**五、进程调度**

**5.1目的**

实现按先来先服务FCFS、短作业优先SJF以及时间片轮转算法调度进程的模拟过程。

**5.2内容**

调度算法中模拟三种调度算法，输出每个作业的到达时刻、运行时间、开始时刻、完成时刻、周转时间、响应时间、平均周转时间、平均响应时间。

银行家算法中模拟银行家算法的过程，给出进资源总数、MAX矩阵、Allocation矩阵，求出Need矩阵、Available矩阵。首先显示出求安全序列的过程，接下来对每次某个进程申请资源，模拟把资源给它的情况。若存在安全序列，真正的把资源分配给该进程，并更新所有矩阵；否则说明资源不能分配给该进程，原所有矩阵不发生改变。

**5.3数据结构**

调度算法中的数据结构如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Define the Array

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const int MAX\_P = 5; // 作业个数

int arriveTime[MAX\_P]; // 到达时刻

int runTime[MAX\_P]; // 运行时间

int startTime[MAX\_P]; // 开始时刻

int finishTime[MAX\_P]; // 完成时刻

int turnTime[MAX\_P]; // 周转时间

double responseTime[MAX\_P]; // 响应时间

double averTurnTime; // 平均周转时间

double averResTime; // 平均响应时间

set<int> finishProcess; // 完成进程集合

typedef struct stfNode // 最短作业有限优先节点

{

int id; // 存作业号

int arriveTime; // 当前作业的到达时刻

int runTime; // 当前作业的运行时间

}stf;

stf stfArray[MAX\_P]; // 最短作业

const int Q = 1; // 时间片大小

银行家算法中的数据结构：

const int MAX\_Z = 3; // 资源种类

const int processNum = 5; // 作业个数

const int maxn = 11; // 数组长度

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The init Array

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int MAX[maxn][maxn]; // Define MAX

int SUM[maxn]; // Define SUM\_Z

int ALLOCATION[maxn][maxn]; // Define Allocation

int NEED[maxn][maxn]; // Define Need

int AVAILABLE[maxn]; // Define Available

set<int> hasFinish; // Define the number of process has finished

vector<int> safePath; // Storage the safe Path

int WORK[maxn][maxn]; // Storage the set of Work

int posWork; // Define the crow of Work

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The copy Array

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int copy\_MAX[maxn][maxn]; // Define MAX

int copy\_SUM[maxn]; // Define SUM\_Z

int copy\_ALLOCATION[maxn][maxn]; // Define Allocation

int copy\_NEED[maxn][maxn]; // Define Need

int copy\_AVAILABLE[maxn]; // Define Available

set<int> copy\_hasFinish; // Define the number of process has finished

vector<int> copy\_safePath; // Storage the safe Path

int copy\_WORK[maxn][maxn]; // Storage the set of Work

int copy\_posWork; // Define the crow of Work

**5.4算法设计及流程图**

调度算法：

FCFS：按照作业来的先后顺序进行调度，而不管所需执行时间的长短。



SJF:在同一段时间内可以做的所有作业，作业执行需时越短，优先级越高，先做它。

****

RR调度：让就绪队列上的每个进程，每次只运行一个时间片，根据时间片的选定不同，最后产生的结果也会有所不同。

****

银行家算法：避免死锁的方法之一，通过判断是否存在安全序列，来分析其是否会发生死锁。存在安全序列后，再判断某一时刻某进程申请资源是否应给分配。

****

**5.5小结**

算法在具体实现时要以它本身特性为根据，把思路向外延伸，考虑各种细节处理，这是很锻炼思维的。银行家算法直接暴力模拟即可。调度算法的最短作业优先和时间片轮转考虑的细节要多点，但仔细想来还是要抓住本身特性，即最短作业优先是在当前时刻能做的作业里调执行需时最短的先做，时间片轮转找好时间片轮转的顺序。