操作系统

实验报告

实验名称：模拟页面置换算法

院系： 计算机学院·网络空间安全学院

班级：

学号：

姓名：

指导老师：匡林爱

日期：2020年11月10日

1. **实验目的和要求：**

(1) 设计并实现一个虚存管理程序，模拟一个单道程序的虚拟页式存储管理。

(2) 建立一张单级页表。

(3) 程序中使用随机数函数rand()产生的随机数作为要访问的虚地址，为简单起见，该随机数的最低位兼做修改标志。分配的主存块号也使用rand()产生。

(4) 实现函数response()响应访存请求，完成虚地址到实地址的定位，同时判断并处理缺页中断。

(5) 实现FIFO页面置换算法。

1. **实验内容：**

本实验程序在Windows 2000/XP+VC++6.0环境下开发，程序模拟了单个程序的内存访问控制过程。

数据结构

① 页表结构如下：

struct pagetable //页表结构

{

int sflag; //状态标志，"1"表示该页在内存

int framenum; //内存块号

int aflag; //访问标志，"1"表示该页已被访问过

int mflag; //修改标志，"1"表示该页已被修改过

};

其中状态标志"1"表示对应的页在内存，"0"表示对应的页不在内存；修改标志为"1"表示对应的页已被修改，当该页被淘汰时须存盘。修改标志为"0"表示对应的页未被修改，当被置换时该页可直接被淘汰，不需存盘。

② 用一维整型数组模拟页号队列(循环队列)。

循环队列：为充分利用向量空间，克服"[假溢出](https://baike.baidu.com/item/%E5%81%87%E6%BA%A2%E5%87%BA" \t "https://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E9%98%9F%E5%88%97/_blank)"现象的方法是：将向量空间想象为一个首尾相接的圆环，并称这种向量为循环向量。存储在其中的队列称为循环队列（Circular Queue）。循环队列是把[顺序队列](https://baike.baidu.com/item/%E9%A1%BA%E5%BA%8F%E9%98%9F%E5%88%97/20832734" \t "https://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E9%98%9F%E5%88%97/_blank)首尾相连，把存储队列元素的表从逻辑上看成一个环，成为循环队列。

1. **实验步骤：**

**主程序**

主程序(主函数)的总体流程如图1所示。

开始

环境初始化

产生访存请求

调用response()函数

处理访存请求

进程休眠一段时间

页面访问结束？

结束

否

是

图1主程序流程图

**通过随机数函数模拟内存块号**

通过随机函数rand()生成一个0~RAND\_MAX之间的随机数来模拟要访问的虚地址和分配的内存块号，RAND\_MAX的值在stdlib.h中定义为0x7fff(即32767)。在使用rand()之前须先调用srand()函数设置用于生成随机数的随机序列“种子”。通常种子可通过srand( (unsigned) time(NULL))的方式进行设置，其中函数time()返回1970年1月1日UTC时间0时0分0秒开始至今的秒数。

**通过函数response( )处理访存请求**

处理访存请的函数response( )完成的工作包括：分解虚地址为页号和页内偏移、判断是否地址越界、查找页表、处理缺页中断、必要时调用页面置换程序进行页面置换、修改页表、实现地址重定位等。简易流程图如图2所示。

分解虚地址为页号和页内偏移

地址越界？

查页表

页在内存？

由块号和页内偏移计算实地址

返回

N

Y

产生缺页中断

有空闲块？

淘汰最先进入内存的页面

页有修改？

写回辅存

转入新页

更新页表

Y

Y

N

N

图2 访存处理函数response()流程图

**进程休眠**

在主函数中，每次产生和相应访存请求后，要使进程休眠一段时间，从而模拟用户程序访存的不定时性。这里用到Windows的API函数Sleep()。

1. **实验源程序**

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

using namespace std;

struct pagetable //页表结构

{

int sflag; //状态标志，"1"表示该页在内存

int framenum; //内存块号

int aflag; //访问标志，"1"表示该页已被访问过

int mflag; //修改标志，"1"表示该页已被修改过

};

pagetable \*ptab; // 指向页表的指针

int psize = 4096; // 页的大小为4K字节

int PN; // 进程的页数

int BN; // 分配给该进程的内存块数

//int MB=65536; // 假设内存共有65536个块

int \*ps; // 指向页号的FIFO队列的指针

void Printptab() //输出(显示)页表

{

cout << "页号\t状态\t块号\t访问位\t修改位" << endl;

for (int i = 0; i < PN; i++)

{

cout << ' ' << i << "\t ";

cout << ptab[i].sflag << "\t ";

cout << ptab[i].framenum << "\t ";

cout << ptab[i].aflag << '\t';

cout << ptab[i].mflag << endl;

}

}

void FIFO(int vaddr, int &k) //FIFO页面置换函数

{

int b; // 内存块号

int pn; // 当前要访问的页号

int pp; // 将被淘汰的页号

int offset; // 页内偏移

unsigned int paddr; // 物理地址

int mf; // 修稿标志

mf = vaddr % 2; // 参数vaddr的最低位兼作修改标志

pn = vaddr / psize;

offset = vaddr % psize;

pp = ps[k]; // 取出将要被淘汰的页号

b = ptab[pp].framenum;// 取出被淘汰页所在的内存块号

if (ptab[pp].mflag == 1)

cout << "被置换的" << pp << "号页存盘后淘汰,";

else

cout << "被置换的" << pp << "号页直接被淘汰,";

// 修改页表

ptab[pp].sflag = 0; // 置pp页已不在内存的标志

ptab[pp].framenum = 0; // 操作系统实际上不作此置0工作

ptab[pp].aflag = 0; // 此处置0只是为了显示页表清楚

ptab[pp].mflag = 0; // 此处置0只是为了显示页表清楚

ptab[pn].sflag = 1; // pn页调入内存

ptab[pn].framenum = b;

ptab[pn].aflag = 1;

ptab[pn].mflag = mf;

ps[k] = pn; // 页号入队

k = (k + 1) % BN; // 调整循环队列ps的指针

paddr = b \* psize + offset; /\*形成物理地址\*/

cout << "虚地址" << vaddr << "所在的" << pn << "页调入" << b

<< "块中,对应实地址是" << paddr << endl; //输出绝对地址

}

bool response(int vaddr, int &n)

{

static int k = 0; // FIFO页号循环队列中，k所指的页是将被淘汰的页

static int bn = 0; // 已分配给该进程的内存块数

int b, bb; // b是申请得到的内存块号

int pn; // 当前要访问的页号

int offset; // 页内偏移

unsigned int paddr; // 物理地址

int mf; // 修改标志

mf = vaddr % 2; // 参数vaddr的最低位兼作修改标志

pn = vaddr / psize;

offset = vaddr % psize;

if (pn >= PN) // 地址越界

{

cout << "虚地址" << vaddr << "所在的页号为" << pn << "发生地址越界错误!" << endl;

return false;

}

if (ptab[pn].sflag == 1) // 所访问的页已在内存

{

cout << "虚地址" << vaddr << "所在的" << pn << "页已经在内存的"

<< ptab[pn].framenum << "块中，其对应的物理地址为 ";

paddr = ptab[pn].framenum\*psize + offset;

cout << paddr << endl;

if (ptab[pn].mflag == 0 && mf == 1)

ptab[pn].mflag = mf; // 修改页表中修改标志位

}

else // 产生缺页中断

{

if (bn < BN) // 尚有内存块可分配

{

bb = rand();

b = bb \* 2 + bb % 2; // 假设内存块号不超过0x7fff \* 2（即65535）

ps[bn] = pn; // 页号入队

bn++; // 已分配给该进程的内存块数增1

// 修改页表

ptab[pn].sflag = 1;

ptab[pn].framenum = b;

ptab[pn].aflag = 1;

ptab[pn].mflag = mf;

paddr = b \* psize + offset; // 计算对应的物理地址(重定位)

cout << "虚地址" << vaddr << "所在的" << pn << "页调入内存的" << b << "块中，";

cout << "其对应的物理地址为 " << paddr << endl;

}

else // 缺页中断中发生页面置换

{

n++; // 页面置换次数增1

FIFO(vaddr, k); // 采用FIFO页面置换算法

}

} // 缺页中断完成

cout << "页号队列：";

for (int i = 0; i < BN; i++)

cout << ps[i] << ' ';

cout << endl;

return true;

}

void init() // 初始化

{

int i;

ps = new int[BN]; // 分配FIFO页号队列空间

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

ps[i] = -1; // 初始化页号队列为空

ptab = new pagetable[PN]; // 分配页表空间

for (i = 0; i < PN; i++) // 初始化页表

{

ptab[i].sflag = 0;

ptab[i].mflag = 0;

ptab[i].aflag = 0;

ptab[i].framenum = 0;

}

}

int main()

{

int \*pqueue; // 指向页面引用串的指针

int L; // 该进程的页面引用串的长度

int nn = 0; // 用于统计页面置换次数

int i, vaddr, vd, pn;

bool flag;

srand((unsigned int)time(NULL)); // 置随机序列的“种子”(使用当前时间为种子)

cout << "请输入进程的页数：";

cin >> PN;

cout << "请输入分配给该进程的内存块数：";

cin >> BN;

cout << "请输入该进程的页面引用串的长度：";

cin >> L;

cout << endl;

init(); // 初始化

pqueue = new int[L]; // 分配页面引用串空间

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

pqueue[i] = -1; // 初始化页面引用串

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

{

vd = rand(); //访问串中页号由随机数模拟

vaddr = vd \* PN / 8; //rand()生成的随机数不超过0x7fff，相当于8页，需折合成PN页。

// vaddr=vd\*(PN+1)/8; //rand()生成的随机数不超过0x7fff，相当于8页，需折合成PN页。

//PN+1是为了使之可能产生“地址越界“”

pn = vaddr / psize; // 计算出页号

pqueue[i] = pn; // 该页号存入页面引用串中

flag = response(vaddr, nn);

if (!flag) //若发生地址越界，则结束程序

{

L = i + 1; //记下已访问的页数

break;

}

Sleep(3000 + (rand() % 5) \* 2000);

}

cout << "\n页面引用串为：";

for (i = 0; i < L - 1; i++)

cout << pqueue[i] << ',';

cout << pqueue[i] << endl;

cout << "页面置换次数为 " << nn << endl;

cout << "\n最终页表变为(不在内存的页对应的内存块号和修改标志无效)：" << endl;

Printptab(); //输出页表

delete[]ps;

delete[]ptab;

delete[]pqueue;

}

1. **实验结果**



1. **实验总结：**

这次实验主要是模拟了一个单道程序的虚拟页式存储管理过程，用到了一些函数比如rand()来实现随机访问，函数response()来响应访存的请求，主要运用的页面置换算法是FIFO算法，为了搞清楚这个算法，我去书上又重新复习了一遍该算法的相关知识，也加强记忆了该算法的优缺点。我觉得这次实验的重要部分是指导书中的页表结构，通过设置三个标志和一个内存块号来实现页表的基本构造，尤其是三个标志基本描述了该结构需要描述的大部分信息。还有一个比较重要的地方就是使用了随机种子，之前我还没明白随机数的设置其实不是随机的，在这次实验中我发现原来还可以通过时间来设置随机数，这给了我一些编程方面的启发。我还了解并学会使用了处理访存请求的函数response()，它主要的工作包括分解虚地址、页内偏移、查找页表、处理缺页中断、页面置换、修改页表以及判断是否越界等等，属实用处巨大。通过代码的debug过程也加深了我对本次实验代码的映像，总而言之，这次实验我学到了很多东西，对我的帮助巨大！