

**操作系统原理课程设计实验报告**

|  |  |
| --- | --- |
| 学生姓名 | 王云鹏 |
|  |  |
| 学生学号 | 8213180228 |
|  |  |
| 指导教师 | 胡小龙 |
|  |  |
| 专业班级 | 物联网1802 |
|  |  |
| 完成日期 | 2020.7.11 |

**计算机学院**

**目 录**

[实验 2](#_Toc500596551)

[一、目的与要求 2](#_Toc500596552)

[二、操作环境 2](#_Toc500596553)

[三、实验内容 2](#_Toc500596554)

[四、实验数据 2](#_Toc500596555)

[实验总结 2](#_Toc500596556)

[参考资料](#_Toc383552634) 3

# 实验

## 目的与要求

题目：页面置换算法模拟

目的：

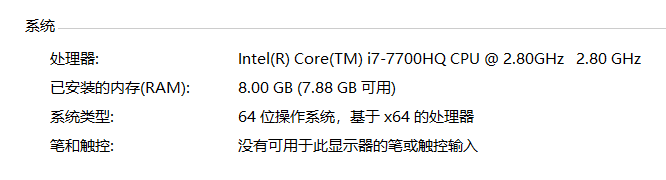
1. 增强学生对计算机操作系统基本原理、基本理论、基本算法的理解
2. 提高和培养学生的动手能力

要求：

1. 每人至少选作1题，多做不限。
2. 每人单独完成，可以讨论，但每人的设计内容不得完全相同，抄袭或有2人/多人设计完全一样者，不能通过。
3. 设计完成后，应上交课程设计文档，文档格式应是学校课程设计的标准格式，所有学生的封面大小、格式也必须一样
4. 同时上交设计的软盘(或以班刻录光盘)

## 二、操作环境

硬件：



软件：



Visual Studio Code

## 三、实验内容

## 

## 实验结果

不同算法对应的内存中页号：

## 

不同算法的命中率随着页面数的变化情况（测试序列）：

## 

## 五、实验源代码

*#include* <iostream>

*#include* <cstring>

using namespace std;

int test[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};

int test\_size = sizeof(test) / sizeof(int);

*// int page\_num=3;*

*// int memory[page\_num]={0};*

double hit\_rate(int miss\_num)

{

*return* 1 - (float)miss\_num / test\_size;

}

int is\_miss(int memory[], int size\_of\_memory, int order)

{*//检查order是否在memory中，是则返回位置,否则返回-1*

*for* (int i = 0; i < size\_of\_memory; i++)

    {

*if* (memory[i] == order)

        {

*return* i;

        }

    }

*return* -1;

}

void print\_array(int memory[], int size\_of\_memory)

{

*for* (int i = 0; i < size\_of\_memory; i++)

    {

        cout << memory[i] << " ";

    }

    cout << endl;

}

void init\_memory(int memory[], int memory\_size, int test[])

{

*for* (int i = 0; i < memory\_size; i++)

    {*//memory初始化*

        memory[i] = -1;

    }

*for* (int i = 0; i < memory\_size; i++)

    {

        memory[i] = test[i];*//预装*

*// print\_array(memory, memory\_size);*

    }

}

int find\_max\_index(int array[], int array\_size)

{

    int max\_index = 0;

*for* (int i = 0; i < array\_size; i++)

    {

*if* (array[i] > array[max\_index])

        {

            max\_index = i;

        }

    }

*return* max\_index;

}

class OPT

{

public:

    int miss\_count = 0;

    OPT(int memory\_size, int test[], int test\_size)

    {

*// cout<<"OPT内存中页号"<<endl;*

        int memory[memory\_size];*//主存*

        init\_memory(memory, memory\_size, test);

*for* (int i = memory\_size; i < test\_size; i++)

        {*//遍历test*

*if* (is\_miss(memory, memory\_size, test[i]) == -1)

            {*//若当前order不在主存中,寻找换入。若在,do nothing*

                miss\_count++;

*//寻找最久不会使用的页面*

                int flag[memory\_size];

*// int index\_long=-1;*

                memset(flag, 0, sizeof(flag));

                int count = 0;

*for* (int j = i; j < test\_size; j++)

                {

                    int index\_find = is\_miss(memory, memory\_size, test[j]);

*if* (index\_find != -1)

                    {

                        flag[index\_find] = 1;*//找到的下标flag都置为1*

                        count++;

                    }

*if* (count == memory\_size - 1)

                    {

*break*;*//剩下的那一个就是最久未使用的*

                    }

                }

*//换入*

*for* (int k = 0; k < memory\_size; k++)

                {

*if* (flag[k] == 0)

                    {

                        memory[k] = test[i];

*break*;

                    }

                }

            }

*// print\_array(memory, memory\_size);*

        }

*// cout << "OPT命中率" << hit\_rate(miss\_count) << endl;*

    }

};

class FIFO

{

public:

    int miss\_count = 0;

    FIFO(int memory\_size, int test[], int test\_size)

    {

*// cout<<"FIFO内存中页号"<<endl;*

        int memory[memory\_size];*//主存*

        init\_memory(memory, memory\_size, test);

        int times[memory\_size];

*for* (int i = 0; i < memory\_size; i++)

        {

            times[i] = memory\_size - i;

        }

*for* (int i = memory\_size; i < test\_size; i++)*//遍历test*

        {

*if* (is\_miss(memory, memory\_size, test[i]) == -1)*//若当前order不在主存中,寻找换入。若在,所有页面存在时间++*

            {

                miss\_count++;

                int max\_index = find\_max\_index(times, memory\_size);

                memory[max\_index] = test[i];

*for* (int j = 0; j < memory\_size; j++)

                {*//存在时间++*

                    times[j]++;

                }

                times[max\_index] = 1;*//换入的存在时间为1*

            }

*else*

            {

*for* (int j = 0; j < memory\_size; j++)

                {

                    times[j]++;

                }

            }

*// print\_array(memory, memory\_size);*

*// print\_array(times,memory\_size);*

        }

*// cout << "FIFO命中率" << hit\_rate(miss\_count) << endl;*

    }

};

class LRU

{

public:

    int miss\_count = 0;

    LRU(int memory\_size, int test[], int test\_size)

    {

*// cout<<"LRU内存中页号"<<endl;*

        int memory[memory\_size];*//主存*

        init\_memory(memory, memory\_size, test);

        int times[memory\_size];

*for* (int i = 0; i < memory\_size; i++)

        {

            times[i] = memory\_size - i;

        }

*for* (int i = memory\_size; i < test\_size; i++)*//遍历test*

        {

*if* (is\_miss(memory, memory\_size, test[i]) == -1)*//若当前order不在主存中,寻找换入。若在,在的时间置1，其他++*

            {

                miss\_count++;

                int max\_index = find\_max\_index(times, memory\_size);

                memory[max\_index] = test[i];

*for* (int j = 0; j < memory\_size; j++)

                {*//存在时间++*

                    times[j]++;

                }

                times[max\_index] = 1;*//换入的存在时间为1*

            }

*else*

            {

*for* (int j = 0; j < memory\_size; j++)

                {

                    times[j]++;

                }

                times[is\_miss(memory, memory\_size, test[i])] = 1;

            }

*// print\_array(memory, memory\_size);*

*// print\_array(times,memory\_size);*

        }

*// cout << "LRU命中率" << hit\_rate(miss\_count) << endl;*

    }

};

class CLOCK

{

public:

    int miss\_count = 0;

    CLOCK(int memory\_size, int test[], int test\_size)

    {

*// cout<<"CLOCK内存中页号"<<endl;*

        int memory[memory\_size];*//主存*

        init\_memory(memory, memory\_size, test);

        int flag[memory\_size];

*for* (int i = 0; i < memory\_size; i++)

        {

            flag[i] = 1;

        }

        int pointer = 0;*//下标指针（伪）*

*for* (int i = memory\_size; i < test\_size; i++)*//遍历test*

        {

*if* (is\_miss(memory, memory\_size, test[i]) == -1)*//若当前order不在主存中,寻找换入。若在,仅置位*

            {

                miss\_count++;

*for* (int j = 0; j <= memory\_size; j++)

                {

*if* (flag[pointer] == 1)

                    {

                        flag[pointer] = 0;

                        pointer = (pointer + 1) % memory\_size;*//移动指针*

                    }

*else*

                    {

                        memory[pointer] = test[i];

                        flag[pointer] = 1;

                        pointer = (pointer + 1) % memory\_size;

*break*;

                    }

                }

            }

*else*

            {

                flag[is\_miss(memory, memory\_size, test[i])] = 1;

            }

*// print\_array(memory, memory\_size);*

*// print\_array(flag,memory\_size);*

        }

*// cout << "CLOCK命中率" << hit\_rate(miss\_count) << endl;*

    }

};

int main()

{

    int page\_num = 3;

    OPT(page\_num, test, test\_size);

    FIFO(page\_num, test, test\_size);

    LRU(page\_num, test, test\_size);

    CLOCK(page\_num, test, test\_size);

cout<<"------------命中率--------------"<<endl;

cout<<"页面数\t"<<"OPT\t"<<"LRU\t"<<"CLOCK\t"<<"FIFO\t"<<endl;

*for* (int i = 1; i < test\_size; i++)

    {

        OPT opt=OPT(i,test,test\_size);

        LRU lru=LRU(i,test,test\_size);

        CLOCK clock=CLOCK(i,test,test\_size);

        FIFO fifo=FIFO(i,test,test\_size);

        cout<<i<<"\t"

        <<hit\_rate(opt.miss\_count)<<"\t"

        <<hit\_rate(lru.miss\_count)<<"\t"

        <<hit\_rate(clock.miss\_count)<<"\t"

        <<hit\_rate(fifo.miss\_count)<<"\t"<<endl;

    }

*return* 0;

}

# 实验总结

这次实验总体难度不是很大，需要实现的算法数目虽然不少，但基本思路较为相似，因此实现起来也并不是十分困难。通过完成这次实验，除了加深了我对几种策略的理解，锻炼了我的编程能力。通过阅读课件再加上自己的理解，我了解了老师的设计思路，感觉这个思路极其巧妙，设计中用到的方法和体现出的很多思想值得我们学习。

一开始做这个实验时，首先是看书，先把书上的替换算法知识点弄明白，要明白各种算法的优缺点和相互之间衍生互补关系。这四个算法中，难以实现的是LRU算法，因为它涉及到访问时间的计算，而且它的开销也比较大。OPT算法次难，它需要计算最近访问时间，并替换最近访问时间最大的页。而FIFO和CLOCK实现起来比较容易，FIFO算法的实现和CLOCK算法的实现很相似，FIFO可视为CLOCK的退化版。我先写了CLOCK算法，再删去一些约束条件就退化为FIFO算法。这就是两者的相同之处。

**参考资料**

**计算机操作系统第四版（汤小丹）**

**<https://www.cnblogs.com/fkissx/p/4712959.html>**

**<https://www.cnblogs.com/schips/p/10920145.html>**

**<https://blog.csdn.net/qq_41209741/article/details/99586257>**

**<https://blog.csdn.net/springtostring/article/details/85331177>**