操作系统实验四 页面置换算法

班级：2017211314　　学号：2017213508　　学生：蒋雪枫

一、任务：

9.21　写个程序来实现本章中介绍的 FIFO 和 LRU 页置换算法。首先，产生一个随机的页面引用序列，页面数从 0～9。将这个序列应用到每个算法并记录发生的页错误的次数。实现这个算法时，要将页帧的数量设为可变（从 1～7）。假设使用请求调页。

二、分析：

关于请求调页的页面置换算法，李老师已经在课堂上给我们讲了很多了。在完成本次实验的时候，学生直接想象上课时老师给出一个引用串与置换算法，我们做题的流程，所以本次实验还是完成地比较顺利的。由于 C++ STL 的 queue 并不支持遍历操作，这给我们的置换算法带来了些许不便，所以本次我们采用 C++ STL vector 来模拟 Frame。

三、变量说明：

int Frame_size; //页帧的数量，用户决定输入 0~7 作为其大小

int Reference_string[]；//引用串

int page_fault_times;//缺页次数

vector<int> q(Frame_size,-1);//页帧模拟，初始化为-1

四、源程序：

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    int Frame_size=5;
    cout<<"Firstly we simulate FIFO method,please input the initial Frames size:"<<endl;
    cin>>Frame_size;
    vector<int> q(Frame_size,-1);
    int Reference_string[]={ 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};
    int N=sizeof(Reference_string)/sizeof(Reference_string[0]);
    int page_fault_times=0;
    int victim=0;
    for(int i=0;i<N;i++)
    {
        int temp=Reference_string[i];
        bool flag=false;//means string[i] not in QUEUE
        for(int j=0;j<Frame_size;j++)
```

```
    {
        if(q[j]==temp)
        {
            flag=true;
            break;
        }
        if(q[j]==-1)
            break;
    }
    if(flag)
    {
        cout<<"TURN:"<<i<<" ,No page fault"<<endl;
    }
    else
    {
        q[victim]=temp;
        victim=(victim+1)%Frame_size;
        cout<<"TURN:"<<i<<" ,page fault! ";
        cout<<"FRMAMES:"<<q[0]<<" "<<q[1]<<" "<<q[2]<<" "<<q[3]<<" "<<q[4]<<endl
;
        page_fault_times++;

    }
}
cout<<"Sum of page fault is "<<page_fault_times<<endl;
cout<<"*********************************************************************"<<endl;
cout<<"Then we move to LRU method"<<endl;
//7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
vector<int> qq(Frame_size,-1);
victim=0;
page_fault_times=0;
N=sizeof(Reference_string)/sizeof(Reference_string[0]);
for(int i=0;i<N;i++)
{
    int temp=Reference_string[i];
    bool flag=false;//means string[i] not in Frames
    for(int j=0;j<Frame_size;j++)
    {
        if(qq[j]==temp)
        {
            flag=true;
            break;
        }
        if(qq[j]==-1)
```

```
            break;
        }
        if(flag)
        {
            cout<<"TURN:"<<i<<", "<<temp<<" ,No page fault"<<endl;
        }
        else
        {
            //stands for we need to refer to REFERENCE_STRING to find a victim in FRAMES
            page_fault_times++;
            int pointer=i;
            int counter=0;
            bool matcher[Frame_size];
            for(int ite=0;ite<Frame_size;ite++)
                matcher[ite]=false;
            while(pointer>=0&&counter<=Frame_size-1)
            {
                if(pointer<=0) break;
                pointer--;
                int comp=Reference_string[pointer];
                for(int it=0;it<Frame_size;it++)
                {
                    if(comp==qq[it])
                    {
                        matcher[it]=true;
                        counter++;
                        break;
                    }
                }
            }
//          cout<<"matcher:"<<matcher[0]<<matcher[1]<<matcher[2]<<matcher[3]<<matche
r[4]<<endl;
            for(int it=0;it<Frame_size;it++)
            {
                if(matcher[it]==false)
                {
                    qq[it]=Reference_string[i];
                    break;
                }
            }
            cout<<"TURN:"<<i<<", "<<temp<<" ,page fault! ";
            cout<<"FRMAMES:"<<qq[0]<<" "<<qq[1]<<" "<<qq[2]<<" "<<qq[3]<<" "<<qq[4]
<<endl;
```

```
    }
  }
  cout<<"Sum of page fault is "<<page_fault_times<<endl;
}
```

五、运行结果实例：

选择李老师上课讲 LRU 的例子作为 20bit 的引用串。

引用串：Reference_string[]={ 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};

```
C:\Users\Administrator\Desktop\OS实验4\PageReplace.exe
Firstly we simulate FIFO method,please input the initial Frames size:
5
TURN:0 ,page fault! FRMAMES:7 -1 -1 -1 -1
TURN:1 ,page fault! FRMAMES:7 0 -1 -1 -1
TURN:2 ,page fault! FRMAMES:7 0 1 -1 -1
TURN:3 ,page fault! FRMAMES:7 0 1 2 -1
TURN:4 ,No page fault
TURN:5 ,page fault! FRMAMES:7 0 1 2 3
TURN:6 ,No page fault
TURN:7 ,page fault! FRMAMES:4 0 1 2 3
TURN:8 ,No page fault
TURN:9 ,No page fault
TURN:10 ,No page fault
TURN:11 ,No page fault
TURN:12 ,No page fault
TURN:13 ,No page fault
TURN:14 ,No page fault
TURN:15 ,No page fault
TURN:16 ,No page fault
TURN:17 ,page fault! FRMAMES:4 7 1 2 3
TURN:18 ,page fault! FRMAMES:4 7 0 2 3
TURN:19 ,page fault! FRMAMES:4 7 0 1 3
Sum of page fault is 9
********************************************************************
Then we move to LRU method
TURN:0, 7 ,page fault! FRMAMES:7 -1 -1 -1 -1
TURN:1, 0 ,page fault! FRMAMES:7 0 -1 -1 -1
TURN:2, 1 ,page fault! FRMAMES:7 0 1 -1 -1
TURN:3, 2 ,page fault! FRMAMES:7 0 1 2 -1
TURN:4, 0 ,No page fault
TURN:5, 3 ,page fault! FRMAMES:7 0 1 2 3
TURN:6, 0 ,No page fault
TURN:7, 4 ,page fault! FRMAMES:4 0 1 2 3
TURN:8, 2 ,No page fault
TURN:9, 3 ,No page fault
TURN:10, 0 ,No page fault
TURN:11, 3 ,No page fault
TURN:12, 2 ,No page fault
TURN:13, 1 ,No page fault
TURN:14, 2 ,No page fault
TURN:15, 0 ,No page fault
TURN:16, 1 ,No page fault
TURN:17, 7 ,page fault! FRMAMES:7 0 1 2 3
TURN:18, 0 ,No page fault
TURN:19, 1 ,No page fault
Sum of page fault is 7
```

结果与实际一致。