**第八次实验：存储管理模拟**

**一、实验目的与要求：**

通过模拟实现请求页式存储管理的几种基本页面置换算法，了解虚拟存储技术的特点，掌握虚拟存储请求页式存储管理中几种基本页面置换算法的基本思想和实现过并比较它们的效率。

**二、实验环境：（硬件环境、软件环境）**

1. LINUX操作系统和文本编辑器

**三、实验内容：（原理、操作步骤等）**

设计一个虚拟存储区和内存工作区，并使用下述算法计算访问命中率。

1、最佳淘汰算法（OPT）

2、先进先出的算法（FIFO）

3、最近最久未使用算法（LRU）

Solution：

I.我觉得图片中的程序有点问题，对其进行了修改，修改的部分用红色标注。

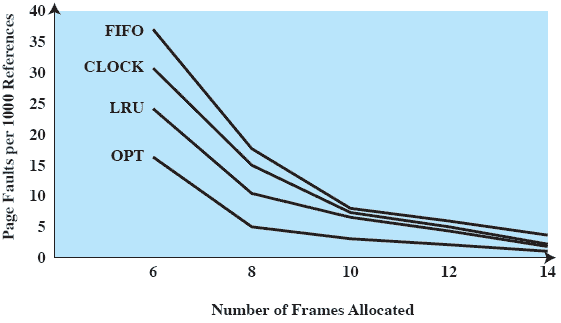
II.添加了轮转算法。

Analysis：

I.OPT算法缺页率最低，这是无疑的，其缺页率也比其它算法低很多。但是这个算法只是停留在理论层次，未来的块使用情况难以获得，难以实现。

II.经测试，先进先出算法的缺页率和轮转算法的缺页率基本相同，先进先出算法的缺页率和最近最久未使用算法非常接近。这是不合常理的，这可能与数据的设置不太合理有关。

IV.缺页率大致符合以下图片：



Code：

//虚拟存储不同置换算法的缺页率的比较

#include <stdlib.h>

#include <stdio.h>

#define pagesize 8 //size of page

typedef struct BLOCK //define a new structure : block

{

int pagenum; //页号

int accessed; //how long it did not been visited

}BLOCK;

int pc; //page number

//缺页：访问的页不在主存，需要os将其调入主存后再访问

int n; //amounts of missing page

static int num[320]; //store 320 random numbers

BLOCK block[pagesize];

void init();

int findExist(int curpage); //查找物理块中是否有该页面

int findSpace(); //查找是否有空闲物理块

int findReplace(); //查找应予置换的页面

void display(); //显示

void random\_init(); //产生随机数

void pagestring(); //显示调用的页面队列

void OPT();

void LRU();

void FIFO();

void CLOCK();

int main()

{

int select;

random\_init();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*corresponding page line\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

pagestring();

do

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*0:exit 1:OPT 2:LRU 3:FIFO 4:CLOCK\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*please choose a page replacement algorithm\*\*\*\*\n");

scanf("%d",&select);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

init();

switch(select)

{

case 1:

printf("optimal page replacement algorithm (OPT):\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

OPT();

break;

case 2:

printf("least recently used replacement algorithm (LRU):\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

LRU();

break;

case 3:

printf("first in frist out replacement algorithm (FIFO):\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

FIFO();

break;

case 4:

printf("CLOCK algorithm (CLOCK):\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

CLOCK();

break;

}

}

while (select!=0);

return 0;

}

void init()

{

int i;

for (i=0;i<pagesize;i++)

{

block[i].pagenum=-1;

block[i].accessed=0;

pc=n=0;

}

}

int findExist(int curpage)

{

int i;

for (i=0;i<pagesize;i++)

{

if (block[i].pagenum==curpage) //该块的框号对应的页号为curpage

return i;

}

return -1;

}

int findSpace()

{

int i;

for (i=0;i<pagesize;i++)

{

if (block[i].pagenum==-1) //该块的框号对应的页号为curpage，表示该块未被使用

return i;

}

return -1;

}

int findReplace()

{

int pos=0;

int i;

for (i=0;i<pagesize;i++)

{

if (block[i].accessed>block[pos].accessed) //找到accessed数值最大的块，accessed对应于不同置换算法有不同的含义

pos=i;

}

return pos;

}

void display()

{

int i;

for (i=0;i<pagesize;i++)

{

if (block[i].pagenum!=-1)

{

printf("%02d",block[i].pagenum); //输出所有被使用的块对应的页号

}

printf("\n");

}

}

//如下产生随机数的方法不行，如输入pc=0，就会出现错误

void random\_init()

{

int flag=0;

int i;

printf("Please input a random number: \n");

scanf("%d",&pc);

pc=pc%320;

printf("\*\*\*create 320 random numbers\*\*\*\n");

for (i=0;i<320;i++)

{

num[i]=pc;

if (flag%2==0)

pc=++pc%320;

if (flag==1)

pc=rand()%(pc-1);

if (flag==3)

pc=pc+1+(rand()%(320-(pc+1)));

flag=++flag%4;

printf("%03d ",num[i]);

if ((i+1)%10==0)

printf("\n");

}

}

void pagestring() //显示调用的页面队列，页面号取法为随机数除10取整

{

int i;

for (i=0;i<320;i++)

{

printf("%02d ",num[i]/10);

if ((i+1)%10==0)

printf("\n");

}

}

//最佳替换算法

void OPT()

{

int exist,space,position;

int curpage;

int i,j,k;

for (i=0;i<320;i++)

{

pc=num[i];

curpage=pc/10;

exist=findExist(curpage);

//如果该页不在内存中

if (exist==-1)

{

space=findSpace();

//如果有空闲的块，则该页放置在空闲块中，缺页次数加一

if (space!=-1)

{

block[space].pagenum=curpage;

display();

n=n+1;

}

//如果没有空闲的块，采用最佳替换算法

else

{

for (k=0;k<pagesize;k++)

{

block[k].accessed=1000; //如果将来没有用，设置为一个很大数

//获得块k未来第一次使用的位置

for (j=i+1;j<320;j++) //change

if (block[k].pagenum==num[j]/10)

{

block[k].accessed=j;

break;

}

}

position=findReplace();

block[position].pagenum=curpage;

display();

n++;

}

}

}

printf("OPT:\n");

printf("missing page times : %d\n",n);

printf("missing page rate : %f%%\n",(n/320.0)\*100);

}

//最近最久未使用算法

void LRU()

{

int exist,space,position;

int curpage;

int i,j;

for (i=0;i<320;i++)

{

pc=num[i];

curpage=pc/10;

exist=findExist(curpage);

//如果该页不在内存中

if (exist==-1)

{

space=findSpace();

//如果有空闲的块，则该页放置在空闲块中，缺页次数加一

if (space!=-1)

{

block[space].pagenum=curpage;

block[space].accessed=-1; //add

display();

n=n+1;

}

//如果没有空闲的块，采用最近最久未使用算法

else

{

position=findReplace(); //没有空闲物理块，进行置换

//未被访问时间最长的块

block[position].pagenum=curpage;

block[position].accessed=-1; //add

display();

n++;

}

}

else

block[exist].accessed=-1; //该块的未被访问时间置0(这里赋值位-1,后面加1变为0)

for (j=0;j<pagesize;j++)

block[j].accessed++; //经过一次页面请求，所有块的未被访问时间加1

}

printf("LRU:\n");

printf("missing page times : %d\n",n);

printf("missing page rate : %f%%\n",(n/320.0)\*100);

}

//先进先出算法

void FIFO()

{

int exist,space,position;

int curpage;

int i,j;

for (i=0;i<320;i++)

{

pc=num[i];

curpage=pc/10;

exist=findExist(curpage);

//如果该页不在内存中

if (exist==-1)

{

space=findSpace();

//如果有空闲的块，则该页放置在空闲块中，缺页次数加一

if (space!=-1)

{

block[space].pagenum=curpage;

display();

n=n+1;

}

//如果没有空闲的块，采用先进先出算法

else

{

position=findReplace(); //没有空闲物理块，进行置换

block[position].pagenum=curpage;

display();

n++;

block[position].accessed=-1; //该块的初始创建时间到当前的间隔时间置0(这里赋值位-1,后面加1变为0)

}

}

//把所有在页面里的页面号的访问次数加1

for (j=0;j<pagesize;j++)

block[j].accessed++; //经过一次页面请求，所有块的初始创建时间到当前的间隔时间加1

}

printf("FIFO:\n");

printf("missing page times : %d\n",n);

printf("missing page rate : %f%%\n",(n/320.0)\*100);

}

//轮转算法

void CLOCK()

{

int exist;

int curpage;

int i;

int used[pagesize];

int pointer=0; //指针指向的块的页号，默认为0

for (i=0;i<pagesize;i++)

used[i]=0; //初始块都设置为未被使用 0代表未被使用，1代表被使用

for (i=0;i<320;i++)

{

pc=num[i];

curpage=pc/10;

exist=findExist(curpage);

//如果该页不在内存中

if (exist==-1)

{

while (1)

{

if (used[pointer]==0)

{

block[pointer].pagenum=curpage; //放置页面到该块

used[pointer]=1; //该块置为被使用

pointer=(pointer+1)%pagesize; //指针后移一块

display();

break;

}

used[pointer]=0; //该块置为未被使用

pointer=(pointer+1)%pagesize; //指针后移一块

}

这里错了，只有上面的while循环执行pagesize次，才能说明所有的页都被占满了

//缺页次数加一

n++;

}

//否则指针位置不变，不用进行任何操作

}

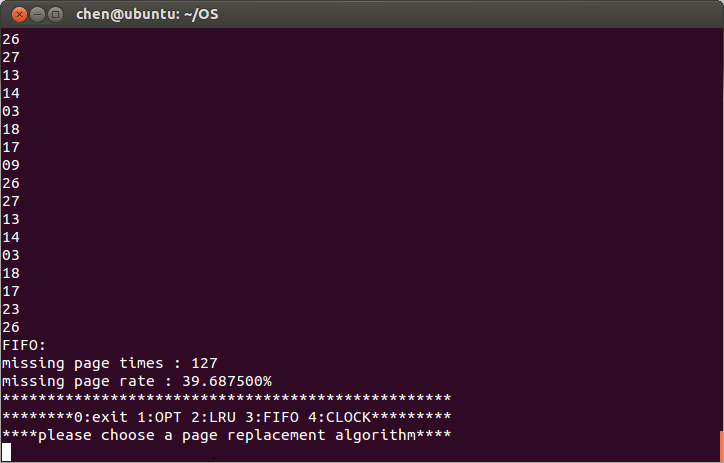
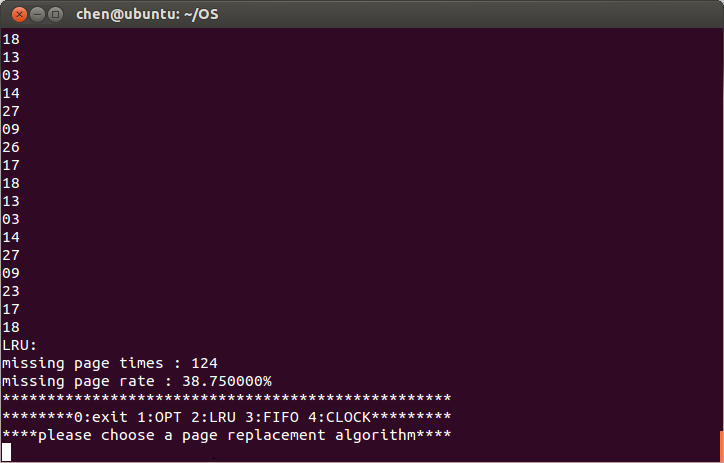
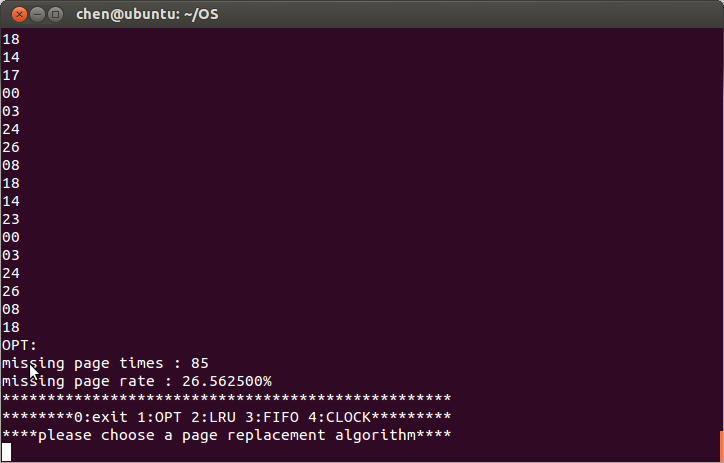
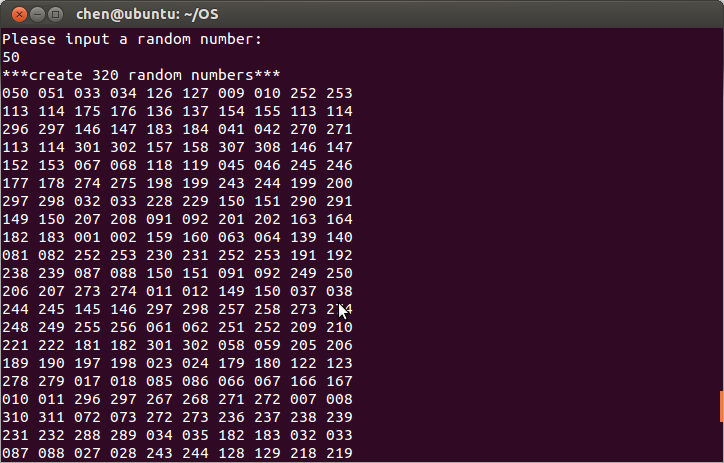
printf("CLOCK:\n");

printf("missing page times : %d\n",n);

printf("missing page rate : %f%%\n",(n/320.0)\*100);

}

Result：



换成另外一个随机数，数据有微小变化，而数据间的关系基本不变：

