

华北电力大学

课程设计报告

(2013-- 2014 年度第 1 学期)

名 称： 操作系统原理课程设计

院 系：

班 级：

学 号：

学生姓名：

指导教师：

设计周数： 1

成 绩：

日期： 2013 年 11 月 15 日

《操作系统原理课程设计 B》课程设计

任 务 书

一、 目的与要求

- 1. 理解和掌握操作系统的基本概念、基本组成与工作原理；
- 2. 理解和掌握操作系统中主要功能模块的工作原理及其实现算法；
- 3. 掌握软件模块设计技能；熟悉并能较好地利用软件开发环境独立编程、调试和分析程序运行情况，逐渐形成创新思维和从事系统软件的研究和开发能力。

二、 主要内容

采用某种程序设计语言，设计与开发一个仿真操作系统，要求具备进程调度、存储器管理、文件及设备管理等基本功能，具体内容包括：

- 1. 模拟单处理器系统的进程调度，加深对进程的概念及进程调度算法的理解；
- 2. 编写一个多线程程序，加深对现代操作系统中线程的理解；
- 3. 编写和调试一个银行家算法程序，加深了解有关资源申请、避免死锁等概念，并体会和了解死锁和避免死锁的具体实施方法；
- 4. 用可变分区算法模拟主存的分配和回收，加深对存储器管理的理解，熟悉各种主存分配方法；

根据个人情况选做其中三个。

三、 进度计划

序号	设计(实验)内容	完成时间	备注
1	熟悉开发环境及程序设计语言	1 天	
2	系统的分析、设计	2 天	
3	调试与集成	1 天	
4	验收，提交实验报告	1 天	

四、 设计（实验）成果要求

至少完成主要内容中的 3 项，统一使用课程设计报告书，文字清楚、工整。

学生姓名：

指导教师：

2013 年 11 月 15 日

一、课程设计(综合实验)的目的与要求

1. 理解和掌握操作系统的基本概念、基本组成与工作原理；
2. 理解和掌握操作系统中主要功能模块的工作原理及其实现算法；
3. 掌握软件模块设计技能；熟悉并能较好地利用软件开发环境独立编程、调试和分析程序运行情况，逐渐形成创新思维和从事系统软件的研究和开发能力。

二、实验环境

Visual C++6.0 和 JCreator

三、设计（实验）正文

多线程：

用 `Runnable` 接口和继承 `Thread` 类的方法实现多线程程序。

设计龟兔赛跑，赛程 1000 米，兔子跑 5 米，乌龟跑 1 米，兔子每 20 米休息 500 毫秒，乌龟每 100 米休息 500 毫秒。谁先到终点就结束程序，并显示获胜方。

调度算法：

编程实现先来先服务和短作业优先调度算法。

先来先服务算法是每次调度从就绪队列中选择一个最先进入该队列的进程，为之分配处理机，使之投入运行。该进程一直运行到完成或发生某件事而阻塞后才放弃处理机。

短作业优先算法是从后备队列中选择一个或若干个估计时间最短的作业，将他们调入内存运行。

程序要求

进程个数 n ；每个进程的到达时间 $T_1, \dots, T_n$ 和服务时间 $S_1, \dots, S_n$ ；选择算法 1-FCFS, 2-SJF。

要求采用先来先服务 FCFS 和短作业优先 SJF 分别调度进程运行，计算每个进程的周转时间，带权周转时间，并且计算所有进程的平均周转时间，带权平均周转时间；

要求输出计算出来的每个进程的周转时间，带权周转时间，所有进程的平均周转时间，带权平均周转时间。

银行家：

2.算法数据结构

1)可利用资源向量 `Available` ，它是一个最多含有 100 个元素的数组，其中的每一个元素代表一类可利用的资源的数目，其初始值是系统中所配置的该类全部可用资源数

目。其数值随该类资源的分配和回收而动态地改变。如果 $Available(j) = k$ ，表示系统中现有 j 类资源 k 个。

2)最大需求矩阵 Max ，这是一个 $n \times m$ 的矩阵，它定义了系统中 n 个进程中的每一个进程对 m 类资源的最大需求。如果 $Max(i, j) = k$ ，表示进程 i 需要 j 类资源的最大数目为 k 。

3)分配矩阵 $Allocation$ ，这也是一个 $n \times m$ 的矩阵，它定义了系统中的每类资源当前一分配到每一个进程的资源数。如果 $Allocation(i, j) = k$ ，表示进程 i 当前已经分到 j 类资源的数目为 k 。 $Allocation_i$ 表示进程 i 的分配向量，有矩阵 $Allocation$ 的第 i 行构成。

4)需求矩阵 $Need$ ，这还是一个 $n \times m$ 的矩阵，用以表示每个进程还需要的各类资源的数目。如果 $Need(i, j) = k$ ，表示进程 i 还需要 j 类资源 k 个，才能完成其任务。 $Need_i$ 表示进程 i 的需求向量，由矩阵 $Need$ 的第 i 行构成。

5)上述三个矩阵间存在关系： $Need(i, j) = Max(i, j) - Allocation(i, j)$ ；

3.银行家算法

设 $Request[i]$ 是进程 i 的请求向量，如果 $Request[i, j] = K$ ，表示进程 i 需要 K 个 j 类型的资源。当 i 发出资源请求后，系统按下述步骤进行检查：

1)如果 $Request_i \leq Need$ ，则转向步骤 2；否则，认为出错，因为它所请求的资源数已超过它当前的最大需求量。

2)如果 $Request_i \leq Available$ ，则转向步骤 3；否则，表示系统中尚无足够的资源满足 i 的申请， i 必须等待。

3)系统试探性地把资源分配给进程 i ，并修改下面数据结构中的数值：

$Available = Available - Request_i$

$Allocation_i = Allocation_i + Request_i$

$Need_i = Need_i - Request_i$

4)系统执行安全性算法，检查此次资源分配后，系统是否处于安全状态。如果安全才正式将资源分配给进程 i ，以完成本次分配；否则，将试探分配作废，恢复原来的资源分配状态，让进程 i 等待。

四、课程设计（综合实验）总结或结论

通过本次实验，基本了解了线程的概念，作用，方法以及使用规则。

也了解到 `run` 方法所包含的代码就是和其他线程同时运行的代码以达到同一时刻运行多段代码的目的。当终止了 `run` 以后。这个线程也就结束了。调用线程的 `start` 方法才会执行 `run` 方法。

对调度算法特别是 **FCFS** 和 **SJF** 有了更深的理解，而且锻炼了我的思维能力，使我能更全面地思考问题，以后还需要多做些这方面的练习。

对于银行家算法也更深刻的知道银行家算法是避免死锁的一种重要方法，通过编写一个简单的银行家算法程序，加深了解有关资源申请、避免死锁等概念，并体会和了解死锁和避免死锁的具体实施方法。死锁的产生，必须同时满足四个条件，即一个资源每次只能由一个进程；第二个为等待条件，即一个进程请求资源不能满足时，它必须等待，但它仍继续保持已得到的所有其他资源；第三个为非剥夺条件，即在出现死锁的系统中一定有不可剥夺使用的资源；第四个为循环等待条件，系统中存在若干个循环等待的进程，即其中每一个进程分别等待它前一个进程所持有的资源。防止死锁的机构只能确保上述四个条件之一不出现，则

系统就不会发生死锁。

总体上此次实验锻炼了自己的耐心。写程序重要的品质就应该是耐心了，如果没有了耐心是写不出程序的，出现了错误稍微长一点时间调试不出就放弃，你永远无法写出程序。写程序切记，戒骄戒躁。极大的调高了自己编程的能力。每一条程序自己都要读懂，每一处错误自己都要亲手调通，这个过程是看书得不到的。编程要的就是一种感觉，亲自动手去写才能培养这种感觉。

五、参考文献

汤子赢 操作系统 西安电子科技大学出版社，第三版. 出版日期

张白一，崔尚森，面向对象程序设计—Java（第二版），西安电子科技大学出版社

附录（设计流程图、程序、表格、数据等）

多线程：[java]

源代码：

```
import java.util.Date;
```

```
public class Test extends Thread{
```

```
    private int tortoise_walk = 0; //          乌龟已跑长度存放变量
```

```
    private int rabbit_walk = 0; //          兔子已跑长度存放变量
```

```
    private int finish = 200; //          终点
```

```
    private volatile boolean hasWinner = false; //          胜利者诞生
```

```
class Tortoise_Run implements Runnable {
```

```
    @Override
```

```
    public void run() {
```

```
        try {
```

```
            while (!hasWinner) {
```

```

        if (tortoise_walk % 100 == 0 && (tortoise_walk != 0||tortoise_walk>=finish))
        { // 乌龟每 100 米休息 500 毫秒

                乌龟休息中.....

                Thread.sleep(500);

                }

        tortoise_walk++;

                乌龟已跑                米

        }

} catch (InterruptedException e) {

        e.printStackTrace();

        }

}

```

```

class  Rabbit_Run implements Runnable {

        @Override

        public void run() {

                try {

                        while (!hasWinner) {

                                if (rabbit_walk % 20 == 0 && (rabbit_walk != 0||rabbit_walk>=finish)) { //

                                        兔子每 20 米休息 500 毫秒

                                                兔子休息中.....

                                Thread.sleep(500);

                                }

                                rabbit_walk=rabbit_walk+5; //                每秒跑 5 步

                                        兔子已跑                米

                                }

}

```



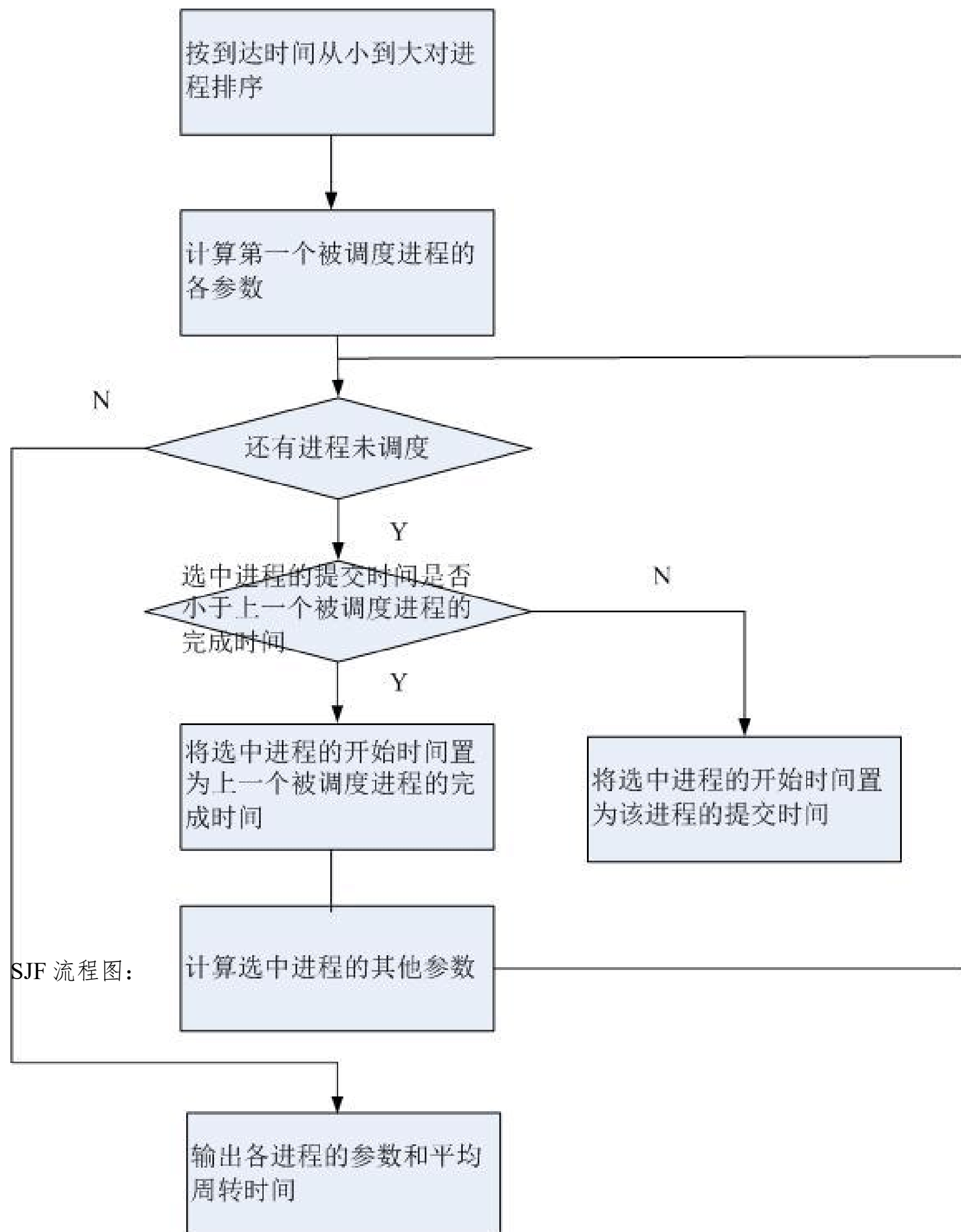
```
兔子已跑5米
兔子已跑10米
兔子已跑15米
兔子已跑20米
兔子休息中.....
乌龟已跑1米
乌龟已跑2米
乌龟已跑3米
乌龟已跑4米
乌龟已跑5米
乌龟已跑6米
乌龟已跑7米
乌龟已跑8米
乌龟已跑9米
乌龟已跑10米
乌龟已跑11米
乌龟已跑12米
乌龟已跑13米
乌龟已跑14米
```

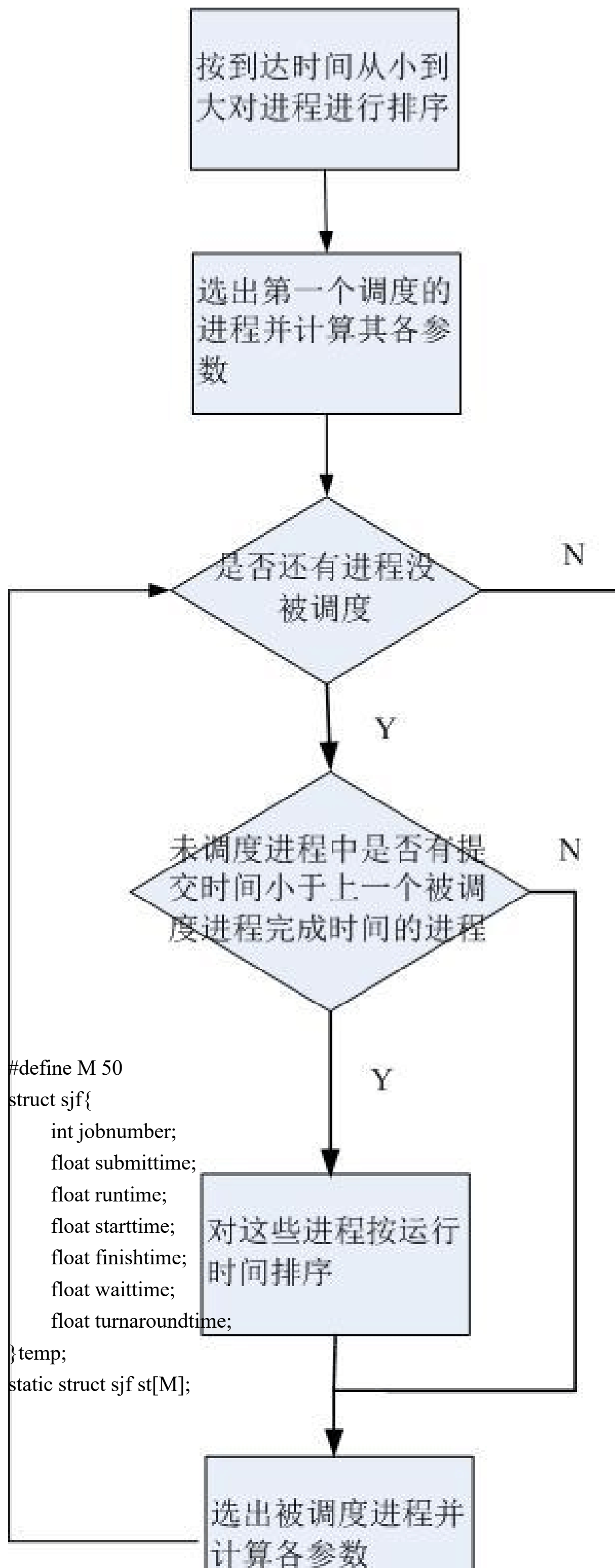
```
乌龟已跑99米
乌龟已跑100米
乌龟休息中.....
兔子已跑25米
兔子已跑30米
兔子已跑35米
兔子已跑40米
兔子休息中.....
乌龟已跑101米
乌龟已跑102米
乌龟已跑103米
乌龟已跑104米
乌龟已跑105米
乌龟已跑106米
乌龟已跑107米
乌龟已跑108米
乌龟已跑109米
```

```
乌龟已跑195米
乌龟已跑196米
乌龟已跑197米
乌龟已跑198米
乌龟已跑199米
乌龟已跑200米
乌龟胜利
比赛结束: Wed Nov 20 10:39:42 CST 2013毫秒
所耗时间: 578毫秒
兔子=40 乌龟=200
兔子已跑45米
Press any key to continue..._
```

调度算法:

FCFS流程图





```
void input(struct sjf *p,int N)
```

```
{
```

```
    int i;
```

```
    for(i=0;i<N;i++)
```

```
    {
```

```
    }
```

```
}
```

```
void print(struct sjf *p,int N)
```

```
{
```

```
    int k;
```

```
    float h,g;
```

```
    for(k=1;k<N;k++)
```

```
    for(k=0;k<N;k++)
```

```
    {
```

```
        h+=p[k].turnaroundtime;
```

```
time,p[k].starttime,p[k].finishtime,p[k].waittime,p[k].turnaroundtime);
```

```
    }
```

```
    g=h/N;
```

```
}
```

```
/*                                */
```

```
void sort1(struct sjf *p,int N)
```

```
{
```

```
    int i,j;
```

```
    for(i=0;i<N;i++)
```

```
    for(j=0;j<=i;j++)
```

```
    if(p[i].submittime<p[j].submittime)
```

```
    {
```

```
        temp=p[i];
```

```
        p[i]=p[j];
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/508050037006006046>