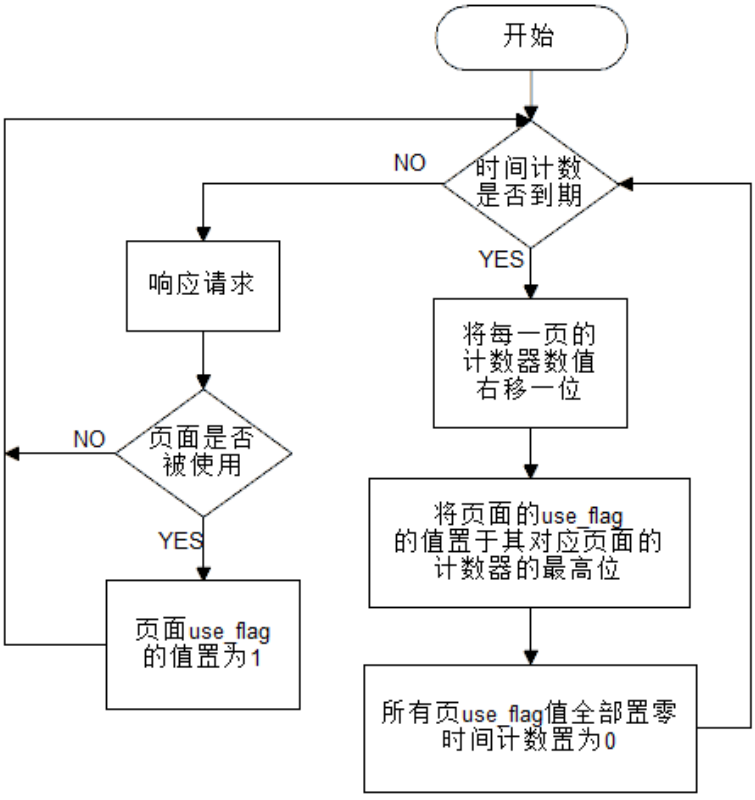
2.1.6 访存请求产生程序流程



2.1.6 LFU 算法流程图



2.1.7 LRU 算法流程图



2.2 功能设计

这里给出各功能模块的设计方案或实现方法。

2.2.1 重要的数据结构设计

(1) 页表项设计

```
typedef struct
{
    unsigned int pageNum;
    unsigned int blockNum;  //物理块号
    BOOL filled;                //页面是否装入实存
    BYTE proType;           //页面保护类型
    BOOL edited;            //页面修改标识
    int useflag;                //LRU算法使用标识
    unsigned long auxAddr;  //外存地址
    unsigned long count;        //页面使用计数器
} PageTableItem, *Ptr_PageTableItem;
```

(2) 快表设计

```
typedef struct
{
    pid_t pro;                  //程序号
    unsigned int pageNum;       //虚页号
    unsigned int blockNum;      //实存块号
    BOOL feature;           //标识是否正在被使用
    BYTE proType;           //保护类型
    unsigned long count;        //使用次数计数器
} TLBItem, *Ptr_TLBItem;
```

(3) 请求的结构设计

```
typedef struct
{
    pid_t pro;                          //发出请求的进程的进程号
    MemoryAccessRequestType reqType;    //访存请求类型
    unsigned long virAddr;              //外存地址
    BYTE value;                         //写请求的值
} MemoryAccessRequest, *Ptr_MemoryAccessRequest;
```

2.2.2 主要函数或接口设计

这里给出主要函数或接口的功能说明、实现方法和调用关系。

2.2.2.1 函数功能说明

文件 vmm.h vmm.c 中

(1) 函数原型：Ptr_ProPage init_page(int pro)

功能说明：为一个特定程序初始化一个页表

参数说明：pro: 要进行初始化的进程的进程号。

(2) 函数原型：void do_init()

功能说明：进行整个程序的初始化。

参数说明：无参数

(3) 函数原型：Ptr_TLBItem TLBLookUp(unsigned int pageNum,pid_t pro)

功能说明：查找一项记录是否在块表中

参数说明:

unsigned int pageNum: 虚存页号

pid_t pro: 进程号

(4) 函数原型：Ptr_TLBItem do_TLB_in(unsigned int pageNum,unsigned int blockNum,pid_t pro,int proType)

功能说明：将一项记录加入快表中

参数说明：

unsigned int pageNum: 虚存页号

unsigned int blockNum: 物理内存块号

pid_t pro: 进程号

int proType: 操作类型

(5) 函数原型：void do_delete(pid_t pro)

功能说明：结束一道程序，删除该程序的数据

参数说明:

pid_t pro: 进程号

(6) 函数原型：void do_response(pid_t pro)

功能说明：对请求进行响应

参数说明：

pid_t pro: 进程号

(7) 函数原型：int do_page_fault(Ptr_PageTableItem ptr_pageTabIt,pid_t pro)

功能说明：处理缺页中断

参数说明：

Ptr_PageTableItem ptr_pageTabIt: 程序的页表指针

pid_t pro: 进程号

(8) 函数原型：int do_LRU(Ptr_PageTableItem ptr_pageTabIt,pid_t pro)

功能说明：用 LRU 算法进行页面淘汰

参数说明：

Ptr_PageTableItem ptr_pageTabIt: 程序的页表指针

pid_t pro: 进程号

(9) 函数原型：void do_page_in(Ptr_PageTableItem ptr_pageTabIt, unsigned int blockNum)

功能说明：将辅存内容写入实存

参数说明：

Ptr_PageTableItem ptr_pageTabIt: 程序的页表指针

unsigned int blockNum: 物理块号

(10) 函数原型：void do_page_out(Ptr_PageTableItem ptr_pageTabIt)

功能说明：将实存内容写回辅存

参数说明：

Ptr_PageTableItem ptr_pageTabIt: 程序的页表指针

(11) 函数原型：void do_error(ERROR_CODE code)

功能说明：错误信息提示和处理

参数说明：

ERROR_CODE code: 错误类型

(12) 函数原型：void do_print_info(int pro)

功能说明：打印页表信息

参数说明：

int pro: 进程号

(13) 函数原型：char *get_proType_str(char *str, BYTE type)

功能说明：获取页面保护类型

参数说明：

char *str 要存入的字符串指针

BYTE type 保护类型

(14) 函数原型：Ptr_ProPage get_ptr_propage(int pro)

功能说明：查找某进程对应的表项

参数说明：int pro 进程号

(15) 函数原型：void do_print_TLB()

功能说明：打印快表内容

参数说明：无参数

(16) 函数原型：void do_print_process()

功能说明：打印所有进程信息

参数说明：无参数

文件 do_req.h do_req.c 中

(1) 函数原型：void do_request()

功能说明：随机产生访存请求

参数说明：无参数

(2) 函数原型：void do_over()

功能说明：显示一个进程结束的信息

参数说明：无参数

(3) 函数原型：void usage()

功能说明：显示程序输入格式信息

参数说明：无参数

(4) 函数原型：int digit(char* argv)

功能说明：将字符串转化为数字

参数说明：

char* argv: 数字字符串

(5) 函数原型：void createChild(int r_cnt)

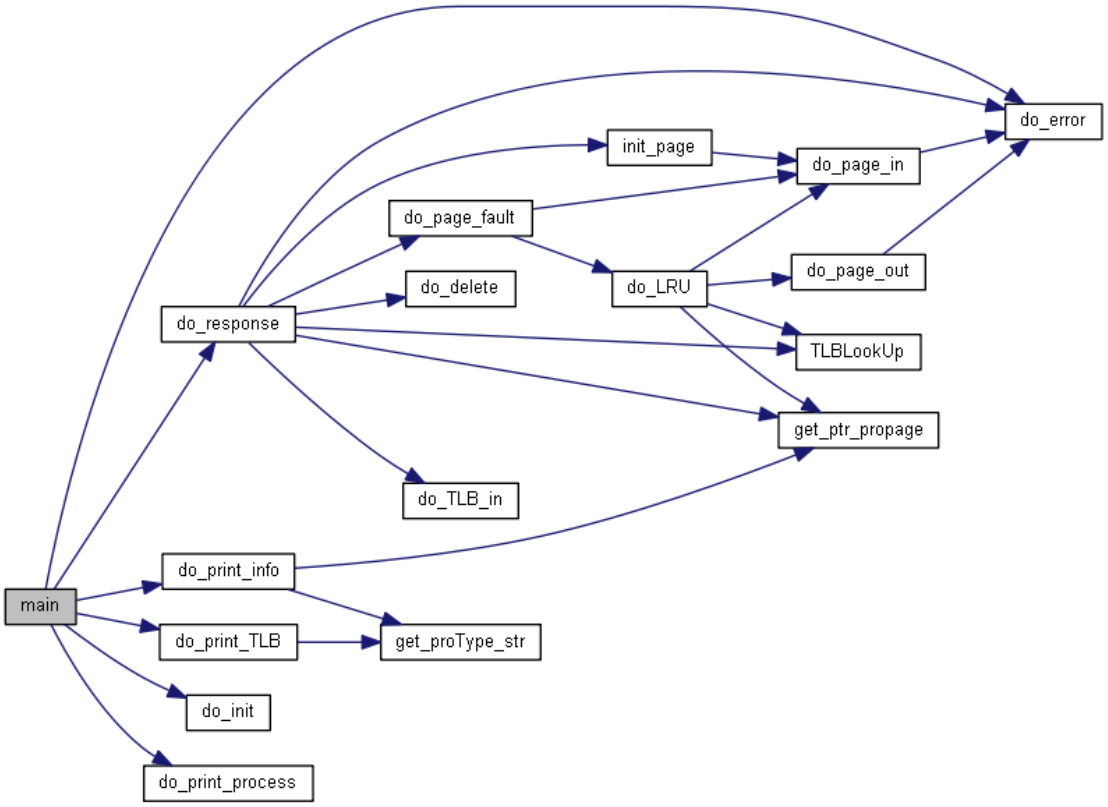
功能说明：创建新进程

参数说明:

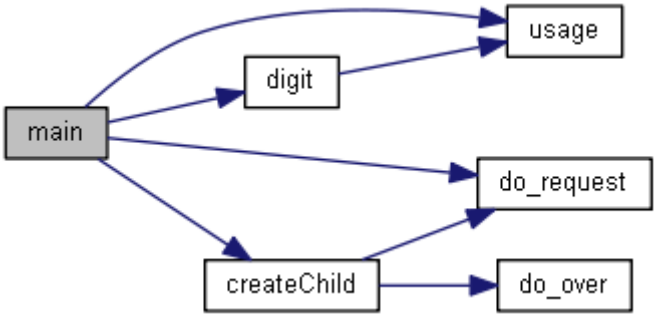
int r_cnt: 新进程要产生的请求数量

2.2.2.2 函数调用关系

(1) 虚存管理模拟程序 vmm 函数调用关系图



(2) 访存请求产生程序函数调用关系图



3 测试和使用说明

3.1 使用说明

列出程序的开发环境，如操作系统、使用的编程语言、开发工具和组件等。

列出程序的运行环境，如操作系统、必要的运行库等。

开发环境：

操作系统：Ubuntu12.04

编程语言：C 语言

开发工具：CodeBlocks

运行环境：

操作系统：Ubuntu 12.04

运行库： Gcc

3.2 测试说明

进入程序目录，执行命令：
./compile.sh
会生成两个可执行程序：vmm，do_req。
vmm 是虚存管理模拟程序，do_req 是请求生成程序。

(1) do_req 命令格式：

```
Usage: do_req (-p p_num [-r r_num] | -m)
        do_req  : App to generate requests.
        -p      : Process(es).
        p_num   : Digit only, number of processes (>0).
        -r      : Optional, request(s).
        r_num   : Digit only, number of requests (>0), default by 10.
        -m      : Manual mode, one process
```

(2) do_req 命令，产生 2 个进程，每个进程产生 5 个请求：

```
root@ubuntu:~/Desktop/vmm# ./do_req -p 2 -r 5
<<Process-4923>>GenReq-- Addr : 228[57:0 39:0]  Type : Execute
<<Process-4922>>GenReq-- Addr : 52[13:0 D:0]    Type : Write    Value : 7E
<<Process-4923>>GenReq-- Addr : 53[13:1 D:1]    Type : Write    Value : BB
<<Process-4922>>GenReq-- Addr : 195[48:3 30:3]  Type : Read
<<Process-4922>>GenReq-- Addr : 22[5:2 5:2]     Type : Write    Value : DA
<<Process-4923>>GenReq-- Addr : 95[23:3 17:3]   Type : Write    Value : 0B
<<Process-4923>>GenReq-- Addr : 185[46:1 2E:1]  Type : Execute
<<Process-4922>>GenReq-- Addr : 168[42:0 2A:0]  Type : Execute
<<Process-4923>>GenReq-- Addr : 151[37:3 25:3]  Type : Execute
<<Process-4922>>GenReq-- Addr : 186[46:2 2E:2]  Type : Execute
<<Process-4923>>GenReq-- Over
<<Process-4922>>GenReq-- Over
```

这里请求随机产生，每产生一个请求随机等待几毫秒的时间，从而两个进程随机交替发出请求。[]中，空格前是十进制显示，后面是十六进制显示，方面查看页表信息并对照。

(3) vmm 对第一个进程第一条请求的处理：

```
Proc[4923] Page No. : 57[0x39]  Offset : 0[0x0]
Load the page for process[4923]...
Paging Success : SecMem 24-->>Block 3
Paging Success : SecMem 32-->>Block 4
Paging Success : SecMem 48-->>Block 6
Paging Success : SecMem 56-->>Block 7
Paging Success : SecMem 80-->>Block 10
Paging Success : SecMem 96-->>Block 12
Paging Success : SecMem 112-->>Block 14
Paging Success : SecMem 120-->>Block 15
Paging Success : SecMem 152-->>Block 19
Paging Success : SecMem 160-->>Block 20
Paging Success : SecMem 168-->>Block 21
Paging Success : SecMem 184-->>Block 23
Paging Success : SecMem 192-->>Block 24
Paging Success : SecMem 200-->>Block 25
Paging Success : SecMem 208-->>Block 26
Paging Success : SecMem 224-->>Block 28
Page fault, begin paging...
Paging Success : SecMem 456-->>Block 0
Load page to TLB.
Real Address : 0[0x0]
Execute Success
OPTIONS:
        [PC]:           Print Process Chain.
        [P %pid]:       Print Page Table for Process %pid.
        [PT]:           Print TLB.
        [C]:            Continue.
        [E]:            Exit.
>>
```

首先，vmm 首先为进程新建一个页表，然后随机装入一些块。然后按请求查询页表，发生缺页中断，调入了新的块，并装入了 TLB，返回了实地址 0x0。最后给出提示信息查看相关信息。

进程页表链：

```
>>PC
Process chain: [4923]
```

目前只有一个进程。

当前进程的页表：

```
        Page Table for Process[4923]
P_No.   B_No.   Load    Mod     Pro     Cnt     SecMem
0       0       0       0       -wx     0       0
1       0       0       0       rw-     0       8
2       0       0       0       rwx     0       16
3       3       1       0       r--     0       24
57      0       1       0       -wx     2       456
58      0       0       0       r--     0       464
59      0       0       0       -w-     0       472
60      0       0       0       rw-     0       480
61      0       0       0       rw-     0       488
62      0       0       0       rw-     0       496
63      0       0       0       rw-     0       504
```

可以看到，页号 57 调入了物理块 0，cnt 也相应增大。

当前快表：

```
        Translation Lookaside Buffer
TLB_No. Proc    P_No.   B_No.   Feature Pro     Cnt
0       4923    57      0       1       -wx     1
1       0       0       0       0       ---     0
2       0       0       0       0       ---     0
3       0       0       0       0       ---     0
4       0       0       0       0       ---     0
5       0       0       0       0       ---     0
6       0       0       0       0       ---     0
7       0       0       0       0       ---     0
```

可以看到，页表内容被复制到快表中。

(4) vmm 对第二个进程第一条请求的处理：

输入 C 回车后，处理下一条请求：

```
>>c
Proc[4922] Page No. : 13[0xD]   Offset : 0[0x0]
Load the page for process[4922]...
Paging Success : SecMem 16-->>Block 2
Paging Success : SecMem 40-->>Block 5
Paging Success : SecMem 72-->>Block 9
Paging Success : SecMem 88-->>Block 11
Paging Success : SecMem 128-->>Block 16
Paging Success : SecMem 136-->>Block 17
Paging Success : SecMem 144-->>Block 18
Paging Success : SecMem 176-->>Block 22
Paging Success : SecMem 216-->>Block 27
Paging Success : SecMem 232-->>Block 29
Paging Success : SecMem 248-->>Block 31
Page fault, begin paging...
Paging Success : SecMem 104-->>Block 1
Load page to TLB.
Real Address : 4[0x4]
Access Failed : The address cannot be Written
```

为第二个进程初始化了页表，并发生缺页中断。

当前进程页表链：

```
>>pc
Process chain: [4923]-->[4922]
```

可以看到进程页表多了一个。

(5) vmm 对未发生缺页中断请求的处理：

```
Proc[4922] Page No. : 5[0x5]    Offset : 2[0x2]
Load page to TLB.
Real Address : 22[0x16]
Write Success
```

不提示缺页中断信息，但是快表中没有该页，所以将页装入到快表。

(6) vmm 对无空闲块的处理：

```
Proc[4922] Page No. : 42[0x2A]  Offset : 0[0x0]
Page fault, begin paging...
No free blocks, begin LRU...
Replace Page No. 22
Paging Success : SecMem 336-->>Block 22
Load page to TLB.
Real Address : 88[0x58]
```

没有空闲块，进行 LRU，替换了 22 号块。

(7) vmm 对结束请求的处理：

```
>>c
Process Complete, delete page table.
```

此时进程链：

```
>>pc
Process chain: [4922]
```

发现结束的进程已经从链中删除。

尝试打印进程 4923 的页表：

```
>>p 4923
No page for process[4923] found.
```

说明进程页表已经被删除。

此时的快表信息：

```
>>pt
        Translation Lookaside Buffer
TLB_No. Proc    P_No.   B_No.   Feature Pro     Cnt
0       4923    57      0       0       -wx     2
1       4922    13      1       1       r--     3
2       4923    13      8       0       -w-     2
3       4922    48      13      1       -w-     3
4       4922    5       5       1       rw-     3
5       4923    23      23      0       rwx     2
6       4923    46      30      0       r--     2
7       4922    46      18      1       rw-     1
```

有关进程 4923 的表项的特征位都被改成了 0。

(8) vmm 在快表中找到请求的页：

```
>>c
Proc[5066] Page No. : 57[0x39]  Offset : 0[0x0]
Found page in TLB.
Real Address : 40
Write Success
```

当前快表：

```
        Translation Lookaside Buffer
TLB_No. Proc    P_No.   B_No.   Feature Pro     Cnt
0       5066    44      18      1       r--     1
1       5066    35      1       1       r-x     1
2       5066    59      3       1       -w-     1
3       5066    16      16      1       rw-     1
4       5066    41      8       1       r-x     1
5       5066    2       2       1       rw-     1
6       5066    42      9       1       r-x     2
7       5066    57      10      1       rwx     2
```

可以看到，请求的页号为 57，为快表的最后一项，块号为 10，得到实地址为$(101000)_2$，所以真实地址为 40（十进制）。

其中页表中 57 页的内容为：

```
57      10      1       0       rwx     1073741824      456
```

(9) do_req 手动产生访存请求：

```
root@ubuntu:~/Desktop/vmm# ./do_req -m
<<Process-5054>>GenReq-- Addr : 109[27:1 1B:1]  Type : Execute
Press 'X' to exit, or other keys to continue...
```

产生了一条请求，并提示输入 X 退出。

vmm 反应如下：

```
Proc[5054] Page No. : 27[0x1B]  Offset : 1[0x1]
Load the page for process[5054]...
Page fault, begin paging...
No free blocks, begin LRU...
Replace Page No. 39
Paging Success : SecMem 216-->>Block 0
Load page to TLB.
Real Address : 1[0x1]
Access Failed : The address cannot be Executed
```

4 会议记录

会议次数	任务	子任务	本次内容	后期计划	会议日期
1	调试工程源代码	配置环境，调试成功，了解实验目的	讨论基础知识，了解实验基本流程	设计快表并实现	2013/5/17
2	设计快表	查阅资料，更改源程序以适应快表	实现快表并调试测试	实现多道程序的页表管理，改进页面淘汰算法	2013/5/19
3	实现多道程序的页表管理，改进页表淘汰算法	查阅资料，了解多道程序页表管理方式和 LRU 实现方法	实现多道程序页表管理和 LRU 淘汰算法	在另外一个程序中实现访存请求程序	2013/5/23
4	在另外一个程序中实现访存请求程序	为 vmm 和 do_req 加入一些交互命令	实现任务，整体测试程序	继续调试，排除 bug，准备检查	2013/5/27

5 其它说明

学号	姓名	分工情况	工作量比例
10231016	童浩（组长）	利用 FIFO 将访存请求实现在另一个程序中，更改程序以适应多道程序的管理，设计交互命令。编写文档。	40%
10231008	解佳琦	实现 LRU 算法，实现快表的管理过程。编写文档。	30%

10231024	陈宇宁	设计快表数据结构。编写文档（未完成）	10%
38230213	邱伟豪	实现多道程序数据结构的设计以及其他相关数据结构的更改。	20%

6 程序清单

6.1 源代码

（路径：~\源代码\vmm）

6.1.1 程序源文件

vmm.c
vmm.h
do_req.c
do_req.h

6.1.2 其他文件

compile.sh 编译并生成可执行文件的 shell 脚本
vmm_auxMem 辅存文件

6.2 可执行程序

（路径：~\可执行程序\）
vmm
do_req
vmm_auxMem 辅存文件