#include "stdio.h"

#include "stdlib.h"

typedef struct item

{

int num; //页号

int time; //等待时间，LRU算法会用到这个属性

}Pro;

int pageNum; //系统分配给作业的主存中的页面数

int memoryNum; //可用内存页面数

void print(Pro \*page1); //打印当前主存中的页面

int Search(int num1, Pro \*memory1); //在页面集memory1中查找num1，如果找到，返回其在memory1中的下标，否则返回-1

int main(void)

{

int i;

int curmemory; //调入内存中的页面个数

int missNum; //缺页次数

float missRate; //缺页率

char c; //得到用户的输入字符，来选择相应的置换算法

Pro \*page; //作业页面集

Pro \*memory; //内存页面集

printf("输入系统分配给作业的主存中的页面数:");

scanf("%d", &pageNum);

printf("输入内存页面数:");

scanf("%d", &memoryNum);

page = (Pro\*)malloc(sizeof(Pro)\*pageNum);

memory = (Pro\*)malloc(sizeof(Pro)\*memoryNum);

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

{

printf("第 %d 个页面号为:", i);

scanf("%d", &page[i].num);

page[i].time = 0; //等待时间开始默认为0

}

do {

for (i = 0; i<memoryNum; i++) //初始化内存中页面

{

memory[i].num = -1; //页面为空用-1表示

memory[i].time = -1; //

}

printf("\*\*\*\*\*f:FIFO页面置换\*\*\*\*\*\n");

printf("\*\*\*\*\*o:OPT页面置换\*\*\*\*\*\n");

printf("\*\*\*\*\*l:LRU页面置换\*\*\*\*\*\n");

printf("\*\*\*\*\*请选择操作类型(f,o,l),按其它键结束\*\*\*\*\*\*\n");

//fflush(stdin);

getchar();

scanf("%c", &c);

i = 0;

curmemory = 0;

if (c == 'f') //FIFO页面置换

{

missNum = 0;

printf("FIFO页面置换情况: \n");

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

{

if (Search(page[i].num, memory)<0)//若在内存中没有找到该页面

{

missNum++;

memory[curmemory].num = page[i].num;

print(memory);

curmemory = (curmemory + 1) % memoryNum; //找出最先进入内存的页面

}

}//end for

missRate = (float)missNum / pageNum;

printf("缺页次数：%d 缺页率: %f\n", missNum, missRate);

}//end if

if (c == 'o') //OPT页面置换算法

{

missNum = 0;

curmemory = 0;

printf("Optimal页面置换情况: \n");

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

{

if (Search(page[i].num, memory) < 0)//若在内存中没有找到该页面

{

//找出未来最长时间内不再被访问的页面

int tem;

int opt = 0;

for (int k = 0; k < memoryNum; k++)

{

if (memory[k].num == -1)

{

curmemory = k;

break;

}

tem = 0; //页面k在未来tem时间内不会出现

int j;

for (j = i+1; j < pageNum; j++)

{

if (page[j].num == memory[k].num)

{

if (tem > opt)

{

opt = tem;

curmemory = k;

}

break;

}

else tem++;

}

if (j == pageNum)

{

opt = tem;

curmemory = k;

}

}

missNum++;

memory[curmemory].num = page[i].num;

print(memory);

}

}//end for

missRate = (float)missNum / pageNum;

printf("缺页次数：%d 缺页率: %f\n", missNum, missRate);

}//end if

if (c == 'l') //LRU页面置换算法

{

missNum = 0;

curmemory = 0;

printf("LRU页面置换情况: \n");

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

{

int rec=Search(page[i].num, memory);

if (rec < 0) //若在内存中没有找到该页面

{

missNum++;

for (int j = 0; j<memoryNum; j++) //找出最近最久未使用的页面

if (memory[j].time == -1) {

curmemory = j; break;

}

else if (memory[j].time > memory[curmemory].time)

curmemory = j;

memory[curmemory].num = page[i].num;

memory[curmemory].time = 0;

print(memory);

}

else memory[rec].time = 0;

for (int j = 0; j<memoryNum; j++) //内存中的所有页面等待时间+1

if (memory[j].num != -1)

memory[j].time++;

}//end for

missRate = (float)missNum / pageNum;

printf("缺页次数：%d 缺页率: %f\n", missNum, missRate);

}//end if

} while (c == 'f' || c == 'l' || c == 'o');

return 0;

}

void print(Pro \*memory1)//打印当前的页面

{

int j;

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

printf("%d ", memory1[j].num);

printf("\n");

}

//在页面集memory1中查找num1，如果找到，返回其在memory1中的下标，否则返回-1

int Search(int num1, Pro \*memory1)

{

int j;

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

{

if (num1 == memory1[j].num)

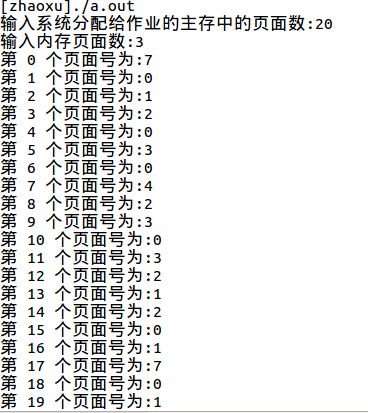
return j;

}

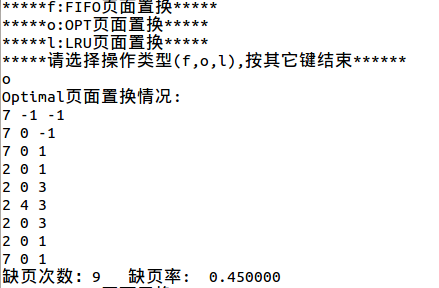
return -1;

}

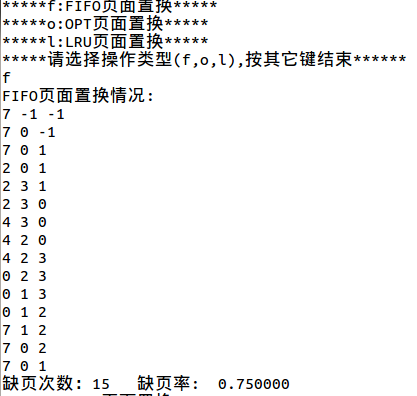
**假设一共有20个页面，内存页面容量为3，页面号顺序如下：**



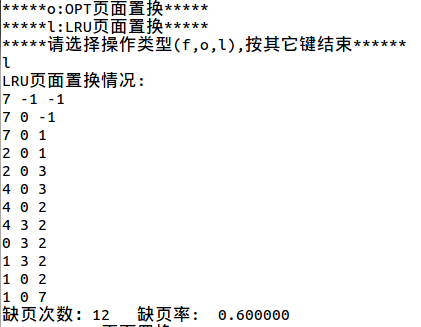
**选择最佳页面置换算法的情况：**



**选择先进先出的页面置换算法的情况：**



**选择最近最久未使用页面置换算法的情况：**



**总结：可见理想的OPT页面置换算法效率最高，其次是LRU算法。**