**动态页式存储管理的模拟实现实验报告**

课程名称 操作系统 信息技术与科学 分院信1605-3 班 组 桌号

实验者姓名 杨蕾 实验日期 2019 年 5 月 13 日

评分 教师签名

一、实验目的

动态页式存储管理是实现虚拟存储器的主要途径，对它的机制进行模拟编程实现，对于加强理解操作系统的存储管理功能具有重要意义。

二、实验要求

1.用C语言编写软件，实现上述要求；

2.软件具有模块化结构，应有比较友好的用户界面；

3.写出实验分析报告。

三、实验过程

1． 准备

A． 查阅相关资料；

通过看书对“最近没有使用页面淘汰算法NUR”和“理想型淘汰算法OPT”有更深的理解，了解其实现的机制，页面分配及回收算法。

B． 初步编写程序；

精读原理简述与图表描述，对NUR、OPT的实现过程有个初步的了解。memset作用是将某一块内存中的内容全部设置为指定的值， 这个函数通常为新申请的内存做初始化工作。memset(void \*s, int ch, size\_t n); 将s中当前位置后面的n个字节，用 ch 替换并返回 s 。

C． 准备测试数据；

进程个数为10，控制页面数为3，每个进程需要的页面为1 4 2 5 3 3 2 4 2 5

D． 运行。

2． 上机调试。

输入测试数据进行调试。

3． 主要流程和源代码。

主要流程：

使用数组定义页面访问号、内存等及定义页面个数、最大页面号基本要素，编写nru()及opt()算法，并通过主函数将函数进行串接。

源代码：

#include <iostream>

#include<map>

#include<set>

#include <algorithm>

#define N 50

using namespace std;

int page[N];//页面引用号

int block[N];//物理块，内存

int dist[N][N];//表示第i次访问内存的时候，内存中的页面j 在以后被访问的最小时间

int n;//页面引用号个数

int m;//物理块数目

int page\_max;//最大页面号

int nru[N];// i 最近时候被访问过

int page\_in\_block[N];//页面 i 在 block的下标索引

int NRU(){

int index = 1;

int page\_lack = 0;

memset(block, -1, sizeof(block));

for (int i = 1; i <= n; ++i){

if (page\_in\_block[page[i]]){// page[i]已经在内存中

nru[page\_in\_block[page[i]]] = 1;//重新标记被访问的页面 cout << endl << "第" << i << "次: 页面" << page[i] << "已经存在物理块" << page\_in\_block[page[i]] << "中" << endl;

}

else {

while (true){

if (index > m) index = 1;

if (block[index] == -1) {

nru[index] = 1;

page\_in\_block[page[i]] = index;

block[index++] = page[i];

++page\_lack;

break;

}

if (block[index] == page[i]){

nru[index++] = 1;

break;

}

else {

index = 1;

if (nru[index] == 0){//替换该页面

nru[index] = 1;

page\_in\_block[block[index]] = 0;

cout << endl << "第" << i << "次: 物理块" << index << "中的页面" << block[index] << "最近未被使用，将要被页面" << page[i] << "替换" << endl;

page\_in\_block[page[i]] = index;

block[index++] = page[i];

++page\_lack;

break;

}

else

nru[index++] = 0;

}

}

}

for (int k = 1; k <= m; ++k)

cout << block[k] << " ";

cout << endl;

}

return page\_lack;

}

int pre[N];//page[i]在page中的索引

int opt(){//最佳页面置换算法

int page\_lack = 0;

memset(pre, 0, sizeof(pre));

memset(dist, 0x3f, sizeof(dist));

memset(block, -1, sizeof(block));

for (int i = n; i >= 1; --i){

for (int j = 0; j <= page\_max; ++j)

if (pre[j])

dist[i][j] = pre[j] - i;

pre[page[i]] = i;

}

for (int i = 1; i <= n; ++i){

int j;

int max\_dist = 0, p;

for (j = 1; j <= m; ++j){

if (block[j] == -1){//直接放入, 产生缺页

block[j] = page[i];

page\_lack++;

break;

}

else if (block[j] == page[i])//存在内存中

break;

if (max\_dist < dist[i][block[j]]){

max\_dist = dist[i][block[j]];//block[j] 对应的页面以后会长时间不会用到

p = j;//block[] 第j个页面会被替换掉

}

}

if (j > m){//页面替换

cout << "页面" << page[i] << "不在内存，将物理块" << p << "中的页面" << block[p] << "替换" << endl;

block[p] = page[i];

page\_lack++;

}

//cout << endl << "当前内存中页面的情况:" << endl;

for (int k = 1; k <= m; ++k)

cout << block[k] << " ";

cout << endl << endl;

}

return page\_lack;//返回缺页次数

}

int main(){

cout << "请输入进程个数" << endl;

cin >> n ;

cout << "请输入内存可以控制的页面数" << endl;

cin>> m;

cout << "请输入" << n << "进程需要的页面数,以空格隔开"<<endl;

for (int i = 1; i <= n; ++i){

cin >> page[i];

page\_max = max(page\_max, page[i]);

}

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "最近没有使用页面淘汰算法NRU" << endl;

cout << "缺页中断次数:" << NRU() << endl;

cout << endl<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "理想型淘汰算法OPT" << endl;

cout << "缺页中断次数:" << opt() << endl;

system("pause");

return 0;

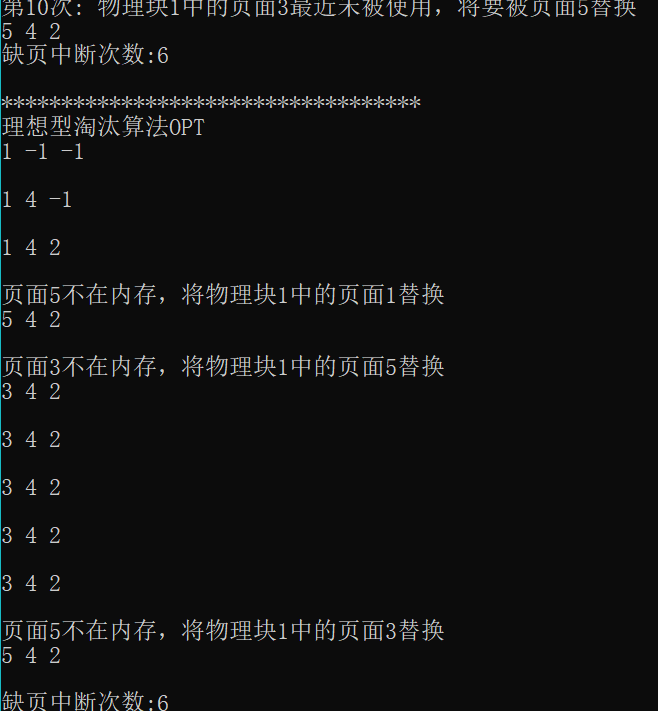
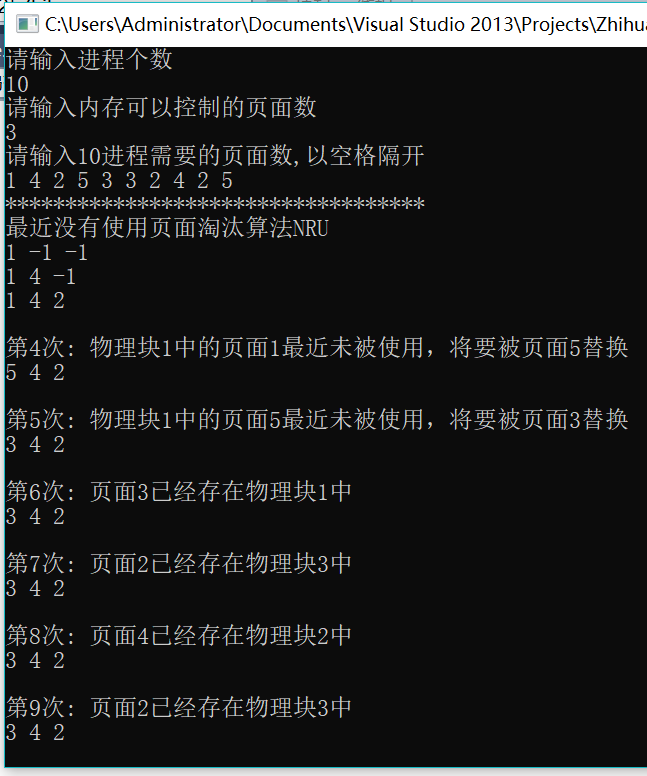
}

4． 遇到的主要问题和解决方法

A．对NRU的理解一开始不明确，有点像clock置换算法，之后发现他是改进的clock置换算法，有访问位及修改位。

B．对算法的实现，置换选择不知道这么执行，通过查询，使用map、set 、algorithm使实现变得简单。

四、实验结果



五、实验总结

最佳置换算法所选择的被淘汰页将是以后永不使用的，或许在最长未来时间内不被访问的页面，最近为用算法是只需为每页设置一位访问位，再将内存中所有页面都通过链接指针链接成一个循环队列，当某页被访问时，其访问位被置1，置换算法在选择一页淘汰时，只需检查页面的访问位，如果是0则换出。对算法原理转换为代码实现有了更深的理解，学会使用c++里的map、set、algorithm函数，进行页面的轮换换出存储选择。