参考代码三　分页存储管理

实验目的：　分页存储管理技术中的内存分配（模拟）

实验说明：
1）按照存储管理实验要求。
2）分配时，随机分配内存块。
3）文件包括两个：DefClass.h（所有数据结构的定义）和 Page_Memory.cpp（主程序和所有函数定义）。
4）没有单独定义 PCB 的数据结构，如就绪队列时，依然进的是 JCB 的数据结构（不太严格）。

实验结果：见程序运行结果（敲回车看下面的结果）。

```
//*************************************************************
//**********          参考代码三 分页存储管理          **************
//**********              头文件 DefClass.h            **************
//**********              版权所有: 陈礼青             ***************
//*************************************************************

const int MS=40;                                    //MS=1M
const int BS=4;                                     //BS=1K

////////////////////////////////////////////////////////////////////////////
//////////      后备作业队列表目              ////////////////
////////////////////////////////////////////////////////////////////////////
class Jcb
{
public:
    int id;                         //作业号
    int mem;                        //作业大小
    int ta;                         //作业到达时间
    int ts;                         //作业运行时间
    float priorty;                  //优先数
    int lefttime;
    Jcb(int aid,int amem,int aser_t,int aarr_t)
    {
        id=aid;
        mem=amem;
        ta=aarr_t;
        ts=aser_t;
        priorty=0;
        lefttime=aser_t;
    }
```

```
    Jcb()
    {
    }
};

////////////////////////////////////////////////////////////////////////////////
////////////////          主存分块表的表目          ////////////////
////////////////////////////////////////////////////////////////////////////////
class Rmbt
{
public:
                              //块号（缺省）
    bool status;              //状态
    int jid;                  //作业号
    int pid;                  //页号
    Rmbt()
    {
        status=false;
        jid=-1;
        pid=-1;
    }
};

////////////////////////////////////////////////////////////////////////////////
////////////////          主存分块表 MBT          ////////////////
////////////////////////////////////////////////////////////////////////////////
class Mbt
{
public:
    int len;
    int n;                    //n(当前空闲块数，初值为 MS/BS)
    Rmbt *mbt;                //表头指针
    Mbt()
    {
        len=n=MS/BS;
        mbt=new Rmbt[n];
    }
    ~Mbt()
    {
        delete mbt;
    }
};

////////////////////////////////////////////////////////////
```

```
class Pt;
////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
///////////////        作业表 JT 表目          /////////////////
////////////////////////////////////////////////////////////////////////////////
class Rjt
{
public:
    bool status;            //表目状态
    int jid;                //作业号
    int ptlen;              //页表长度
    Pt *page;               //页表始址（可用页表名表示）
    Rjt()
    {
        status=false;
        jid=-1;page=0;
        ptlen=0;
    }
};

////////////////////////////////////////////////////////////////////////////////
///////////////          页表 PT          ////////////////////
////////////////////////////////////////////////////////////////////////////////
class Pt
{
public:
    int k;                  //页表长度 K
    int *pt;                //页表实址
    Pt(int l):k(l)
    {
        pt=new int[k];
    }
    ~Pt()
    {
        delete pt;
    }
};
```

```
//**************************************************************************
//**********          参考代码三 分页存储管理          **********
//**********          主程序文件 Page_Memory.cpp       **********
//**********              版权所有: 陈礼青             **********
//**************************************************************************

#include <list>
#include <iostream>
#include <algorithm>
#include <stdlib.h>
#include <time.h>
#include "DefClass.h"
using namespace std ;

////////////////////////////////////////////////////////

const bool busy=true;
const bool leisure=false;
int Clock=0;                              //系统时钟
int id=-1;
bool wait=false;
bool CPU=false;
Mbt MBT;                                  //建立一主存分块表 MBT
list<Rjt> JT;                             //建立一个作业表 JT
list<Jcb> Queue;                          //估计队列
list<Jcb> mothball_q;                     //后备作业队列
list<Jcb> ready_q;                        //就绪队列
int cpusch[100];                          //CPU 调度表

////////////////////////////////////////////////////////

void Job_Sch(Jcb &);
void Realse_Mem(Jcb &);
bool New_Jcb_Com(list<Jcb> &,Jcb &);
void Print_Jcb(list<Jcb> &);
void Print_MBT();
void Print_JT();
void Print_PT(Pt*);
void Print_CPU_Sch();
void Print_Table();
```

```
//*************************************************************************
//**********                    主程序，FCFS                    **********
//*************************************************************************
void main()
{
    Jcb job,cur_p,newjcb;

    //系统将要运行的所有预期作业
    cout<<"**********系统将要运行的所有预期作业**********\n\n";

    ////////////////////////////////////////////////////
    //Jcb(作业号，作业大小 k，作业运行时间，作业到达时间)
    Queue.push_back(*new Jcb(1,12,4,0));
    Queue.push_back(*new Jcb(2,20,2,1));
    Queue.push_back(*new Jcb(3,10,2,2));
    Queue.push_back(*new Jcb(4,30,3,4));
    Queue.push_back(*new Jcb(5,10,1,5));
    Print_Jcb(Queue);
    ////////////////////////////////////////////////////

    //初始时刻：有若干个作业要求运行，调用建立后备作业队列模块

    //时刻二：假设某作业运行完毕，则调用释放内存模块，
    //若后备作业队列不空，则选择一个或几个作业，为其分配内存

    while(!Queue.empty()||!mothball_q.empty()||!ready_q.empty())
    {                                                    //系统运行时

        if(New_Jcb_Com(Queue,newjcb))                    //如果新作业来，入后备作业队列
            mothball_q.push_back(newjcb);

        while(!wait&&!mothball_q.empty())
        {                                                 //试分配
            job=mothball_q.front();
            Job_Sch(job);
            if(!wait)
            {
                ready_q.push_back(job);       //分配成功，入就绪队列
                cout<<"时刻"<<Clock<<"后瞬间:作业分配模块分配成功,作业";
                cout<<job.id<<"从外存调进主存，并创建 PCB 快，入就绪队列!\n";
            }
            else cout<<"时刻"<<Clock<<"后瞬间:作业分配模块分配失败,作业"<<job.id<<"
等待！\n";
        }
```

```
        if(CPU==busy)
        {                                          //如果有进程占有 CPU,继续运行
            if(cur_p.lefttime!=0)
            {
                cpusch[Clock]=id;
                cur_p.lefttime--;
            }
        }
        else
        {                                          //否则从就绪队列中选出一个，挑选最先
到的
            if(!ready_q.empty())
            {
                cur_p=ready_q.front();
                CPU=busy;
                id=cur_p.id;
                cout<<"时刻"<<Clock<<"后瞬间:进程"<<id<<"获得 CPU,出就绪队列!\n";
                cpusch[Clock]=id;
                ready_q.pop_front();          //出就绪队列
                cur_p.lefttime--;
            }
        }

        if(cur_p.lefttime==0&&CPU==busy)
        {
            CPU=leisure;
            Realse_Mem(cur_p);
            id=-1;
        }

        Clock++;
        Print_Table();

    }

    cout<<"******时刻"<<Clock<<"：所有的作业运行完毕******\n\n";
}

////////////////////////////////////////////////////////////////////////////////
//////////////////          用户的新作业到来              ////////////////
////////////////////////////////////////////////////////////////////////////////
bool New_Jcb_Com(list<Jcb> &Q,Jcb &newjcb)
{
```

```
    list<Jcb>::iterator iter;
    for(iter=Q.begin();iter!=Q.end();iter++)
        if((*iter).ta==Clock)
        {
            newjcb=(*iter);
            cout<<"时刻"<<Clock<<"作业"<<newjcb.id<<"到来!\n";
            Queue.erase(iter);
            return true;
        }
    return false;
}

////////////////////////////////////////////////////////////////////////////////
////////////////          释放内存模块          ////////////////
////////////////////////////////////////////////////////////////////////////////
void Realse_Mem(Jcb &p)
{
    for(int i=0;i<MBT.len;i++)
        if(MBT.mbt[i].jid==p.id)
        {
            MBT.mbt[i].status=false;
            MBT.mbt[i].jid=-1;
            MBT.mbt[i].pid=-1;
            MBT.n++;
        }
    delete JT.front().page;                    //删除页表
    JT.pop_front();                            //删除作业表项
    wait=false;                                //置阻塞变量为假
    cout<<"时刻"<<Clock+1<<":作业"<<p.id<<"释放内存!\n";
}

////////////////////////////////////////////////////////////////////////////
////////////////          向上取整          //////////////
////////////////////////////////////////////////////////////////////////////
int Up_Int(int a,int b)
{
    return a%b==0?a/b:a/b+1;
}

////////////////////////////////////////////////////////////////////////////
////////////////          作业调度模块          //////////////
////////////////////////////////////////////////////////////////////////////
void Job_Sch(Jcb &job)
{
```

```
    int k=Up_Int(job.mem,BS);                    //k=[X/L]
                                                 //查 MBT 表
    if(MBT.n<k)
        wait=true;                               //无法分配
    else
    {
        Rjt rjt;                                 //作业表填写
        rjt.status=true;
        rjt.jid=job.id;
        rjt.ptlen=k;
        rjt.page=new Pt(k) ;
        JT.push_back(rjt);

        srand( (unsigned)time( NULL ) );
        for(int p,i=0;i<k;)
        {
            p=rand()%MBT.len;                    //随机分配内存快
            if(MBT.mbt[p].status==false)
            {
                (*rjt.page).pt[i]=p;             //填写页表
                MBT.mbt[p].jid=job.id;           //填写 MBT 表
                MBT.mbt[p].pid=i;
                MBT.mbt[p].status=true;
                i++;
                MBT.n--;
            }
        }
        mothball_q.pop_front();                  //出后备作业队列
        cout<<"时刻"<<Clock<<"后瞬间的 MBT 表\n";
        Print_MBT();
        getchar();
    }
}

////////////////////////////////////////////////////////////////////////////////////////
//////////////////                  打印                  //////////////////////
////////////////////////////////////////////////////////////////////////////////////////
void Print_Table()
{
    list<Rjt>::iterator iter;
    if(id>=0)
        cout<<"\n 时刻"<<Clock<<":CPU 被进程"<<id<<"占有\n 各表状态：\n";
    else
        cout<<"\n 时刻"<<Clock<<":CPU 瞬间空闲\n 此时各表状态：\n";
```

```
    cout<<"********后备作业队列********\n";
    Print_Jcb(mothball_q);
    getchar();
    cout<<"********就绪队列********\n";
    Print_Jcb(ready_q);
    getchar();
    Print_CPU_Sch();
    getchar();
    Print_JT();
    getchar();
    for(iter=JT.begin();iter!=JT.end();iter++)
    {
        cout<<"进程"<<(*iter).jid<<"页表:\n";
        Print_PT((*iter).page);
        getchar();
    }
}

//////////////////////////////////////////////////////////////////

void Print_Jcb(list<Jcb> &p)
{
    list<Jcb>::iterator iter;
    cout<<"作业(进程)标示      作业大小\t 到达时间\t 服务时间\n";
    if(!p.empty())
    {
        for(iter=p.begin();iter!=p.end();iter++)
            cout<<(*iter).id<<"\t\t      "
                <<(*iter).mem<<"k\t\t"
                <<(*iter).ta<<"\t\t"
                <<(*iter).ts<<endl;
    }
}

//////////////////////////////////////////////////////////////////////////

void Print_MBT()
{
    cout<<"********MBT 表********\n 块号(缺省)\t 状态\t\t 作业号\t\t 页号\n";
    for(int i=0;i<MBT.len;i++)
        cout<<i<<"\t\t"
            <<MBT.mbt[i].status<<"\t\t"
            <<MBT.mbt[i].jid<<"\t\t"
```

```
                <<MBT.mbt[i].pid<<endl;
}

////////////////////////////////////////////////////////////////////////////

void Print_JT()
{
    list<Rjt>::iterator iter;
    cout<<"--------*作业表*--------\n 表目状态\t 作业号\t\t 页表长度\t 页表始址\n";
    for(iter=JT.begin();iter!=JT.end();iter++)
        cout<<(*iter).status<<"\t\t"
                <<(*iter).jid<<"k\t\t"
                <<(*iter).ptlen<<"\t\t"
                <<(*iter).page<<endl;
}

////////////////////////////////////////////////////////////////////////////

void Print_PT(Pt *page)
{
    cout<<"页号\t 块号\n";
    for(int i=0;i<(*page).k;i++)
        cout<<i<<"\t"<<(*page).pt[i]<<endl;
}

////////////////////////////////////////////////////////////////////////////

void Print_CPU_Sch()
{
    cout<<"CPU 调度表:(时刻 0->时刻"<<Clock<<")\n";
    for(int i=0;i<Clock;i++)
        cout<<" "<<cpusch[i];
}
```