参考代码一　调度算法

1.说明

本调度算法程序中，分别编制了四个作业调度算法，即：

void FCFS(JCB*);　　//先来先服务调度算法

void SJF(JCB*);　　//短作业优先调度算法

void RR(JCB*);　　//时间片轮转调度算法

void HRN(JCB*);　　//最高响应比优先调度算法

通过对同一个作业就绪队列的调度，从而得出短作业优先调度算法的平均周转时间最短这一结论。

2.数据结构

本程序用到了下列数据结构：

(1)作业控制块 JCB

```
class JCB
{
public:
    int Id;                     // Process's identity
    int Lefttime;               // The left time of the process to be served
    int Ts;                     // Time of service
    int Ta;                     // Time of arriving
    int Tw;                     // Time of waiting
    float priority;             // The priority of the process

    void SetJCB(int id,int ser_t,int arr_t)
    {
        Id=id;
        Ts=ser_t;
        Lefttime=ser_t;
        Ta=arr_t;
        Tw=0;
        priority=0;
    }

    friend bool operator < (const JCB &pre,const JCB &next)
    {
        if(pre.Ts<next.Ts) return true;
        return false;
    }
};
```

(2)运行结果记录 Record

```
class Record
{
public:
    void R_Dis(JCB *jcb)
    {
        float sum=0;
        for(int i=0;i<num;i++)
        {
            Tq[i]=FinishTime[i]-jcb[i].Ta;
            TqTs[i]=(float)(Tq[i])/jcb[i].Ts;
            sum+=TqTs[i];
        }
        float ave=sum/num;
        cout<<"进程标识:\t";
        for(i=0;i<num;i++) cout<<"P"<<jcb[i].Id<<"\t";
        cout<<endl;
        cout<<"到达时间:\t";
        for(i=0;i<num;i++) cout<<jcb[i].Ta<<"\t";
        cout<<endl;
        cout<<"服务时间:\t";
        for(i=0;i<num;i++) cout<<jcb[i].Ts<<"\t";
        cout<<endl;
        cout<<"周转时间:\t";
        for(i=0;i<num;i++) cout<<Tq[i]<<"\t";
        cout<<endl;
        cout<<"带权周转时间:\t";
        for(i=0;i<num;i++) cout<<TqTs[i]<<"\t";
        cout<<endl<<"平均带权周转时间:"<<ave<<endl;
    }
    int FinishTime[num];
    int Tq[num];
    float TqTs[num];
};
```

(3)全局变量

```
    const int num=5;    // Number of jobs/processes
    int Quantum=5;      // Time slice in Round Robin
    int N_Clock=0;      // The timer used to count the finish time
    class JCB;
```

3.程序及运行结果

(1)程序清单　　(C++语言编写，在 VC 6.0 环境下运行)

```
//******************************************************************************
//*******************              参考代码一  调度算法            *******************
//*******************        主程序文件 Schedule_Algorithms.cpp      *******************
//*******************              版权所有：陈礼青                 *******************
//******************************************************************************

#include <list>
#include <iostream>
#include <algorithm>
using namespace std ;

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

const int num=5;     / Number of jobs/processes
int Quantum=5;     // Time slice in Round Robin
int N_Clock=0;     // The timer used to count the finish time

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

class JCB
{
public:
    int Id;                          // Process's identity
    int Lefttime;                    // The left time of the process to be served
    int Ts;                          // Time of service
    int Ta;                          // Time of arriving
    int Tw;                          // Time of waiting
    float priority;                  // The priority of the process
    void SetJCB(int id,int ser_t,int arr_t)
    {
        Id=id;
        Ts=ser_t;
        Lefttime=ser_t;
        Ta=arr_t;
```

```
        Tw=0;
        priority=0;
    }

    friend bool operator < (const JCB &pre,const JCB &next)
    {
        if(pre.Ts<next.Ts) return true;
        return false;
    }
};

class Record
{
public:
    void R_Dis(JCB *jcb)
    {
        float sum=0;
        for(int i=0;i<num;i++)
        {
            Tq[i]=FinishTime[i]-jcb[i].Ta;
            TqTs[i]=(float)(Tq[i])/jcb[i].Ts;
            sum+=TqTs[i];
        }
        float ave=sum/num;
        cout<<"进程标识:\t";
        for(i=0;i<num;i++) cout<<"P"<<jcb[i].Id<<"\t";
        cout<<endl;
        cout<<"到达时间:\t";
        for(i=0;i<num;i++) cout<<jcb[i].Ta<<"\t";
        cout<<endl;
        cout<<"服务时间:\t";
        for(i=0;i<num;i++) cout<<jcb[i].Ts<<"\t";
        cout<<endl;
        cout<<"周转时间:\t";
        for(i=0;i<num;i++) cout<<Tq[i]<<"\t";
        cout<<endl;
```

```
        cout<<"带权周转时间:\t";
        for(i=0;i<num;i++) cout<<TqTs[i]<<"\t";
        cout<<endl<<"平均带权周转时间:"<<ave<<endl;
    }
    int FinishTime[num];
    int Tq[num];
    float TqTs[num];
};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
bool NewCome(JCB *jcb,JCB &n_p)
{
    for(int i=1;i<num;i++)
        if(jcb[i].Ta==N_Clock)
        {
            n_p=jcb[i];
            return true;
        }
    return false;
}

void Init_BQ(JCB *jcb)
{
    jcb[0].SetJCB(0,3,0);
    jcb[1].SetJCB(1,6,2);
    jcb[2].SetJCB(2,4,4);
    jcb[3].SetJCB(3,5,6);
    jcb[4].SetJCB(4,2,8);
}

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

void FCFS(JCB*);        //First-Come-First-Served Scheduling Algorithm
void SJF(JCB*);         //Shortest-Job-First Scheduling Algorithm
void RR(JCB*);          //Round-Robin Scheduling Algorithm
void HRN(JCB*);         //Highest-Response Ratio-Next Scheduling Algorithm
```

```
//*************************************************************
//**********************        以下为主函数        **************
//*************************************************************
void main()
{
    JCB jcb[num];
    Init_BQ(jcb);
    FCFS(jcb);
    cout<<"Press any key to go on...";getchar();
    Init_BQ(jcb);
    SJF(jcb);
    cout<<"Press any key to go on...";getchar();;
    Init_BQ(jcb);
    RR(jcb);
    cout<<"Press any key to go on...";getchar();
    Init_BQ(jcb);
    HRN(jcb);
    cout<<"\n 由以上数据可知，短作业优先调度算法的平均周转时间最短!\n\n";
}

//*************************************************************************
//**************************        以下分别为四个调度算法        ************
//*************************************************************************
void FCFS(JCB *jcb)
{
    cout<<"\n1.*****先来先服务调度算法*****\n";
    list<JCB> Q;
    list<JCB>::iterator iter;
    N_Clock=0;
    JCB cur_p,new_p;
    Record r_fcfs;
    Q.push_front(jcb[0]);
    while(!Q.empty())
    {
        cur_p=Q.front();
        while(cur_p.Lefttime!=0)
```

```
        {
            N_Clock++;
            cur_p.Lefttime--;
            if(NewCome(jcb,new_p)) Q.push_back(new_p);
        }
        r_fcfs.FinishTime[cur_p.Id]=N_Clock;
        Q.pop_front();
    }
    r_fcfs.R_Dis(jcb);
}

void SJF(JCB *jcb)
{
    cout<<"\n2.*****短作业优先调度算法*****\n";
    list<JCB> Q;
    list<JCB>::iterator iter;
    N_Clock=0;
    JCB cur_p,new_p;
    Record r_SJF;
    Q.push_front(jcb[0]);
    while(!Q.empty())
    {
        cur_p=Q.front();
        while(cur_p.Lefttime!=0)
        {
            N_Clock++;
            cur_p.Lefttime--;
            if(NewCome(jcb,new_p))
                Q.push_back(new_p);
        }
        r_SJF.FinishTime[cur_p.Id]=N_Clock;
        Q.pop_front();
        Q.sort();
    }
    r_SJF.R_Dis(jcb);
}
```

```
void RR(JCB *jcb)
{
    cout<<"\n3.*****时间片轮转调度算法(时间片大小="<<Quantum<<")*****\n";
    list<JCB> Q;
    list<JCB>::iterator iter;
    N_Clock=0;
    JCB cur_p,new_p;
    Record r_rr;
    int temp_clock;
    Q.push_front(jcb[0]);
    while(!Q.empty())
    {
        cur_p=Q.front();
        if(cur_p.Lefttime<=Quantum)
        {
            while(cur_p.Lefttime!=0)
            {
                N_Clock++;
                cur_p.Lefttime--;
                if(NewCome(jcb,new_p)) Q.push_back(new_p);
            }//while
            r_rr.FinishTime[cur_p.Id]=N_Clock;
            Q.pop_front();
        }//if
        else
        {
            temp_clock=N_Clock;
            while(N_Clock+1<=temp_clock+Quantum)
            {
                N_Clock++;
                cur_p.Lefttime--;
                if(NewCome(jcb,new_p)) Q.push_back(new_p);
            }//while
            Q.pop_front();
            Q.push_back(cur_p);
```

```
            }//else
        }//while
        r_rr.R_Dis(jcb);
}

void rp(JCB &jcb)
{
        jcb.Tw++;
        jcb.priority=(jcb.Tw+jcb.Ts)/jcb.Ts;
}

bool pr(const JCB &pre,const JCB &next)
{
            if(pre.priority<next.priority) return true;
            return false;
}

void HRN(JCB *jcb)
{                                   // Ts (3,1)<Ts (2,2)
        cout<<"\n4.*****最高响应比优先调度算法(非抢占方式)*****\n";
        list<JCB> Q;
        list<JCB>::iterator iter;
        N_Clock=0;
        JCB cur_p,new_p;
        Record r_hrp;
        Q.push_front(jcb[0]);
        while(!Q.empty())
        {
            iter=max_element(Q.begin(),Q.end(),pr);
            cur_p=(*iter);
            Q.erase(iter);
            while(cur_p.Lefttime!=0)
            {
                N_Clock++;
                cur_p.Lefttime--;
                if(NewCome(jcb,new_p)) Q.push_back(new_p);
```

```
                for_each(Q.begin(),Q.end(),rp);
            }
            r_hrp.FinishTime[cur_p.Id]=N_Clock;
        }
        r_hrp.R_Dis(jcb);
}
```

(2)运行结果

在本程序运行过程中，首先打印先来先服务调度算法的运行结果，按下任何键，则打印下一个调度算法的运行结果，最后可知，短作业优先调度算法的平均周转时间最短。具体的运行结果略。