浙江工业大学

操作系统课程设计项目二——
进程调度 算法实现

2019/2020(2)

项目参与成员 赵水

任课教师 吴福理

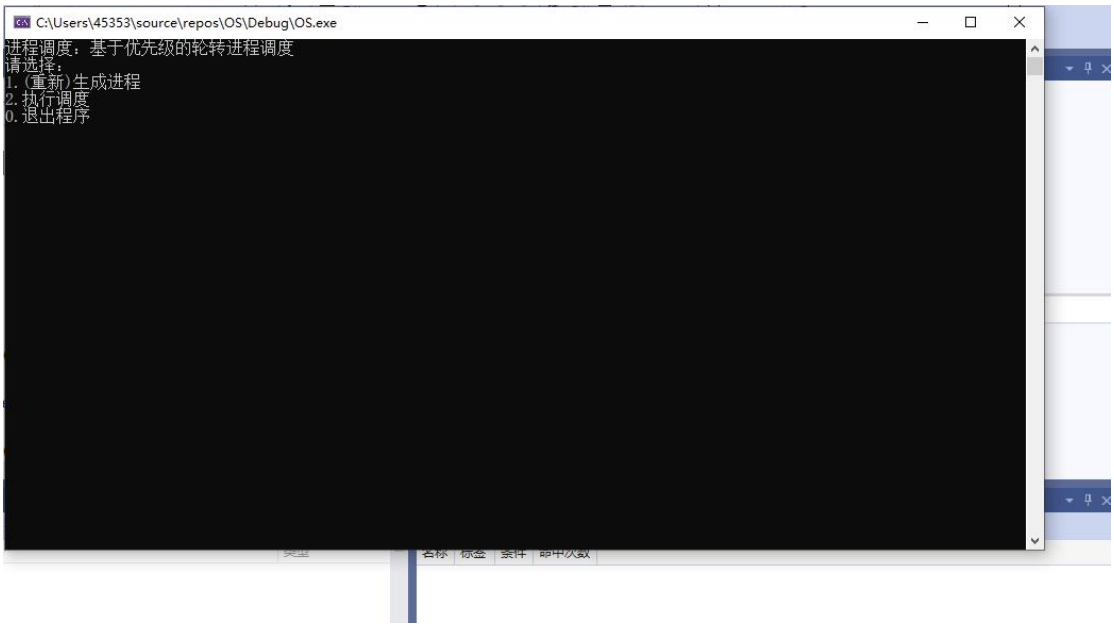
目录

一、项目内容与目标

进程调度 算法实现
1、 实现基于优先级的轮转进程调度
2、 至少创建 10 个进程并输出它们在上述调度算法下的调度情况，并输出到终端，以检查算法 的执行情况

二、设计思路和实现

设计思路 1——总体目标：
以控制台程序为整体目标，实现能够与用户简单交互的程序。
菜单功能：所有的功能豆浆封装成函数的形式，用户可以根据菜单提示的内容自由输入选择项，系统将把其作为作为参数调用相应模块的函数，选择程序功能或者退出程序。如下图所示：
函数申明 int menu（）；//返回为用户输入的选项值。



实现：

```
int menu()
{
    cout << "进程调度：基于优先级的轮转进程调度" << endl << "请选择：\n1.(重新)生成进程\n2.执行调度\n0.退出程序\n";
    int choice;
    cin >> choice;
    return choice;
}
```

设计思路 2——进程的表示：我创建了一个数据结构 Task 用来储存进程，其拥有五个字段：

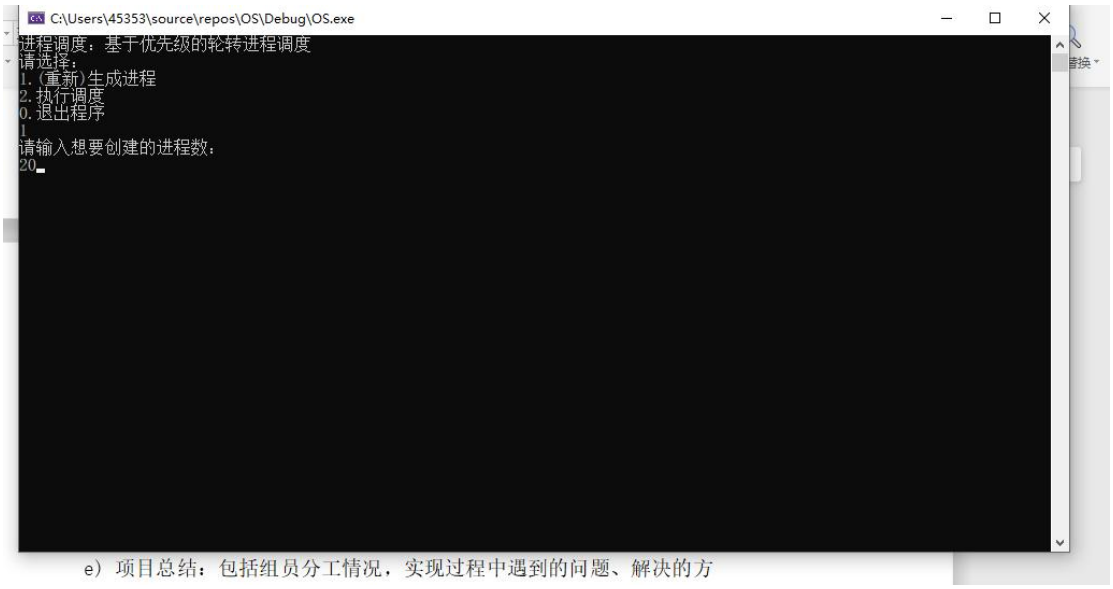
1. id ——用来表示进程号，区分进程；
2. needtime——用来记录执行完进程还需要的 CPU 时间
3. wholetime——用来记录执行完进程总共所需的 CPU 时间
4. state——用来表示进程状态，0 表示进程还未执行完，1 表示进程执行完毕。
5. priority——用来表示进程的优先级

我使用这样的数组来储存若干个进程

实现：

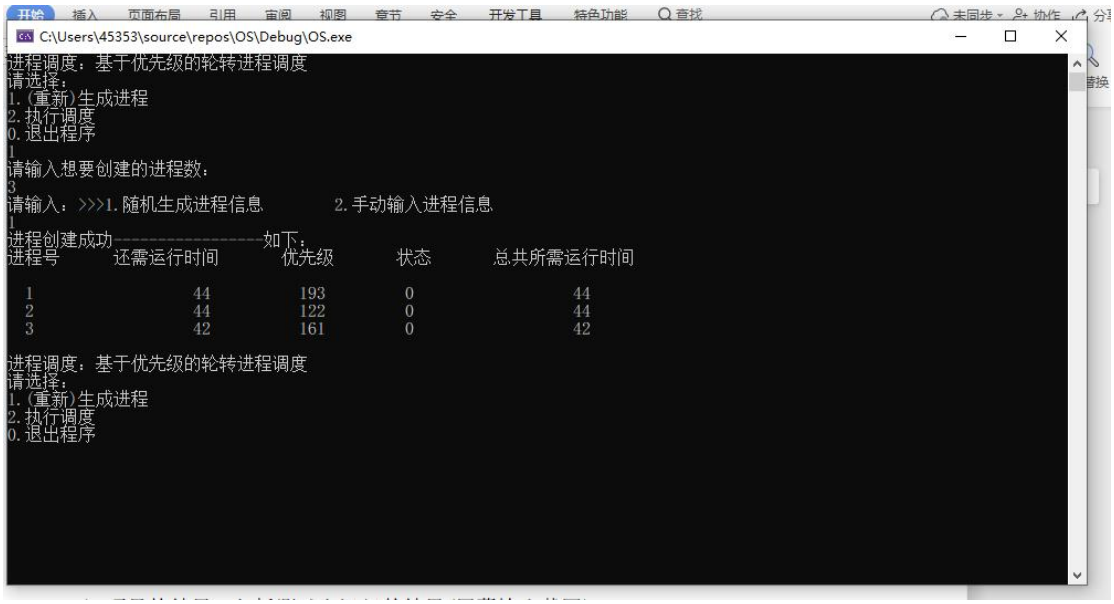
```
struct Task {
    int id;//进程号
    int needtime;//所需 CPU 时间
    int wholetime;//总共所需时间
    int state;//状态字，1 为执行完毕，0 为待执行
    int priority;//优先级
};
Task * tasks;//定义进程数组指针，以备动态创建数组
```

设计思路 3——进程的生成：进程根据用户输入的进程数生成，如图：

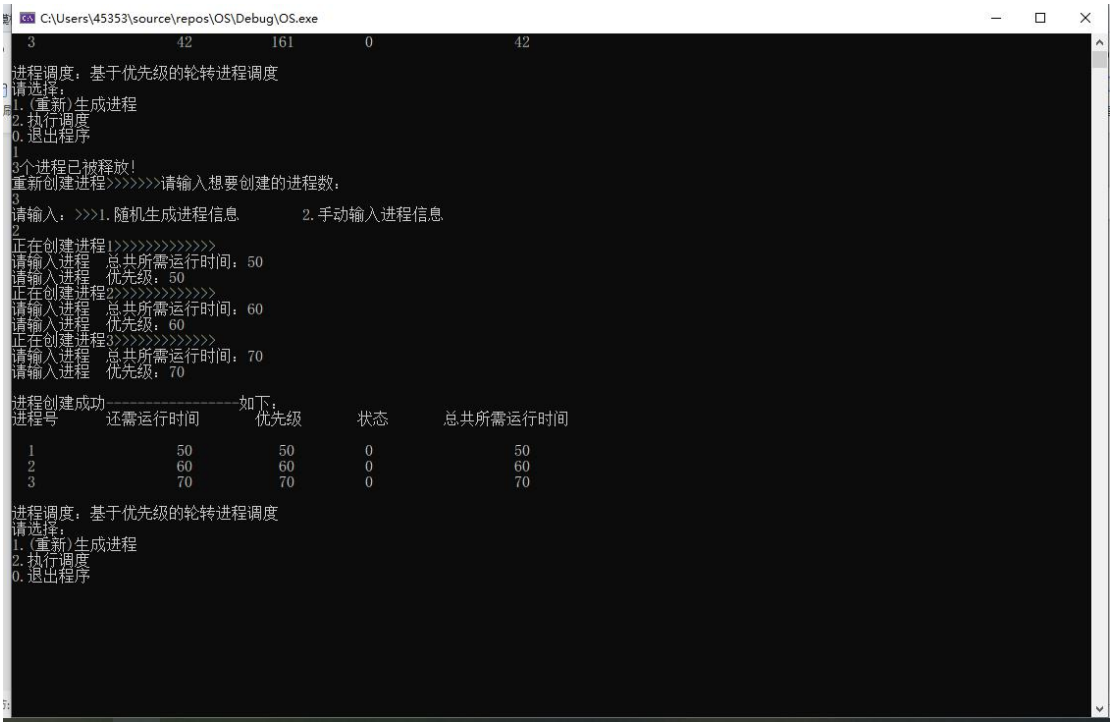


因此，需要动态创建进程数组，所以使用了进程数组指针 Task * tasks 以创建若干个进程。之后，用户还可以选择是否手动输入进程信息或者系统随机生成进程信息，以下是两种选择结果：

随机生成：



手动输入：



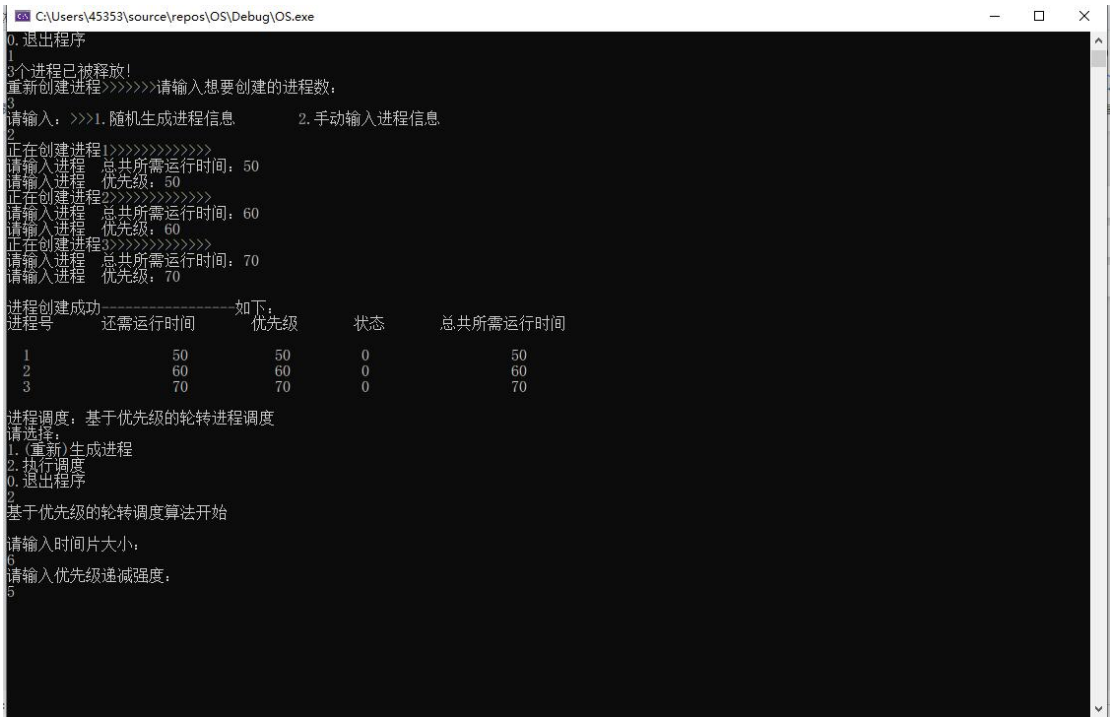
此外，为方便用户想进行多次测试，生成进程时会检查是否还有还未被释放的进程，我使用布尔值变量 isIni 来记录是否有未被释放的进程，以方便后面释放进程前的检测。如果有，则会释放之前的进程，然后重新进行进程生成。

实现：

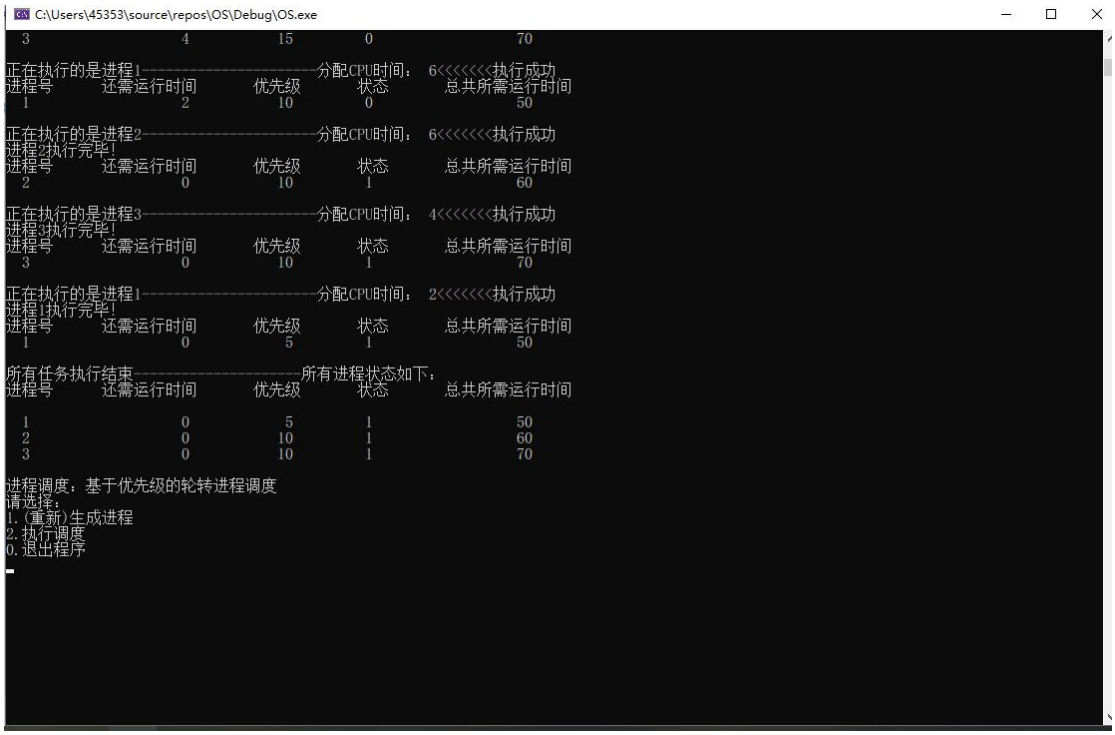
```
void initTasks() {
    if (isIni) {
            delete []tasks;
            cout << numOfTask << "个进程已被释放！\n 重新创建进程>>>>>>>";
            isIni = false;
    }

    cout << "请输入想要创建的进程数：  " << endl;
    cin >> numOfTask;

    tasks = new Task[numOfTask];//动态创建进程数组

    cout << "请输入：>>>1.随机生成进程信息          2.手动输入进程信息" << endl;
    int isRandom;
    cin >> isRandom;

    if (isRandom == 1) {//随机生成进程信息
        srand(unsigned int(time(0)));
        for (int i = 0;i < numOfTask;i++) {
            tasks[i].id = i + 1;
            tasks[i].wholetime = tasks[i].needtime = rand() % 50;
            tasks[i].state = 0;
            tasks[i].priority = rand() % 100 + 100;
        }
    }
    if (isRandom == 2) {//手动输入进程信息
        for (int i = 0;i < numOfTask;i++) {
            cout << "正在创建进程" << i+1 << ">>>>>>>>>>>>>" << endl;
            cout << "请输入进程   总共所需运行时间：";
            cin >> tasks[i].needtime;
            tasks[i].wholetime = tasks[i].needtime;
            cout << "请输入进程   优先级：";
            cin >> tasks[i].priority;
            tasks[i].id = i + 1;
            tasks[i].state = 0;
        }
        cout << endl;
    }


    cout << "进程创建成功-----------------如下：" << endl;
    printAll();
```

```
    isIni = true;
}
```

设计思路 4——调度算法：根据所给定的时间片大小，优先级最高的进程首先获得该时间片，该进程所需的 CPU 时间减少对应的时间片大小，并将其优先级降低以避免该进程一直抢占时间片，该过程便算一次调度。而后再重新索引进程数组，遍历其以找到当前优先级最高的进程，若其状态值为 1，则将其作为活跃进程进行下一次调度。我使用一个计数器用来记录当前已经调度执行完毕的进程数，每当有一个进程执行完调度，计数器便自增一次，而后将该进程的状态值置为 0（即表示已经完成调度）。当计数器的值与进程数相同时，中断 while 循环，进程调度结束，所有进程执行完毕。我在实现时，用户可以自由设置时间片大小和进程优先级递减强度，如图



执行完毕后如图：



实现：

```
void RRSchedule() {
    cout << "基于优先级的轮转调度算法开始" << endl<<endl;
    int time;
    cout << "请输入时间片大小：" << endl;
    cin >> time;

    int priLevel;
    cout << "请输入优先级递减强度：" << endl;
    cin >> priLevel;

    int finishNum = 0;//计数已经完成执行的进程数

    while (finishNum < numOfTask) {
        int maxPri = -1;//获得优先级最高且状态值为1（即待执行）的进程
        for (int i = 0;i < numOfTask;i++)
            if (tasks[i].state == 0) {
                maxPri = i;
                break;
            }
        if (maxPri == -1)
            break;
        for (int i = 0;i < numOfTask;i++) {
            if ((tasks[maxPri].priority < tasks[i].priority)&&tasks[i].state == 0)
                maxPri = i;
        }
```

```
        if (tasks[maxPri].needtime <= time) {
            cout << "正在执行的是进程" << maxPri+1 << "----------------------分配
CPU 时间：  " << tasks[maxPri].needtime <<"<<<<<<<执行成功"<< endl;
            tasks[maxPri].needtime = 0;
            tasks[maxPri].state = 1;
            finishNum++;
            cout << "进程" << maxPri + 1 << "执行完毕！" << endl;
        }
        else {
            tasks[maxPri].needtime -= time;
            cout << "正在执行的是进程" << maxPri+1 << "----------------------分配
CPU 时间：  " << time <<"<<<<<<<执行成功"<< endl;
        }
        tasks[maxPri].priority -= priLevel;//递减优先级
        print(maxPri);
    }

    cout << "所有任务执行结束--------------------所有进程状态如下：" << endl;
    printAll();
}
```

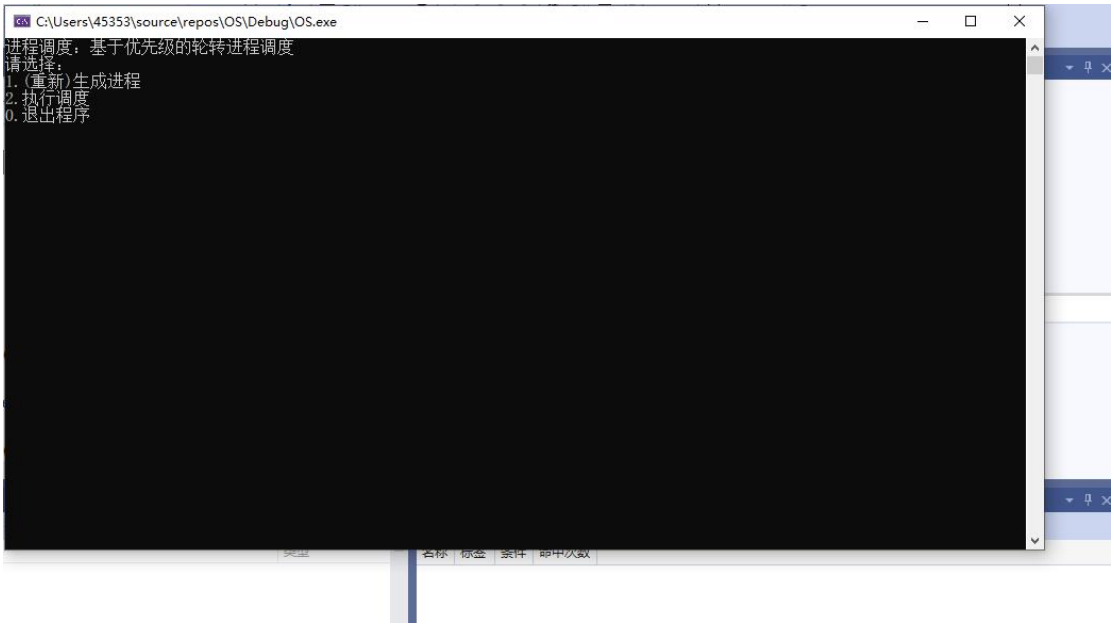
其他的一些辅助输出函数和 main 函数：

```
void printAll() {
    cout << "进程号      还需运行时间      优先级      状态      总共所需运行时间"
<< endl << endl;
    for (int i = 0;i < numOfTask;i++) {
        cout << setw(3) << tasks[i].id << setw(20) << tasks[i].needtime << setw(13)
<<tasks[i].priority << setw(10) << tasks[i].state<< setw(20) <<
tasks[i].wholetime<< endl;
    }
    cout << endl;
}

void print(int id) {
    cout << "进程号      还需运行时间      优先级      状态      总共所需运行时间"
<< endl;
    cout << setw(3) <<  tasks[id].id << setw(20) << tasks[id].needtime << setw(13)
<< tasks[id].priority <<setw(10) << tasks[id].state << setw(20) <<
tasks[id].wholetime << endl << endl;
}

int main()
{
```

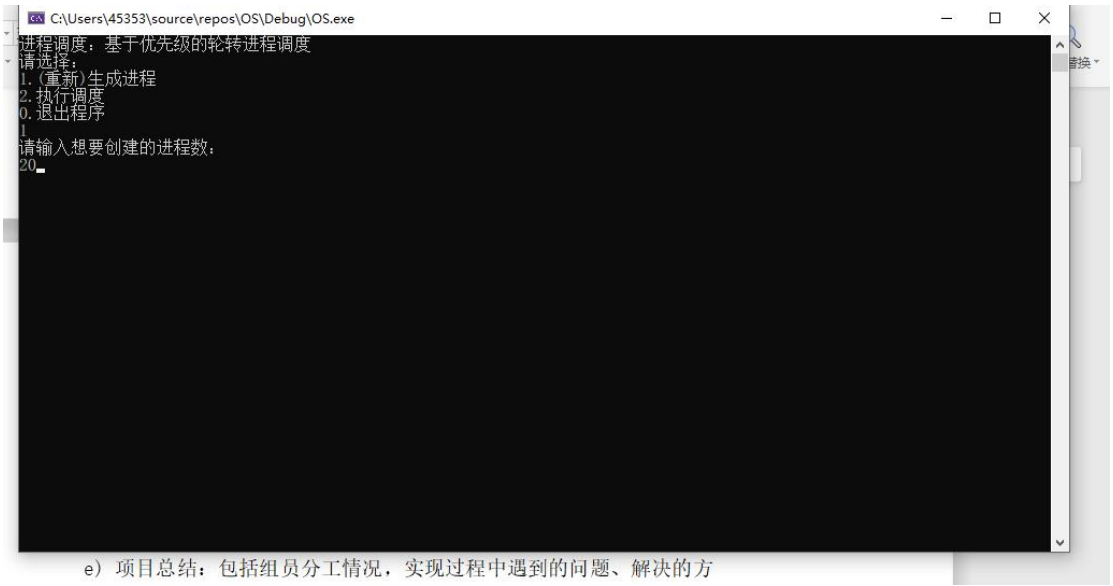
```
    int choice = menu();
    while (choice) {
        switch (choice) {
        case 1:
            initTasks();//选择 1 执行生成进程
            break;
        case 2:
            if (isIni)
                RRSchedule();//若已经生成进程才执行调度
            else
                cout << "请先初始化进程！" << endl;
            break;
        default:
            cout << "无效输入！\n 请重新选择：\n";
        }
        choice = menu();
    }

    if (isIni) {//释放内存
        delete[]tasks;
        cout << numOfTask << "个进程已被释放！";
    }

    return 0;
}
```

三、项目测试和运行的结果

程序运行：



选择（重新）生成进程

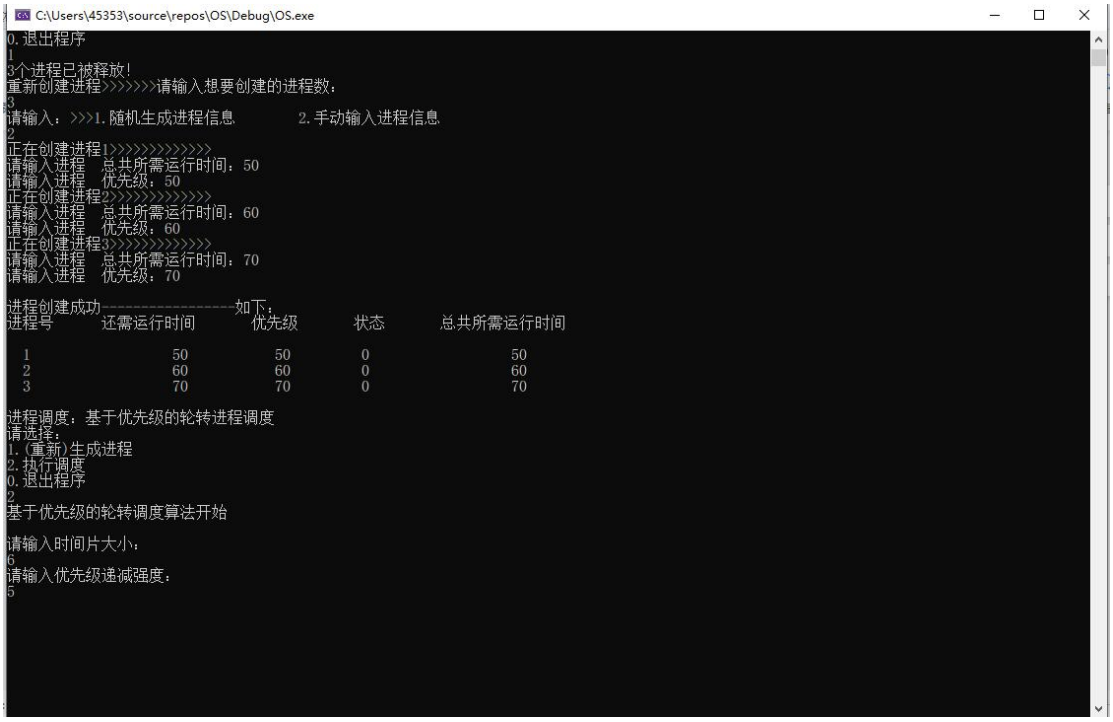


e）项目总结：包括组员分工情况，实现过程中遇到的问题、解决的方

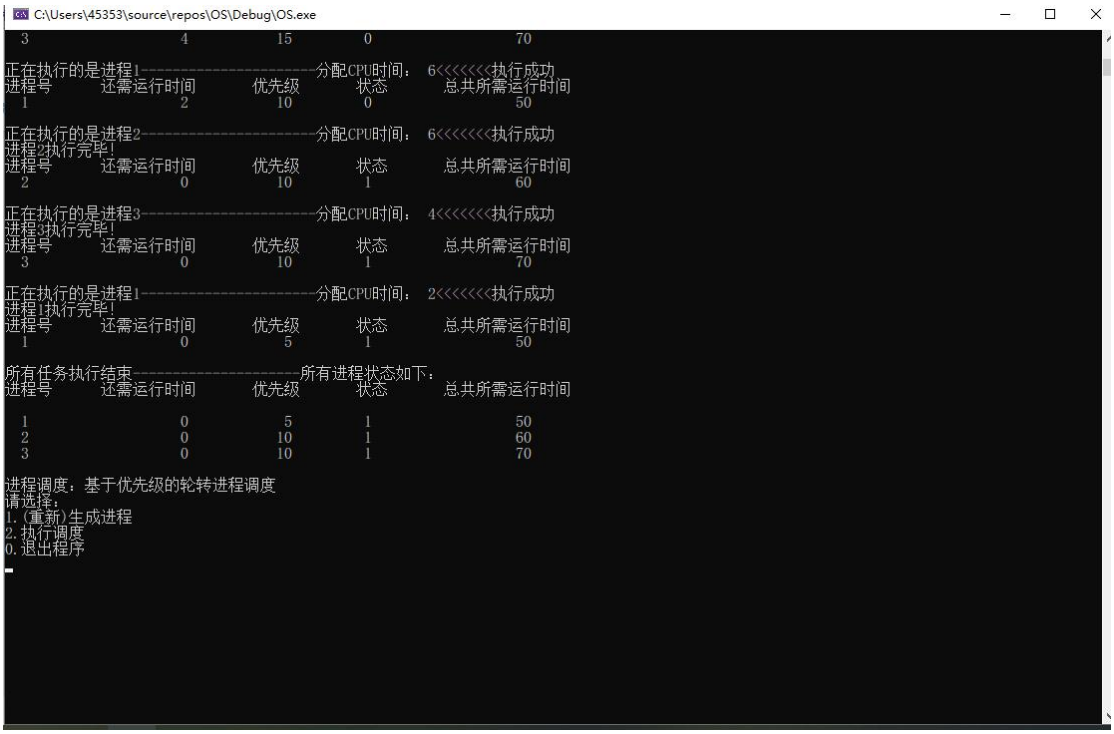
随机生成：

手动输入进程所需时间和优先级：

选择进程调度并给定时间片大小和优先级递减强度：



执行完毕后如图:



退出程序:

四、项目总结

该项目由我（赵水）一人完成
在设计过程中基本上没有遇到问题。

通过本次设计，我了解到了进程调度是为了使系统各个服务运行更加高效，通过优先级的方式来体现不同任务的紧急程度，而给定适当的时间片大小可以使 CPU 时间分配的更加灵活。