



**JISHOU　UNIVERSITY**

**本科生课程设计**

**操作系统原理课程设计**

|  |  |
| --- | --- |
| **题目：** | 进程调度算法 |
| **作者：** | 刘家骏，肖定，刘卉 |
| **学号：** | 2019404022、2019404024、2019404026 |
| **所属学院：** | 信息科学与工程学院 |
| **班级：** | 19计科1班 |
| **完成时间：** | 2021年12月25日 |

**评分表**

|  |  |  |
| --- | --- | --- |
| 序号 | 评分项 | 分数 |
| 1 | 出勤及表现 |  |
| 2 | 功能完善 |  |
| 3 | 文档格式与文字表达 |  |
| 4 | 代码风格 |  |
| 5 | 测试调试 |  |
| 6 | 图表 |  |
| 7 | 功能设计 |  |
| 8 | 目录结构设计 |  |
| 9 | 界面与流程 |  |
| 10 | 加分 |  |
| 合计 | |  |

课程设计小组组长签字：

年 月 日

目录

[一.问题描述与分析 4](#_Toc17798)

[1.1.设计构想 4](#_Toc31040)

[1.2基本要求 4](#_Toc9524)

[1.3.需求分析 4](#_Toc30880)

[二. 设计分析 5](#_Toc12922)

[2.1概要设计 5](#_Toc31422)

[2.2详细设计 5](#_Toc18921)

[三. 系统编译与调试分析 10](#_Toc2103)

[四. 使用说明 10](#_Toc7883)

[五. 测试结果 11](#_Toc10181)

[六. 课程设计心得体会 20](#_Toc9002)

[七.参考文献： 21](#_Toc14512)

[八.附录 21](#_Toc16616)

# 一.问题描述与分析

## 1.1.设计构想

程序能够完成以下操作:选择调度算法;查看历史记录;创建进程;进程调度:进程创建完成后就选择进程调度算法，每次执行的结果都在屏幕上输出。

## 1.2基本要求

1.利用高级语言实现七种不同及进程调度算法：先来先服务调度算法，优先级调度算法，时间片轮转调度算法，短进程优先调度算法，最短剩余时间调度算法，最高响应比优先调度算法，多级反馈队列调度算法

2.通过实验理解有关进程控制块进程队列等的概念。

## 1.3.需求分析

1、需求：在多道程序环境下，主存中有着多个进程，其数目往往多于处理机数目，要使这多个进程能够并发地执行，这就要求系统能按某种算法，动态地把处理机分配给就绪队列中的一-个进程，使之执行。分配处理机的任务是由处理机:调度程序完成的。由于处理机是最重要的计算机资源，提高处理机的利用率及改善系统必(吞吐量、响应时间)，在很大程度上取决于处理机调度性能的好坏，因而，处理机调度便成为操作系统设计的中心问题之一。

2、软件的基本功能：本软件在vs code环境下实现先来先服务、短作业优先、高响应比优先、时间片轮转调度算法、优先级调度算法和多级反馈队列调度算法，满足不同需求调度。

3、输入/输出形式：I/O输入：data数组键盘输入。I/O输出：运行结果从屏幕输出。文本输出：新的键盘输入的data数组添加到历史数据中。

4、测试数据要求：对data数组赋值时，赋值的长度不可大于data数据的长度。

**1.4理论依据**

为了描述和管制进程的运行，系统为每个进程定义了一个数据结构一进程控制块，程控制块中记录了操作系统所需的、用于描述进程的当前情况以及控制进程运行的全部信息,系统总是通过进程控制块对进程进行控制，亦即，系统是根据进程的进程控制块而不是任何别的什么而感知进程的存在的，进程控制块是进程存在的惟一标志。本次课程设计用不同结构体代替进程控制块的功能。

**1.5课程任务**

(1)用C++编程实现操作模拟操作系统进程调度子系统的基本功能;运用多种算法实现对进程的模拟调度。

(2)通过编写程序实现先来先服务调度算法，优先级调度算法，时间片轮转调度算法，短进程优先调度算法，最短剩余时间调度算法，最高响应比优先调度算法，多级反馈队列调度算法等七个算法，使学生进一步掌握进程调度的概念和算法，加深对处理机分配的理解。

(3)通过控制台显示用户界面。

# 设计分析

## 2.1概要设计

数据结构分七个算法：先来先服务调度算法，优先级调度算法，时间片轮转调度算法，短进程优先调度算法，最短剩余时间调度算法，最高响应比优先调度算法，多级反馈队列调度算法

typedef struct FSA

{

    int pro\_num; //进程号

    int time;    //进程所需时间

} fsa;

typedef struct PSA

{

    int pro\_num;  //进程号

    int time;     //进程所需时间

    int priority; //优先级

    bool operator<(const struct PSA &a) const

    {

        return priority < a.priority;

    }

} psa;

typedef struct TSA

{

    int pro\_num; //进程号

    int time;    //进程所需时间

    bool operator<(const struct TSA &a) const

    {

        return time < a.time;

    }

} tsa;

typedef struct SSA

{

    int pro\_num; //进程号

    int time;    //进程所需时间

    bool operator<(const struct SSA &a) const

    {

        return time < a.time;

    }

} ssa;

typedef struct RSSA

{

    int pro\_num; //进程号

    int time;    //进程所需时间

    bool operator<(const struct RSSA &a) const

    {

        return time < a.time;

    }

} rssa;

typedef struct HSA

{

    int pro\_num;  //进程号

    int time;     //进程所需时间

    int waittime; //等待时间

    float prinum; //优先数

    int finish;   //是否完成

    bool operator<(const struct HSA &a) const

    {

        return prinum > a.prinum;

    }

} hsa;

typedef struct MSA

{

    int pro\_num;  //进程号

    int time;     //进程所需时间

    int priority; //优先级

    int waittime; //进程等待时间

    bool operator<(const struct MSA &a) const

    {

        return priority < a.priority;

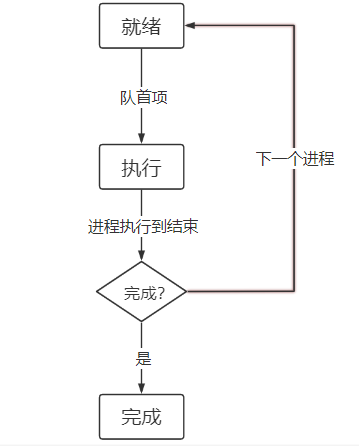
    }

} msa;

## 2.2详细设计

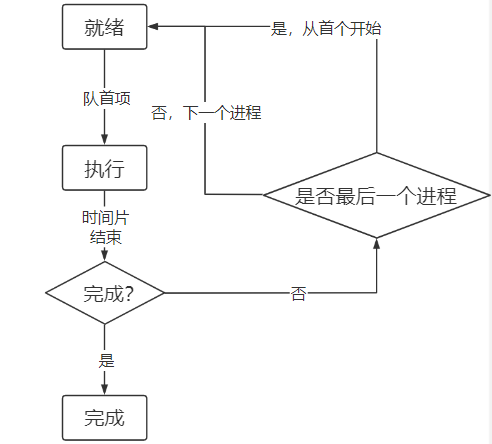
1. 先来先服务调度算法：

按照进程进入就绪队列的先后次序进行选择。一旦一个进程得到处理机，他就会一直运行下去，直到该进程完成任务，才会让出处理机。



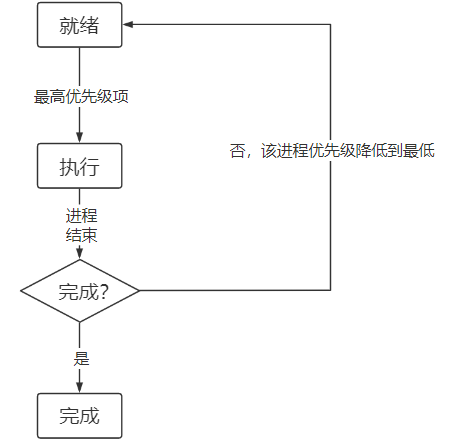
1. 时间片轮转调度算法：

在时间片轮转调度算法中，系统将所有的就绪进程排成一个就绪队列。 系统可设置每隔一定时间(如30ms)便产生一次中断，去激活进程调度程 序进行调度，把CPU分配给下一个进程，并令其执行一个时间片。当它 运行完毕后，又把处理机分配给就绪队列中下一个进程，也让其执行一 个时间片，当到队列尾再回到队首。



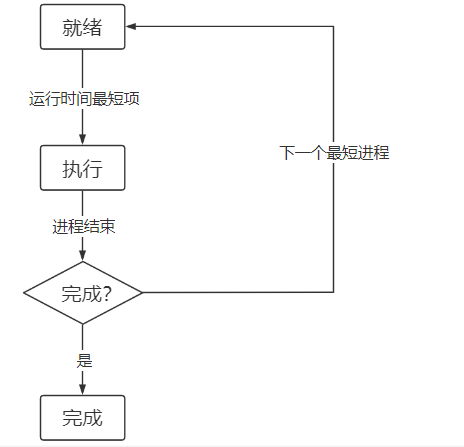
1. 优先级调度算法：

优先级调度算法是指在创建进程之初便确定的，由优先级高的开始执行。



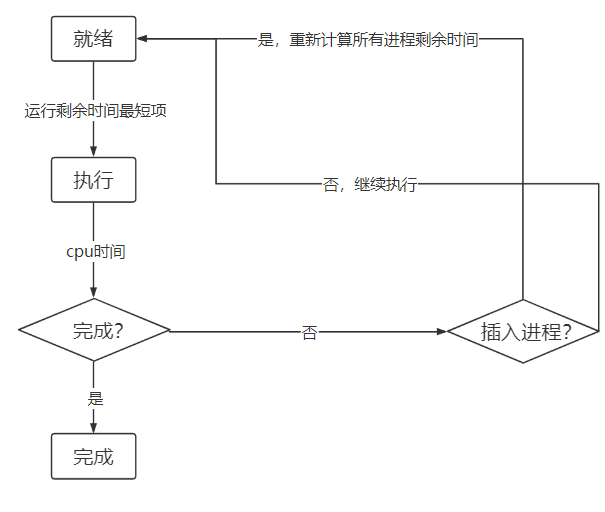
1. 短进程优先调度算法：

短进程优先调度算法是从进程的就绪队列中挑选运行时间最短进程进入主存运行，一旦选中某个短进程，应该尽可能快地完成并让出处理机。这样减少了就绪队列中等待的进程数，同时也缩短了进程的平均等待时间



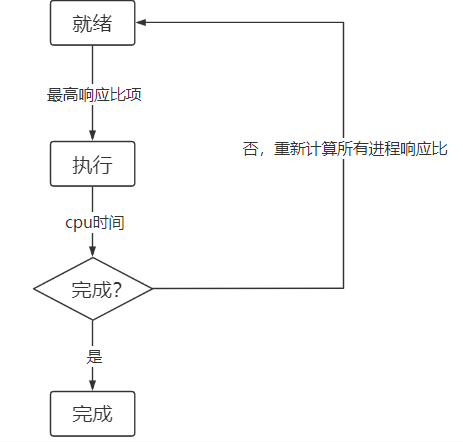
1. 最短剩余时间优先调度算法：

最短剩余时间优先调度算法是将短进程优先调度算法用于分时环境的变形。



1. 最高响应比优先调度算法：

最高响应比优先调度算法是指在创建进程之初，先赋予其一个优先级， 然后其值随着进程的推进或等待时间的增加而改变响应比，以便获得更 好的调度性能。若所有的就绪进程具有各不相同的优先级初值，那么对 于优先级初值低的进程，在等待了足够的时间后，也可以获得处理机。



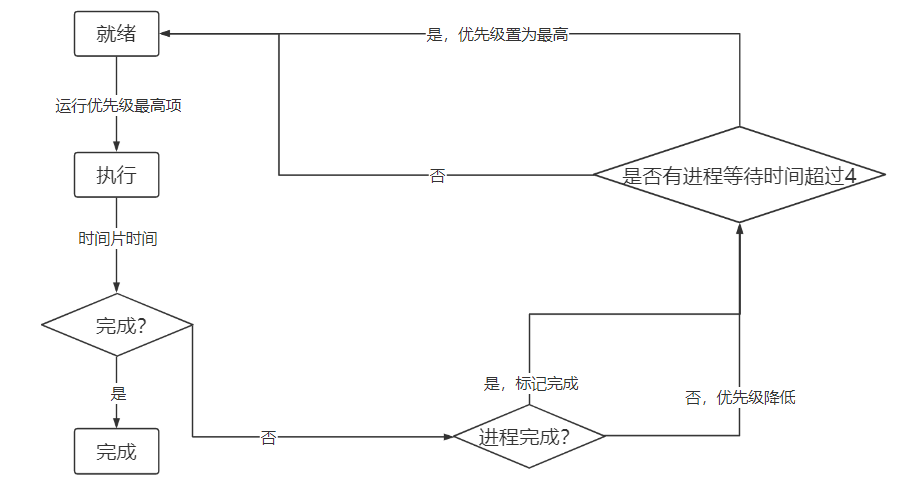
1. 多级反馈队列调度算法

短进程优先调度算法或最短剩余时间优先调度算法均是在估计的进程运行时间基础上进行调度的，但在程序开发环境或其他情况下，往往难以估计进程的运行时间。这里所研究的算法是时间片轮转调度算法的发展，不必估计进程运行时间。系统中有多个进程就绪队列，每个就绪队列对应一个调度级别。第1级队列的优先级最高，以下各级队列的优先级逐次降低。调度时，选择高优先级队列中的第1个就绪进程。

规则1：如果A的优先级大于B的优先级，运行A不运行B  
规则2：如果A的优先级等于B的优先级，轮转运行A和B

规则3：工作进入系统时，放在最高优先级（最上层）队列

规则4：工作用完整个时间片后，降低其优先级（移入低一级队列）  
规则5：每经过一段时间，就将系统中所有工作重新加入最高优先级队列



# 系统编译与调试分析

文件写入会覆盖上一条记录，使用追加模式

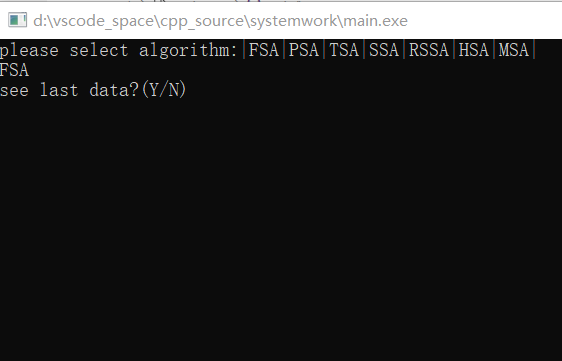
算法方面每个算法设置一个函数接口，同时使用数组存储各个进程的时间

# 使用说明

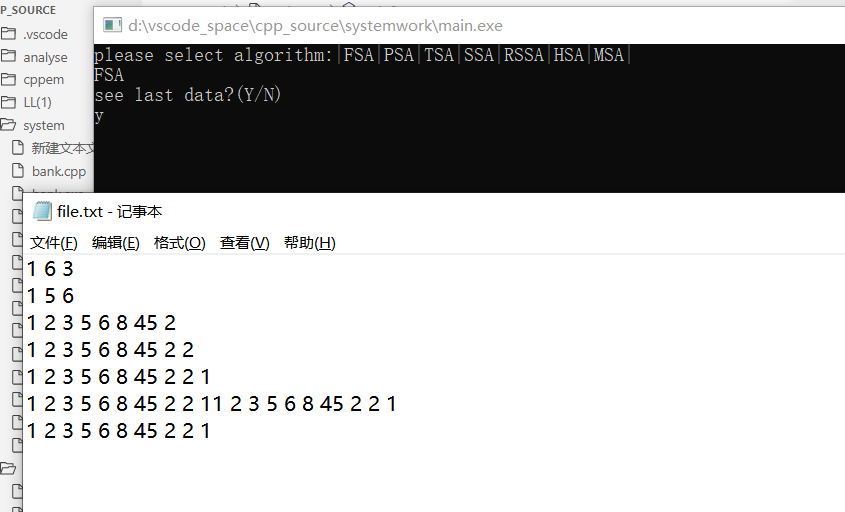
选择算法->是否查看历史记录->输入进程所需时间->对应算法运行->输入exit退出系统

# 测试结果

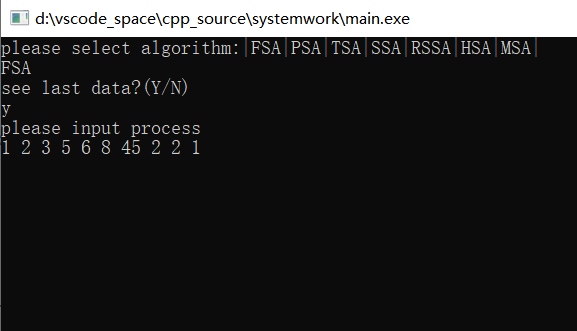
1. **选择算法**



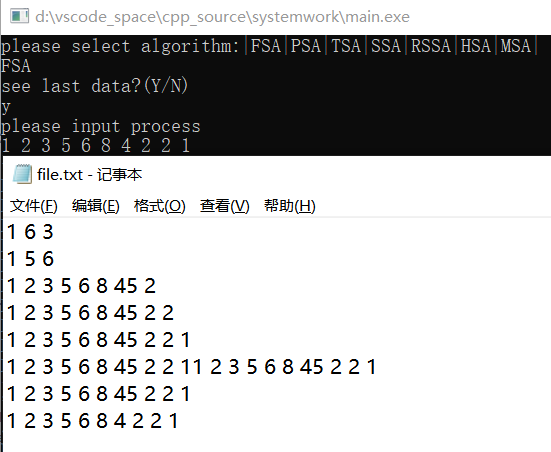
1. **是否查看历史记录**



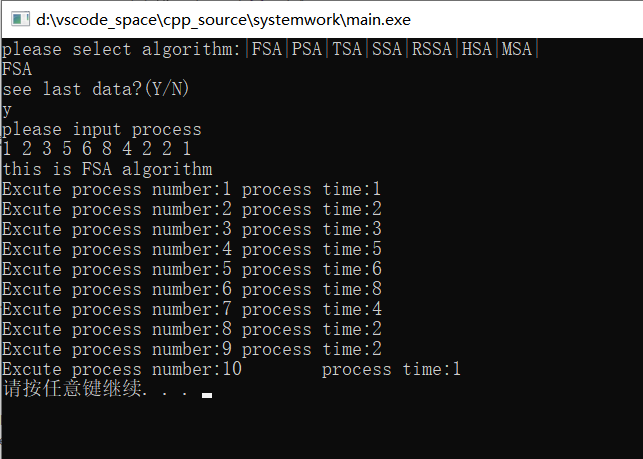
1. **输入进程所需时间（添加到历史记录中，自动生成进程号）**



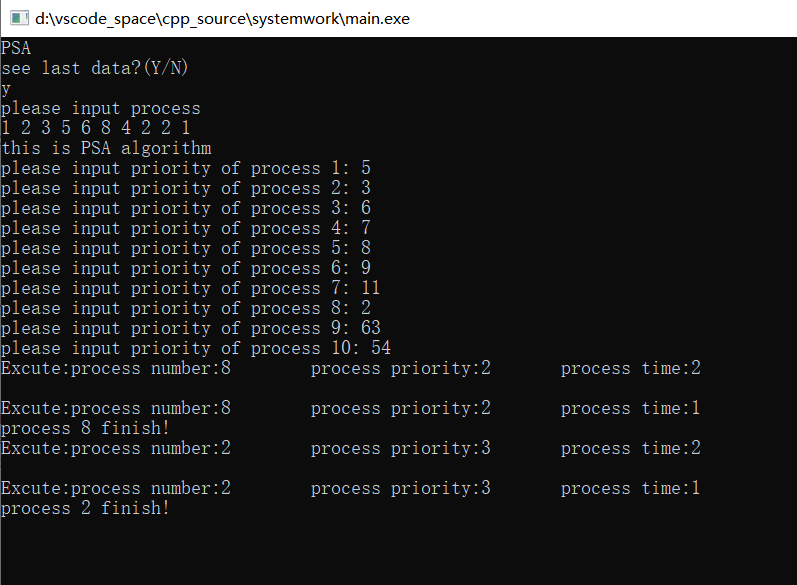
非历史记录自动添加到历史记录中

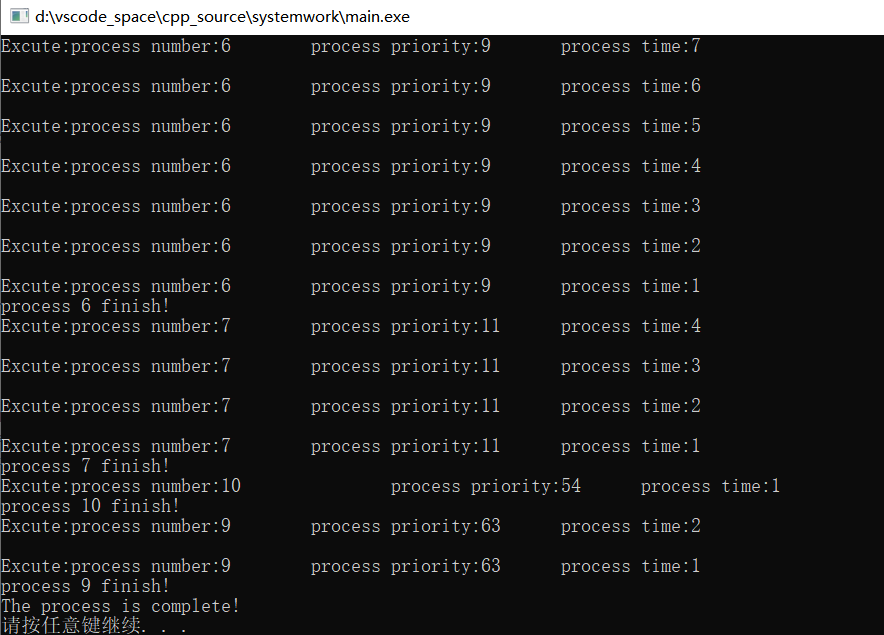


1. **算法对进程实现不同操作**
   1. **先来先服务调度算法（FSA）**

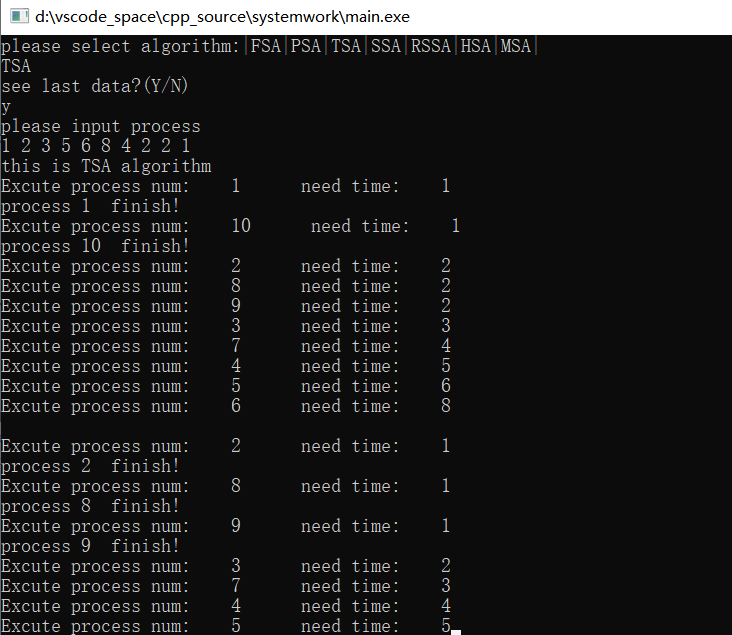
1

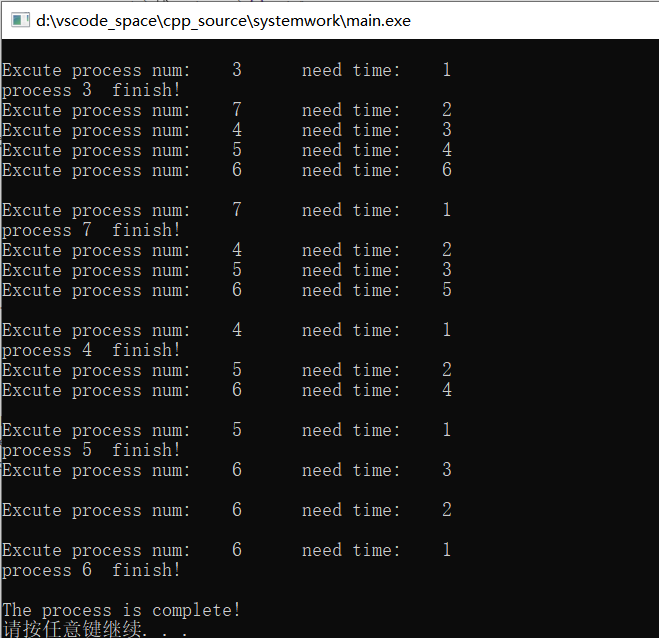
* 1. **优先级调度算法**



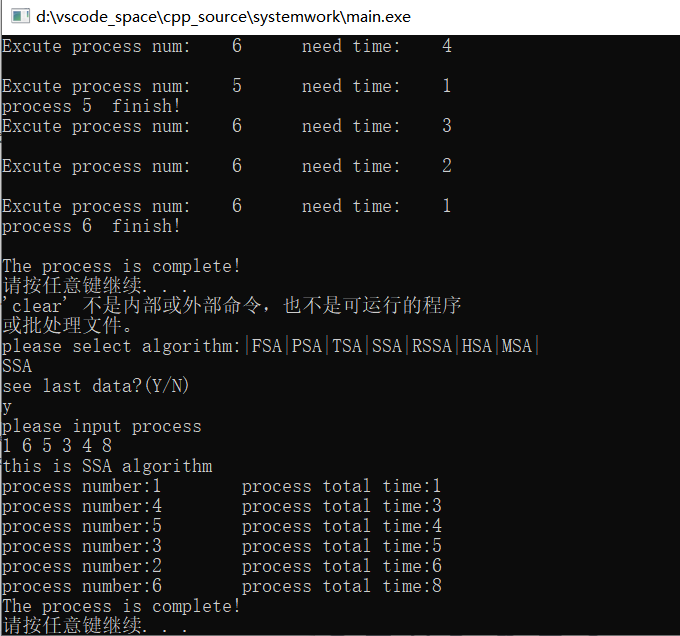


* 1. **时间片轮转调度算法**

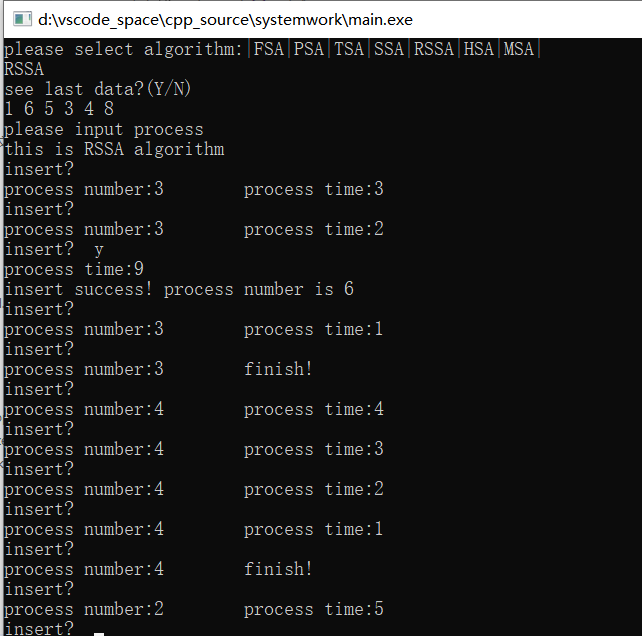


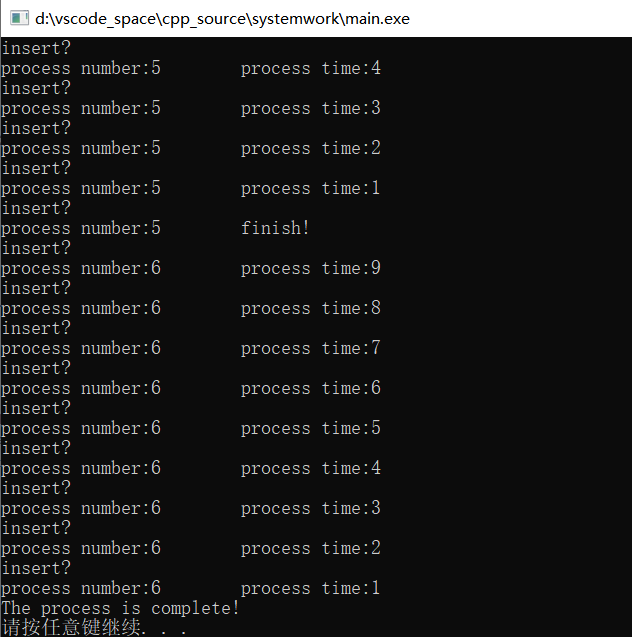


* 1. **短进程优先调度算法**

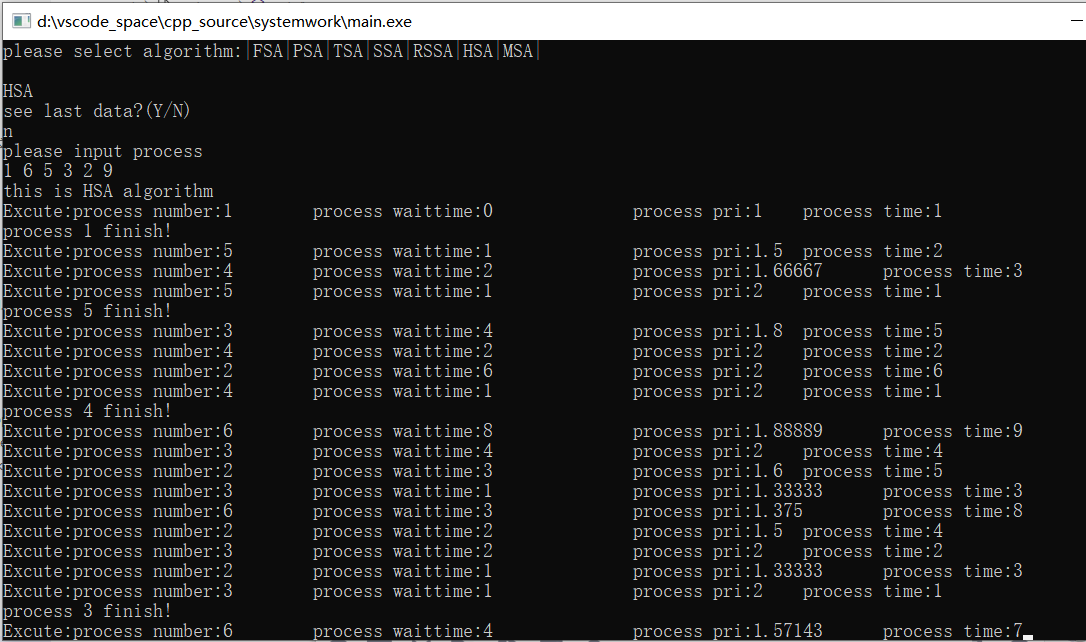


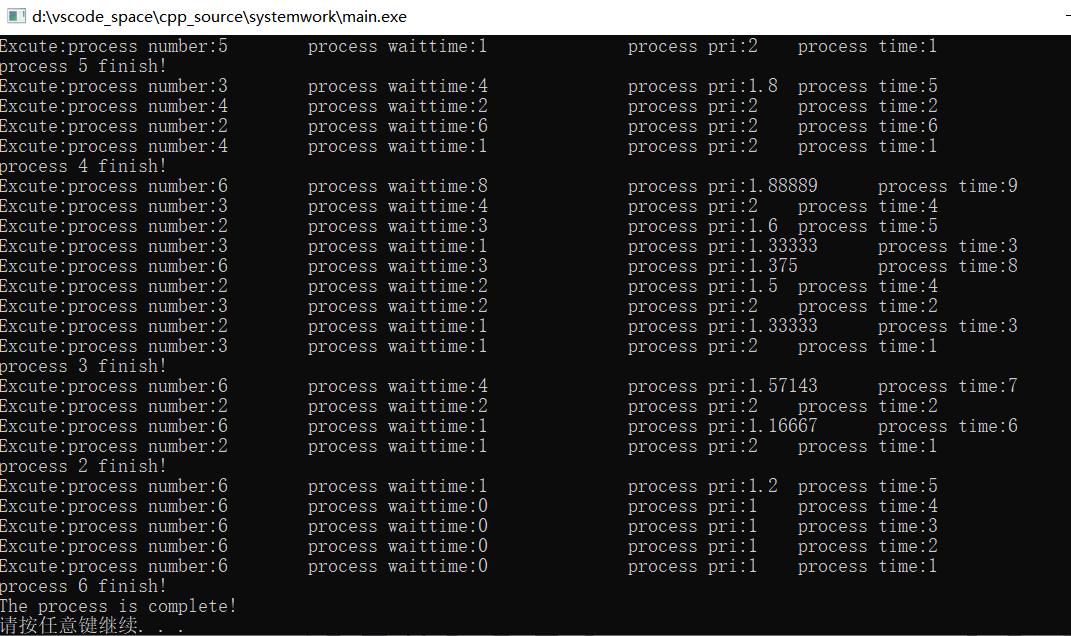
* 1. **最短剩余时间调度算法（可以实现添加进程）**



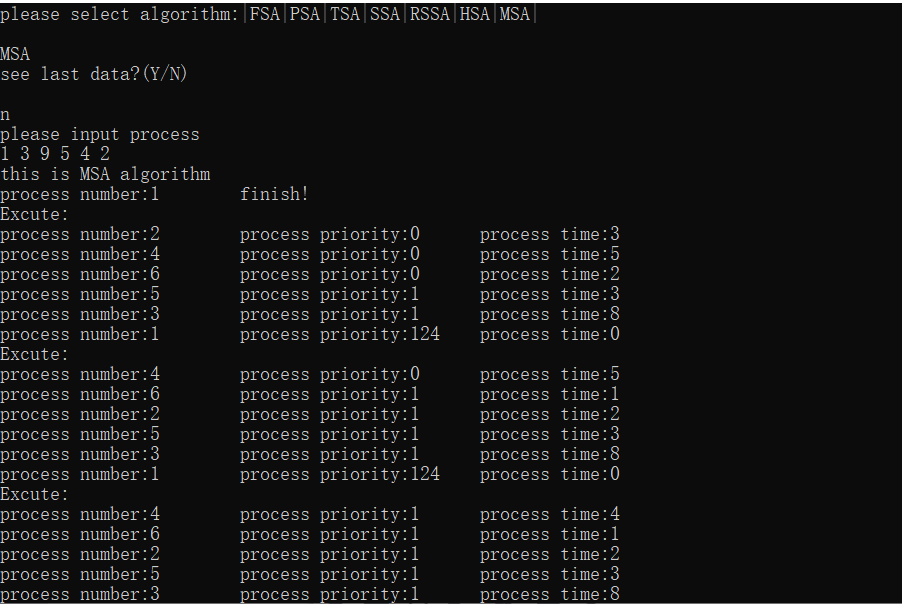


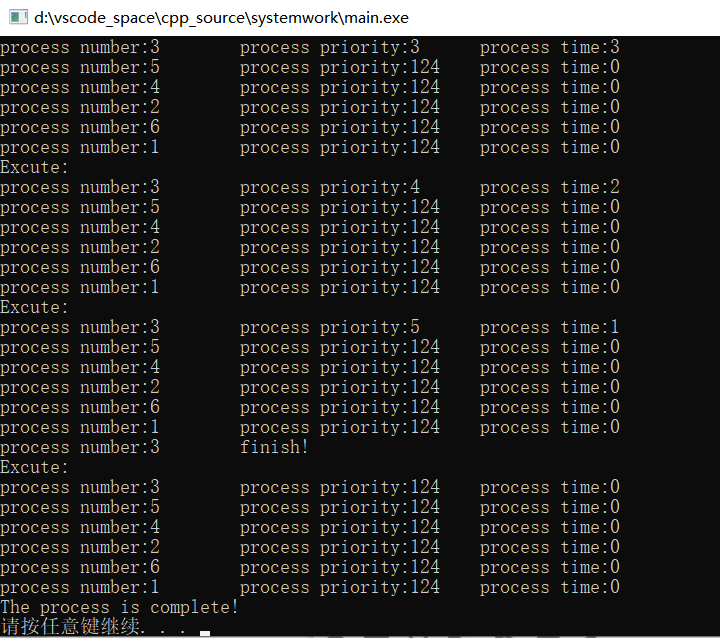
* 1. **最高响应比优先调度算法**



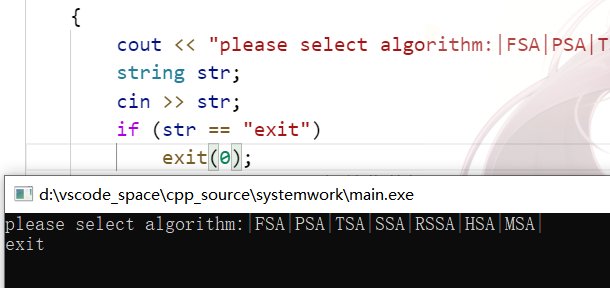


* 1. **多级反馈队列调度算法**





1. **退出（输入exit）**



# 课程设计心得体会

刘卉：了解了这两个算法的特点。通过编写高响应比算法，发现作业等待时候相同，有利于短作业；当响应比相同时，有利于剩余升级最短的。时间片轮转调度算法的特点，是最公平的调度算法。通过设计设想，了解本软件的大概框架。通过实验要求和需求分析，了解该程序的的具体要求与格式。通过画流程图，更加熟悉两个算法的流程。

刘家骏：通过此次课设，了解了短进程优先调度算法和先来先服务调度算法。在使用先来先服务调度算法时，每次调度是从就绪队列中选择一个最先进入该队列的进程，令其运行。此进程可用于作业调度，也可用于进程调度；适用于长作业，不适用于短作业。在使用短进程优先调度算法时，能够感受到相对于fcfs调度算法来说，其短作业所需的时间降低了。通过本次课设，更加清楚的了解了这两个算法调度的特点和流程。

肖定：了解了进程调度的特点，不同的算法对于不同的进程处理有所不同，部分算法只能对于静态的进程进行分析，但实际中往往会有进程的插入。对于多级调度算法中有许多不足，虽然考虑了cpu的处理，但是对于外部中断没有进行调度，如果外部中断，是否应该变换优先级。同时根据课题要求，加入了文件的写入，使得进程能够得到保存，但是写入是静态的，插入并没有写入到文件中。

通过这次课程设计，对于调度算法的实现细节有了更加完善的了解并进行了思考。

2021年12月25日

# 七.参考文献：

1.高等学校规划教材《操作系统（第四版）》电子工业出版社

2.《Java入门基础》清华大学出版社

# 八.附录

省略