四 川 大 学 计 算 机 学 院、软 件 学 院

实 验 报 告

学号：2022141460180 姓名：封欢欢 专业：计算机科学与技术 班级：行政 4 班 第 8-9 周

课程名称	操作系统课程设计	实验课时	4 小时
实验项目	作业调度	实验时间	第 8 周到第 9 周
实验目的	1）掌握周转时间、等待时间、平均周转时间等概念及其计算方法； 2）理解五种常用的作业调度算法（FCFS, SJF, HRRF, HPF, RR），区分算法之间的差异性，并用 c 语言模拟实现各算法； 3）了解操作系统中高级调度、中级调度和低级调度的区别和联系；		
实验环境	HUAWEI 电脑 WSL2 Ubuntu2204		
实验内容（算法、程序、步骤和方法）	本次实验是用 c 语言模拟作业调度算法，我选择了其中的三种：SJF、HRRF、HPF。 读取文件 data.txt 的内容如下： 作业ID 到达时间 执行时间 优先权 1 800 50 0 2 815 30 1 3 830 25 2 4 835 20 2 5 845 15 2 6 700 10 1 7 820 5 0 作业信息结构：		

[*] debug.cpp

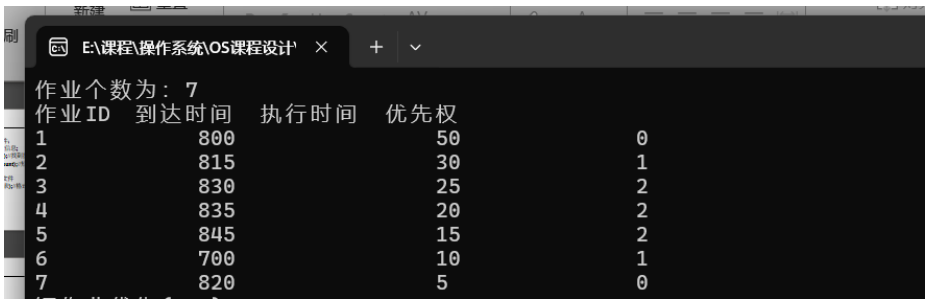
```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <algorithm>>
using namespace std;
int WorkNumber;//统计作业数

typedef struct node {
    int number;          //作业号
    int reach_time;      //作业抵达时间
    int need_time;       //作业的执行时间
    int privilege;     //作业优先权
    float excellent;     //响应比
    int wait_time;       //等待时间
    int tr_time;         //周转时间
    double wtr_time;    //带权周转时间
    int visited;          //作业是否完成
} job;
```

main 函数以及完整输出截图：

```cpp
int main() {

    read_Jobdata("data.txt");//读数据文件
    job jobs[WorkNumber];
    initial_jobs(jobs,"data.txt");//初始化作业信息
    cout<<"短作业优先(SJF):"<<endl;
    SJF(jobs,0);
    averagedata(jobs);//输出平均等待时间，平均周转时间，平均带权周转时间

    initial_jobs(jobs,"data.txt");
    cout<<"高相应比优先(HRRF):"<<endl;
    HRRF(jobs,0);
    averagedata(jobs);

    initial_jobs(jobs,"data.txt");
    cout<<"优先权高者优先(HPF):"<<endl;
    HPF(jobs,0);
    averagedata(jobs);
    return 0;
}
```

```
E:\课程\操作系统\OS课程设计\
作业个数为: 7
作业ID   到达时间   执行时间   优先权
1           800              50             0
2           815              30             1
3           830              25             2
4           835              20             2
5           845              15             2
6           700              10             1
7           820              5              0
短作业优先(SJF):
执行完的作业是: 6号作业, 等待时间为0 周转时间为10 带权周转时间为1.00
执行完的作业是: 1号作业, 等待时间为0 周转时间为50 带权周转时间为1.00
执行完的作业是: 7号作业, 等待时间为30 周转时间为35 带权周转时间为7.00
执行完的作业是: 5号作业, 等待时间为10 周转时间为25 带权周转时间为1.67
执行完的作业是: 4号作业, 等待时间为35 周转时间为55 带权周转时间为2.75
执行完的作业是: 3号作业, 等待时间为60 周转时间为85 带权周转时间为3.40
执行完的作业是: 2号作业, 等待时间为100 周转时间为130 带权周转时间为4.33
平均等待时间: 33.57
平均周转时间:55.71
平均带权周转时间:3.02
高相应比优先(HRRF):
执行完的作业是: 6号作业, 等待时间为0 周转时间为10 带权周转时间为1.00
执行完的作业是: 1号作业, 等待时间为0 周转时间为50 带权周转时间为1.00
执行完的作业是: 7号作业, 等待时间为30 周转时间为35 带权周转时间为7.00
执行完的作业是: 2号作业, 等待时间为40 周转时间为70 带权周转时间为2.33
执行完的作业是: 5号作业, 等待时间为40 周转时间为55 带权周转时间为3.67
执行完的作业是: 4号作业, 等待时间为65 周转时间为85 带权周转时间为4.25
执行完的作业是: 3号作业, 等待时间为90 周转时间为115 带权周转时间为4.60
平均等待时间: 37.86
平均周转时间:60.00
平均带权周转时间:3.41
优先权高者优先(HPF):
执行完的作业是: 6号作业, 等待时间为0 周转时间为10 带权周转时间为1.00
执行完的作业是: 1号作业, 等待时间为0 周转时间为50 带权周转时间为1.00
执行完的作业是: 7号作业, 等待时间为30 周转时间为35 带权周转时间为7.00
执行完的作业是: 2号作业, 等待时间为40 周转时间为70 带权周转时间为2.33
执行完的作业是: 3号作业, 等待时间为55 周转时间为80 带权周转时间为3.20
执行完的作业是: 4号作业, 等待时间为75 周转时间为95 带权周转时间为4.75
执行完的作业是: 5号作业, 等待时间为85 周转时间为100 带权周转时间为6.67
平均等待时间: 40.71
平均周转时间:62.86
```

读入文件显示：



除调度算法函数外的其他函数

作业相关信息输出：

```
void per_output(job a) //输出每一个完成的作业的相关信息
{
    cout<<"执行完的作业是："<<a.number<<"号作业，等待时间为"<<a.wait_time<<" 周转时间为"<<a.tr_time<<" 带权周转时间为";
    cout<<fixed<<setprecision(2);//保留两位小数
    cout<<a.wtr_time<<endl;
}
```

重载 sort 比较

```
bool cmp(job a,job b) { //sort比较
    return a.reach_time<=b.reach_time;
}
```

读取文件函数：

```
void read_Jobdata(string file) { //读取数据文件;
    ifstream ifs;
    ifs.open(file,ios::in);
    if (!ifs.is_open()) {                    //判断文件是否打开
        cout << "打开文件失败！！！";
        return ;
    }   //打开文件
    char c;
    int lineCnt=0;//统计行数
    while(ifs.get(c)) {
        if(c=='\n')
            lineCnt++;
    }
    WorkNumber=lineCnt;
    cout<<"作业个数为："<<lineCnt<<endl;
    ifs.close();

    ifs.open(file,ios::in);
    if (!ifs.is_open()) {                    //判断文件是否打开
        cout << "打开文件失败！！！";
        return ;
    }   //打开文件
    string data;
    while( getline(ifs,data) )
        cout<<data<<endl;//按行查看文件信息
    ifs.close();
}
```

初始化所有作业信息函数：

```cpp
void initial_jobs(job jobs[],string file) { //初始化所有作业信息;
    ifstream ifs;
    ifs.open(file,ios::in);
    if (!ifs.is_open()) {                    //判断文件是否打开
        cout << "打开文件失败！！！";
        return ;
    }
    //输入到对应位置
    int u, i=0,j=0;
    string data;
    getline(ifs,data);//先把标签行处理了
    while(ifs>>u) {
        if(j==0)    jobs[i].number=u;
        if(j==1)    jobs[i].reach_time=u;
        if(j==2)    jobs[i].need_time=u;
        if(j==3)    jobs[i].privilege=u;
        j++;
        if(j>3)     i++,j=0;
    }
    ifs.close();
    for(int i = 0; i < WorkNumber; i++) {//其他的初始化为0
        jobs[i].tr_time=0;
        jobs[i].wait_time=0;
        jobs[i].excellent=0;
        jobs[i].wtr_time=0;
        jobs[i].visited=0;
    }
    sort(jobs,jobs+7,cmp);//根据到达时间对作业进行排序,
}
```

每个算法的平均时间输出函数：

```cpp
void averagedata(job v[]) //输出平均等待时间，平均周转时间，平均带权周转时间
{
    double wait=0,tr=0,wtr=0;
    for(int i = 0; i < WorkNumber; i++) {
        wait+=v[i].wait_time;
        tr+=v[i].tr_time;
        wtr+=v[i].wtr_time;
    }
    cout<<"平均等待时间："<<wait/(double)WorkNumber<<endl;
    cout<<"平均周转时间:"<<tr/(double)WorkNumber<<endl;
    cout<<"平均带权周转时间:"<<wtr/(double)WorkNumber<<endl;
}
```

算法一：短作业优先（SJF）

算法基本思想：根据作业控制块中作业申请时指出的执行时间，选取执行时间最短的作业优先调度。

代码：

```cpp
void SJF( job v[],int count) { //短作业优先算法，count表示目前时刻
    int st=1e8;//用于后面和作业运行时间比较，找出最小
    int j=-1;//记录取哪个 并且作为退出递归标志
    for(int i=0 ; i < WorkNumber; i++)//在已经抵达里的作业找最短作业
        if(count>=v[i].reach_time&&v[i].visited==0)
            if(st>v[i].need_time) {
                st=v[i].need_time;
                j=i;//最终j即为最短作业的数组索引
            }
    if(j==-1)//说明还没有作业到达，得新找count
        for(int i=0 ; i < WorkNumber; i++)
            if(v[i].visited==0) {
                count=v[i].reach_time;
                j=i;
                break;//前面以及排过序，所以找到的第一个即使第一个到达的
            }

    if(j==-1)   return;       //说明所有的作业都已经访问了 ,退出递归

    v[j].wait_time=count-v[j].reach_time;
    v[j].tr_time=v[j].need_time+v[j].wait_time;//周转时间
    v[j].wtr_time=(double)v[j].tr_time/(double)v[j].need_time;//带权周转时间
    per_output(v[j]);//输出函数
    v[j].visited=1;//作业已经完成

    SJF(v,count+v[j].need_time);//刷新时刻，递归调用SJF
}
```

输出：

```
短作业优先(SJF):
执行完的作业是: 6号作业, 等待时间为0 周转时间为10 带权周转时间为1.00
执行完的作业是: 1号作业, 等待时间为0 周转时间为50 带权周转时间为1.00
执行完的作业是: 7号作业, 等待时间为30 周转时间为35 带权周转时间为7.00
执行完的作业是: 5号作业, 等待时间为10 周转时间为25 带权周转时间为1.67
执行完的作业是: 4号作业, 等待时间为35 周转时间为55 带权周转时间为2.75
执行完的作业是: 3号作业, 等待时间为60 周转时间为85 带权周转时间为3.40
执行完的作业是: 2号作业, 等待时间为100 周转时间为130 带权周转时间为4.33
平均等待时间: 33.57
平均周转时间:55.71
平均带权周转时间:3.02
```

算法二：高响应比优先（HRRF）

算法基本思想：每次调度前都要进行响应比计算，响应比高者优先

$$响应比 = \frac{响应时间}{处理时间} = \frac{等待时间 + 处理时间}{处理时间} = 1 + \frac{等待时间}{处理时间}$$

代码：

```
void HRRF(job v[],int count) { //高响应比优先算法
    double st=0;//用于后面和响应比比较，找出最大
    int j=-1;//记录取哪个 并且作为退出递归标志
    for(int i=0 ; i < WorkNumber; i++)//在已经抵达里的作业找响应比最高的作业
        if(count>=v[i].reach_time&&v[i].visited==0) {
            v[i].wait_time=count-v[i].reach_time;
            v[i].excellent=1+(float)v[i].wait_time/(float)v[i].need_time;
            if(st<v[i].excellent) {
                st=v[i].excellent;
                j=i;//最终j即为最短作业的数组索引
            }
        }
    if(j==-1)//说明还没有作业到达，得新找count
        for(int i=0 ; i < WorkNumber; i++)
            if(v[i].visited==0) {
                count=v[i].reach_time;
                j=i;
                break;//前面以及排过序，所以找到的第一个即使第一个到达的
            }
    if(j==-1)    return;       //说明所有的作业都已经访问了，退出递归
    v[j].wait_time=count-v[j].reach_time;
    v[j].tr_time=v[j].need_time+v[j].wait_time;//周转时间
    v[j].wtr_time=(double)v[j].tr_time/(double)v[j].need_time;//带权周转时间
    per_output(v[j]);//输出函数
    v[j].visited=1;//作业已经完成

    HRRF(v,count+v[j].need_time);//刷新时刻，递归调用HRRF
}
```

输出：

```
高相应比优先(HRRF):
执行完的作业是: 6号作业, 等待时间为0 周转时间为10 带权周转时间为1.00
执行完的作业是: 1号作业, 等待时间为0 周转时间为50 带权周转时间为1.00
执行完的作业是: 7号作业, 等待时间为30 周转时间为35 带权周转时间为7.00
执行完的作业是: 2号作业, 等待时间为40 周转时间为70 带权周转时间为2.33
执行完的作业是: 5号作业, 等待时间为40 周转时间为55 带权周转时间为3.67
执行完的作业是: 4号作业, 等待时间为65 周转时间为85 带权周转时间为4.25
执行完的作业是: 3号作业, 等待时间为90 周转时间为115 带权周转时间为4.60
平均等待时间: 37.86
平均周转时间:60.00
平均带权周转时间:3.41
```

算法三:优先权高者优先(HPF)

算法基本思想：系统根据作业的优先权进行作业调度，每次选取优先权高的作业优先调度。作业的优先权通常用一个整数表示，也叫做优先数。优先数的大小与优先权的关系由系统或者用户来规定，本实验采用优先权值越小，优先权越高。

<table>
<tr><td>（接上）
实验内容（算法、程序、步骤和方法）</td><td>

代码：

```
void HPF(job v[],int count) {//优先权高者优先算法
    int st=1e8;//用于后面和作业的优先权值比较，找出最小
    int j=-1;//记录取哪个 并且作为退出递归标志
    for(int i=0 ; i < WorkNumber; i++)//在已经抵达里的作业找优先权最大(即优先权值最小)的作业
        if(count>=v[i].reach_time&&v[i].visited==0)
            if(st>v[i].privilege)
            {
                st=v[i].privilege;
                j=i;//最终j即为最短作业的数组索引
            }
    if(j==-1)//说明还没有作业到达，得新找count
        for(int i=0 ; i < WorkNumber; i++)
            if(v[i].visited==0) {
                count=v[i].reach_time;
                j=i;
                break;//前面以及排过序，所以找到的第一个即使第一个到达的
            }

    if(j==-1)    return;        //说明所有的作业都已经访问了 ,退出递归

    v[j].wait_time=count-v[j].reach_time;
    v[j].tr_time=v[j].need_time+v[j].wait_time;//周转时间
    v[j].wtr_time=(double)v[j].tr_time/(double)v[j].need_time;//带权周转时间
    per_output(v[j]);//输出函数
    v[j].visited=1;//作业已经完成

    HPF(v,count+v[j].need_time);//刷新时刻，递归调用HPF
}
```

输出：

```
优先权高者优先(HPF):
执行完的作业是： 6号作业，等待时间为0 周转时间为10 带权周转时间为1.00
执行完的作业是： 1号作业，等待时间为0 周转时间为50 带权周转时间为1.00
执行完的作业是： 7号作业，等待时间为30 周转时间为35 带权周转时间为7.00
执行完的作业是： 2号作业，等待时间为40 周转时间为70 带权周转时间为2.33
执行完的作业是： 3号作业，等待时间为55 周转时间为80 带权周转时间为3.20
执行完的作业是： 4号作业，等待时间为75 周转时间为95 带权周转时间为4.75
执行完的作业是： 5号作业，等待时间为85 周转时间为100 带权周转时间为6.67
平均等待时间： 40.71
平均周转时间:62.86
平均带权周转时间:3.71
```

</td></tr>
<tr><td>数据记录
和计算</td><td>计算：
响应比=1+等待时间/处理时间
等待时间=开始运行时间-作业到达时间
周转时间=等待时间+运行时间
带权周转时间=周转时间/运行时间</td></tr>
<tr><td>结　论
（结　果）</td><td>SJF:短作业优先调度算法选取执行时间最短的作业优先调度，只考虑了作业的运行时间而忽略了作业的等待时间，对于长作业很不友好。
HRRF:高响应比优先调度算法为 FCFS 和 SJF 的折中，使长作业不会长时间等待，但是每次调度前都要进行响应比计算。
HPF:优先权高者优先调度算法根据作业的优先权进行作业调度，综合考虑了作业执行时间和等待时间的长短、作业的缓急度、作业对外部设备的使用情况等因素。</td></tr>
<tr><td>小　结</td><td>通过此次实验，我掌握了等待时间，周转时间，带权周转时间的概念和计算，深入理解了在不同作业调度算法下，一个作业的调度过程。对调度算法之间的区别和优劣也有了进一步的认识。
还需要不断学习总结加深理解，以便熟练运算，运用自如。</td></tr>
</table>

指导老师评议	成绩评定：　　指导教师签名：