**广州大学学生实验报告**

**开课学院及实验室：**计算机科学与工程实验室 **2019年11月11日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | **计算机科学与网络工程学院** | **年级/专业/班** | **计科173** | **姓名** | 谢绍波 | **学号** | 1706100109 |
| **实验课程名称** | **操作系统实验** | | | | | **成绩** |  |
| **实验项目名称** | 实验三 内存管理实验 | | | | | **指导老师** | 张汛涞 |

**一、实验目的及要求**

**二、实验设备与平台**

1. 实验设备：计算机；

2. 平台：Ubuntu虚拟机

**三、实验内容与步骤**

1、访问页面号以数字为模拟，输入数据从文本文件中读出，数字间以空格为间隔，假设有50个页面数据。

2、编写程序分别使用下述四个算法计算访问命中率：最佳淘汰算法（OPT），先进先出的算法（FIFO），最近最久未使用算法（LRU），最不经常使用算法（LFU）。

3、输出访问顺序和命中率结果。

1. 返回已分配给变量的内存地址；
2. 返回释放后的内存地址；
3. 释放已分配的内存空间后，返回释放内存后未使用内存的大小。

四、实验程序

（一）

我的代码：

#include<iostream>

#include<fstream>

#include<vector>

#include<set>

#include<map>

#include<string>

#include <algorithm>

using namespace std;

void read\_xsb();

void OPT\_xsb();

void FIFO\_xsb();

void LRU\_xsb();

void LFU\_xsb();

int kuaishu;        //内存块数

int len;            //数据长度

int \*data;          //数据

int main(){

puts("计科173谢绍波1706100109");

    read\_xsb();

}

void read\_xsb(){

    fstream input("data.txt",ios::in);

    if(!input){

        cerr << "File open error!\n";

    }else{

        int n;

        input >> n;

        while(n--){

            input >> kuaishu;

            input >> len;

            data = new int[len];

            for(int i = 0; i < len; i++){

                input>>data[i];

            }

            for(int i = 0; i < len; i++){

                cout << data[i]<<" ";

            }

            cout << endl;

            OPT\_xsb();

            FIFO\_xsb();

            LRU\_xsb();

            LFU\_xsb();

        }

    }

    input.close();

}

void OPT\_xsb(){

    /\*

    思路：

    if 页框内是否有：

        不变

    elif 空槽：

        填入页框，缺页加一

    else：

        依次查看后面数字，与页框内相同的数在后多少位

        if 找不到:

            直接替换

        elif 大于max:

            max更新

        置换max

        缺页加一

    \*/

   vector<int>memory;

   int count = 0; //缺页数

   for(int i=0;i<len;i++){

       int item = data[i];

       auto it = find(memory.begin(),memory.end(),item);

       if(it!=memory.end()){    //内存中已经存在

          //什么都不变

       }else if(memory.size()<kuaishu){

           memory.push\_back(item);

        //    count++;

       }else{

           count++;

           int max=i;

           int change\_num=0;

           bool found=false;

           for(int j = 0;j<memory.size();j++){

               for(int k =i+1;k<len;k++){

                   if(memory[j]==data[k]){

                       found=true;

                       if(k>max){max=k;change\_num=j;}

                       break;

                   }

               }

               if(!found){

                       change\_num=j;

                       j=memory.size();

                   }

               found=false;

           }

           memory[change\_num]=item;

       }

    //    for(auto val:memory){

    //        cout<<val<<" ";

    //    }

    //    cout<<endl;

   }

   cout<<"\nOPT 缺页数:"<<count<<"\t访问命中率:"<<float(1-float(count)/float(len))<<endl;

}

void FIFO\_xsb(){

    /\*

    思路：

    if 页框内是否有：

        不变

    // elif 空槽：                          //我们留意到，队列的弹出和添加和缺页数有关，即存在

    //     填入页框，缺页加一                //这次队列要替换的就是，第 缺页数%物理块号 个

    //     把填入页框号放入队列中            //所以不需要用到队列，只需要缺页数就行了。

    // else：

    //     弹出队列头

    //     填入队列头对应的页框号中。

    elif 空槽：

        填入页框，缺页加一

    else:

        缺页+1

        替换缺页数%物理块号

    \*/

   vector<int>memory;

   int count = 0; //缺页数

    for(int i=0;i<len;i++){

        int item = data[i];

        auto it = find(memory.begin(),memory.end(),item);

        if(it!=memory.end()){    //内存中已经存在

            //什么都不变

        }else if(memory.size()<kuaishu){

            memory.push\_back(item);

            count++;

        }else{

            count++;

            memory[count%kuaishu]=item;

        }

    }

    cout<<"FIFO\_xsb 缺页数:"<<count-kuaishu<<"\t访问命中率:"<<float(1-float(count-kuaishu)/float(len))<<endl;

}

void LRU\_xsb(){

    /\*

    思路:

    LRU就是向前版OPT

    \*/

   vector<int>memory;

   int count = 0; //缺页数

   for(int i=0;i<len;i++){

       int item = data[i];

       auto it = find(memory.begin(),memory.end(),item);

       if(it!=memory.end()){    //内存中已经存在

          //什么都不变

       }else if(memory.size()<kuaishu){

           memory.push\_back(item);

       }else{

           count++;

           int min;

           int change\_num=0;

           set<int>temp;

           for(int j=i-1;j>=0;j--){

               temp.insert(data[j]);

               if(temp.size()==kuaishu){

                   min=data[j];

               }

           }

           for(int j=0;j<memory.size();j++){

               if(memory[j]==min){

                   memory[j]=item;

                   break;

               }

           }

       }

    }

    cout<<"LRU\_xsb 缺页数:"<<count<<"\t访问命中率:"<<float(1-float(count)/float(len))<<endl;

}

void LFU\_xsb(){

    vector<int>memory;

    int count = 0; //缺页数

    map<int,int>frequency;

    for(int i=0;i<len;i++){

        int item = data[i];

        auto it = find(memory.begin(),memory.end(),item);

        if(it!=memory.end()){   //内存中已经存在

            frequency[item]++;

        }else if(memory.size()<kuaishu){

            memory.push\_back(item);

            frequency[item]++;

        }else{

            count++;

            int min=99999;

            int change=0;

            int change\_num=0;

            for(auto it=frequency.begin();it!=frequency.end();it++){

                if(it->second<min){

                    min=it->second;

                    change=it->first;

                }

            }

            for(int j=0;j<memory.size();j++){

                if(memory[j]==change){

                    memory[j]=item;

                    break;

                }

            }

            frequency[item]++;

        }

    }

    // for(auto it=frequency.begin();it!=frequency.end();it++){

    //     cout<<it->first<<" "<<it->second<<endl;

    // }

    cout<<"LFU\_xsb 缺页数:"<<count<<"\t访问命中率:"<<float(1-float(count)/float(len))<<"\n\n";

}

（二）

#include<stdlib.h>

#include<stdio.h>

#include<malloc.h>

#include<iostream>

using namespace std;

int main(){

    puts("计科173谢绍波1706100109");

    int \*string;

    string = (int \*)malloc(10);

    if(string == NULL) printf("Insufficient memory available\n");

    else{

        printf("Memory space allocated for path name\n");

        cout<<"string="<<string<<endl;

        free(string);

        printf("Memory freed\n");

    }

    int \*stringy;

    stringy = (int\*)malloc(12);

    if(string==NULL){

        printf("Insufficient memory available\n");

    }else{

        printf("Memory space allocated for path name\n");

        cout<<"stringy="<<stringy<<endl;

        free(stringy);

        printf("Memory freed\n");

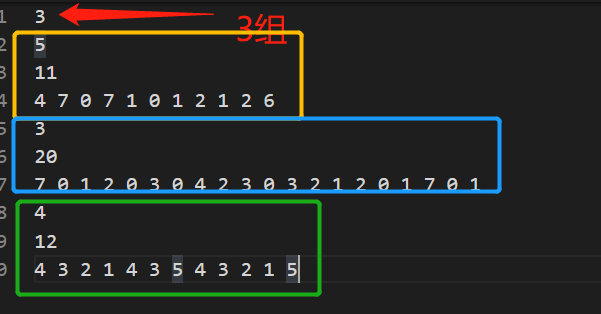
    }

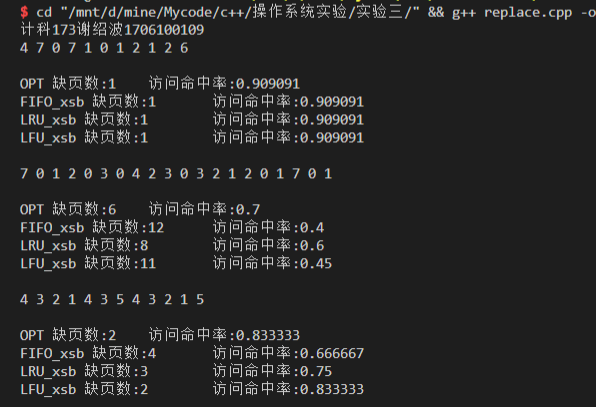
}

五、验证数据和运行结果

数据：

（这里是把三个data，放在一起，一并读取。）

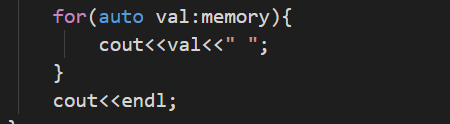




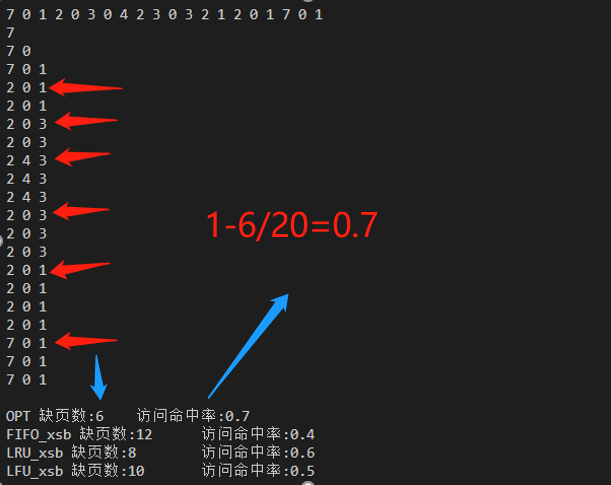
我们就对其中两个做展开的分析：

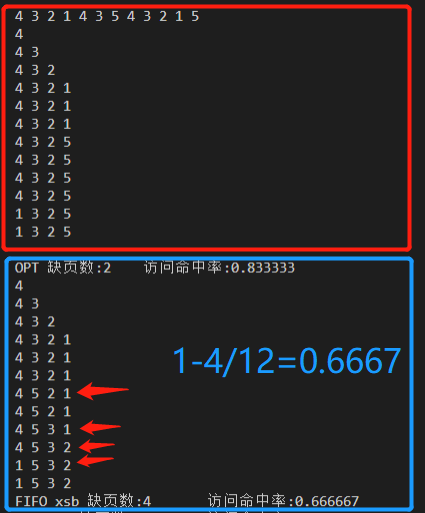
（其他是同样，将这段代码加上，就可以了。）

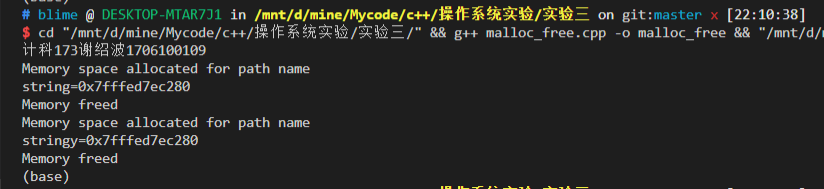
（关键代码如下：）



OPT：

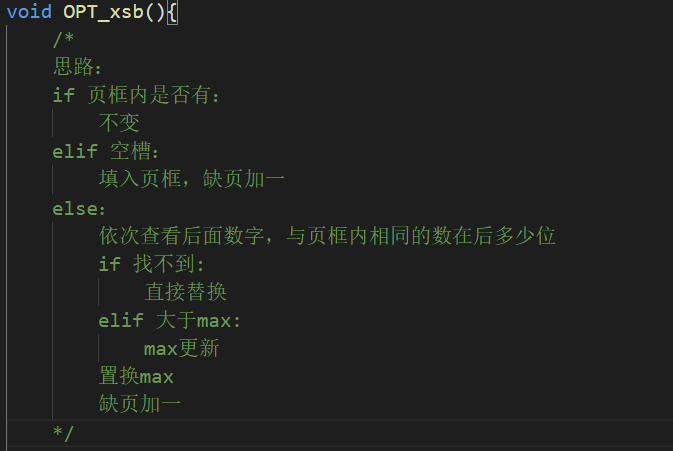




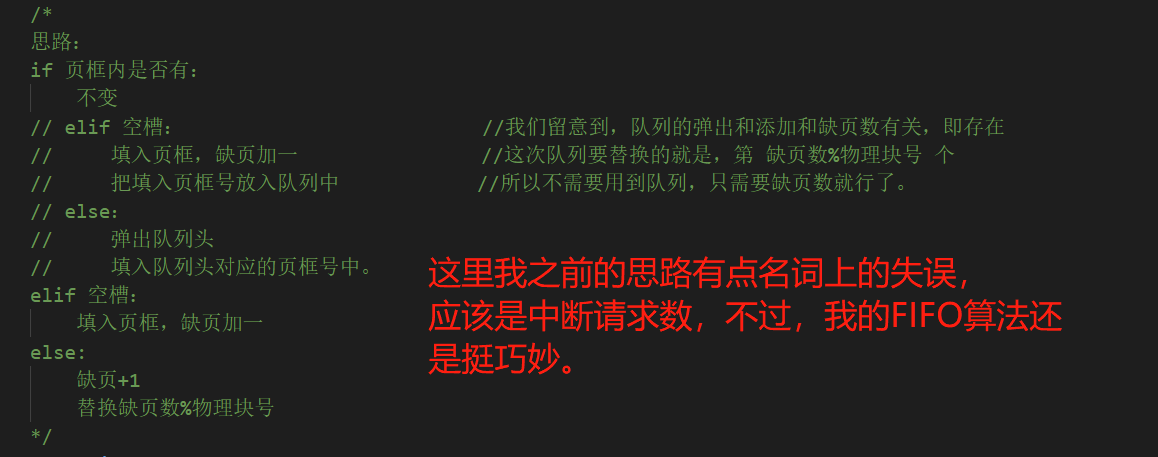


六、思考和分析

思考：

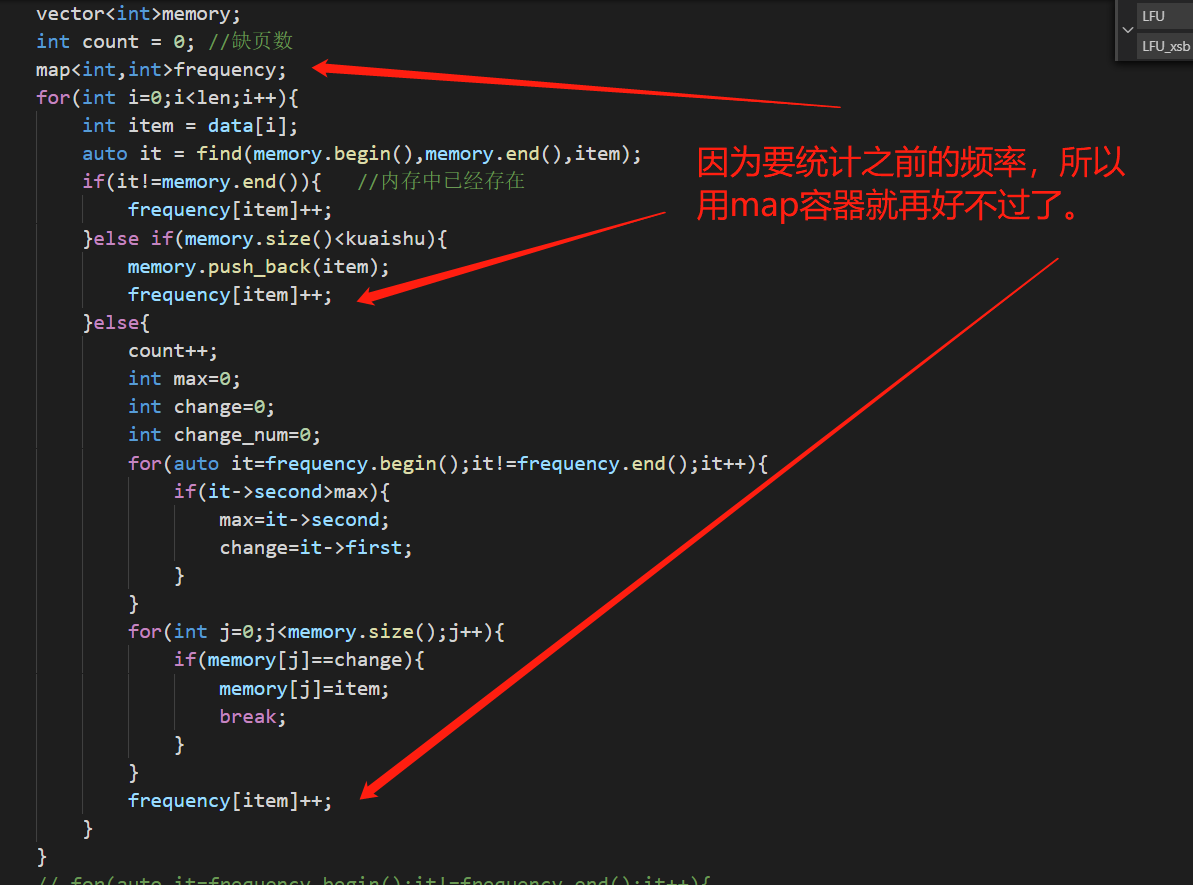


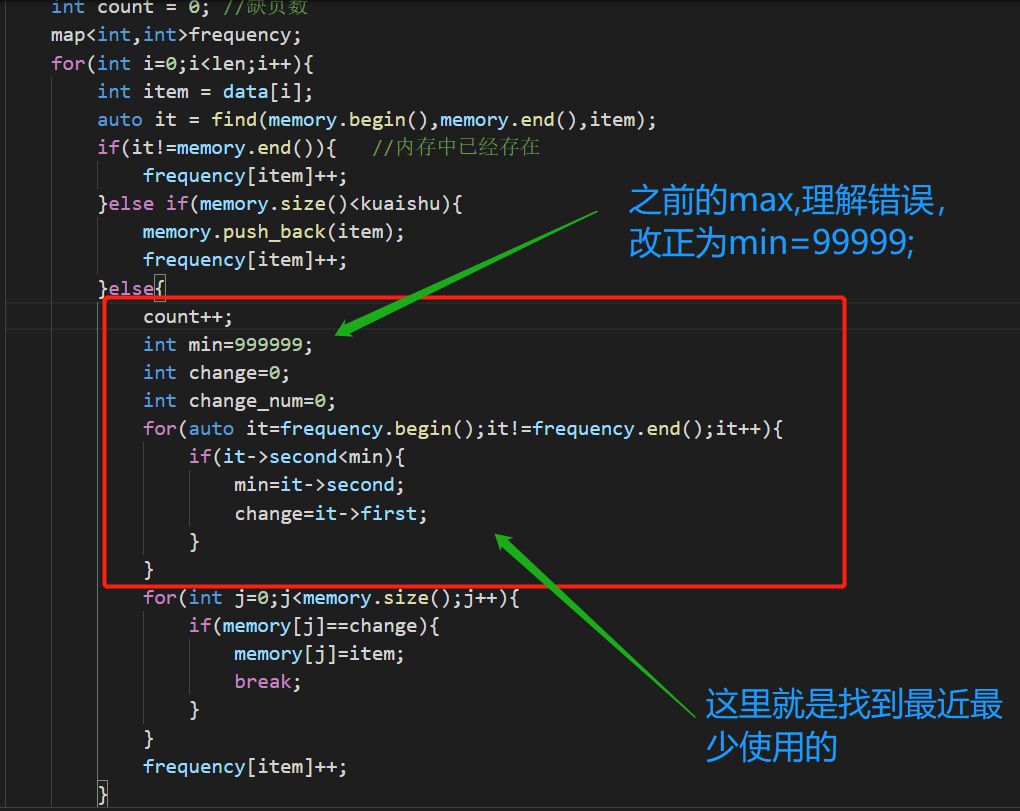
FIFO:

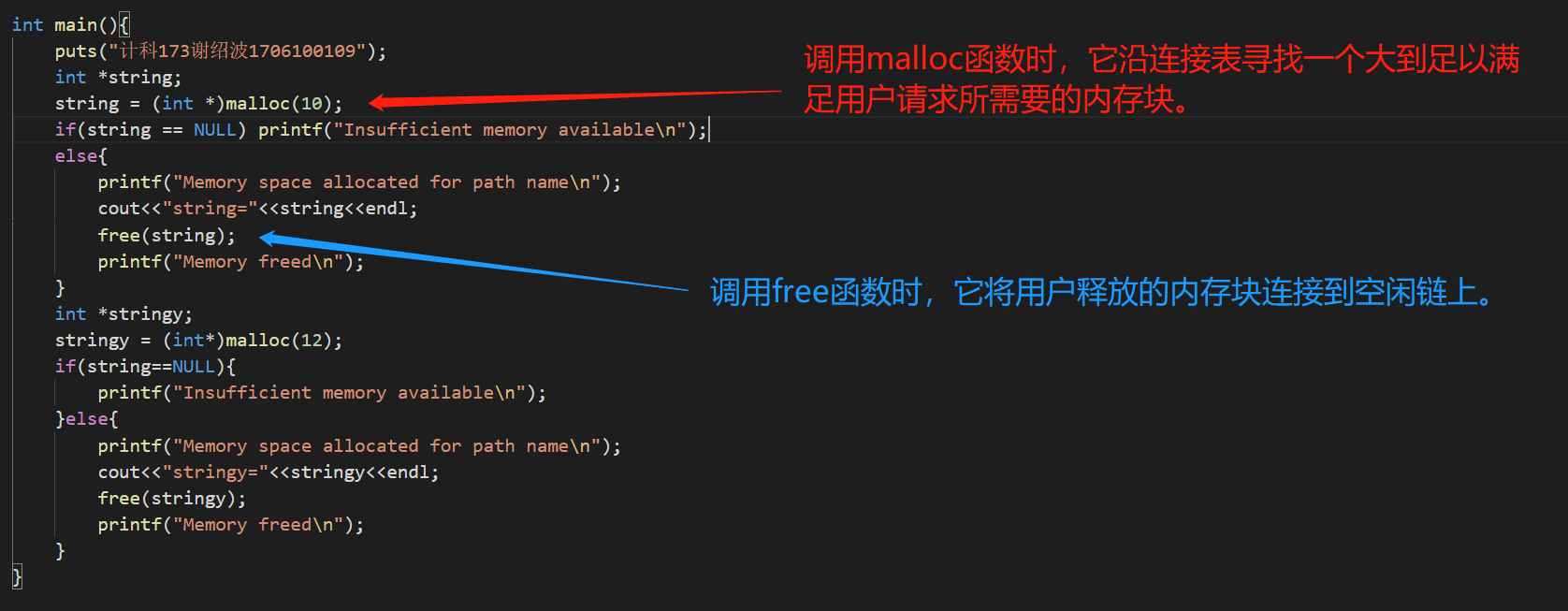


LRU：









实验心得：

通过本次实验，认真复习与理解了页面置换算法，并认真完成了四个算法，并求得其访问命中率。并且通过必要，了解了每个算法的特点，尤其是FIFO算法，通过对概念的理解，直接发现了中断请求数与页面置换的关系，用简单的数学公式模拟了队列。很是巧妙。

后面LRU，LFU算法，其实思路很像，为了方便，就直接用了set,和map，两个容器，对算法进行比较巧妙的实现。易读性大大提高。

同时，也理解了通过malloc和free函数，进行内存的分配和回收。

收获满满，受益匪浅。