# 进程控制和银行家算法的模拟实现

# 设计目的

进程是操作系统最重要的概念之一，进程和处理机管理的一个重要任务就是进程控制。通过对进程控制过程的模拟实现，掌握进程的创建、阻塞、唤醒、撤销等进程控制方法的实现过程，深入理解进程在执行过程中如何通过进程控制来完成进程各状态间的转换，从而达到多进程高效率并发执行和协调、实现资源共享的目的。由于多进程竞争共享资源，进程并发执行的顺序不当就会产生死锁，反而会降低系统的资源利用率，因此可以采用适当的资源分配算法来达到消除死锁的目的。通过对银行家算法的模拟实现，进一步理解并掌握死锁避免这种动态预防策略的基本原理和实现过程。

# 系统设计

## 需求分析

**模拟实现进程控制过程：**

每个进程都处于执行E(execute)、就绪R(ready)、等待W(wait)和完成F（Finish）状态之一，并假定初始状态均为R。如果被唤醒的进程的优先数大于现行的执行进程，则剥夺现行进程的执行权，而将CPU分配给被唤醒的进程。当系统处于死锁或所有进程都执行完毕时系统退出运行。

采用最高优先数优先的调度算法（即把处理机分配给优先数最高的进程）和先来先服务算法。每个进程有一个进程控制块（PCB）表示。进程控制块包含如下信息：进程名、优先数、到达时间、需要运行时间、已用CPU时间、进程状态，是否被分配资源，需求资源数。进程的运行时间以时间片为单位进行计算。就绪进程获得 CPU后都只能运行一个时间片。用已占用CPU时间加1来表示。如果运行一个时间片后，进程的已占用 CPU时间已达到所需要的运行时间，则撤消该进程，如果运行一个时间片后进程的已占用CPU时间还未达所需要的运行时间，也就是进程还需要继续运行，此时应将进程的优先数减1（即降低一级），然后把它插入就绪队列等待CPU。

**模拟实现死锁避免的银行家算法：**

设置相应的数据结构并实现算法，模拟出银行家算法执行检测过程并显示（初始系统可用资源数，各进程已占用资源数、还需资源数；能否进行试分配；试分配后系统可用资源数，各进程已占用资源数、还需资源数的变化情况；检测结果是否存在安全序列及安全序列情况等

**可视化界面：**

为系统的算法提供一个友好的可视化界面，为进程调度和银行家算法分别设置一个可查看的界面，同时允许用户通过输入填入自己想要的数据，显示资源分配表和进程状态表，包括每个进程的已分配资源、最大需求量、需求资源量等信息。提供清晰的信息提示，指导用户正确地操作和解读系统状态。

## 系统结构

**进程信息模块：**

内部包含每个进程创建应该包含的信息，进程名，到达时间，需要运行时间  
**，**优先级，已用CPU时间，需求资源，状态，是否分配资源，并做好每创建一个进程时的初始化工作

**进程管理模块：**

负责管理系统中的进程，包括进程的创建、销毁、状态转换等操作。维护每个进程的优先级信息。

**调度器模块：**

根据进程的优先级算法来确定下一个执行的进程，实现优先级调度算法的逻辑，决定进程的执行顺序。

**进程状态转换管理：**

负责管理进程状态之间的转换，包括进程的创建、就绪、运行、阻塞和终止等状态的转换。

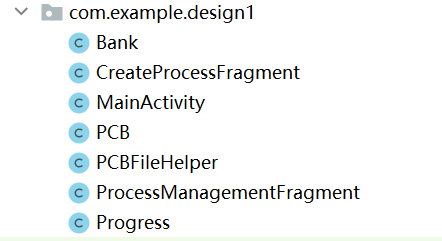
确保进程状态的变化符合调度算法的规则，并且能够及时更新进程的状态信息。

**银行家算法模块：**

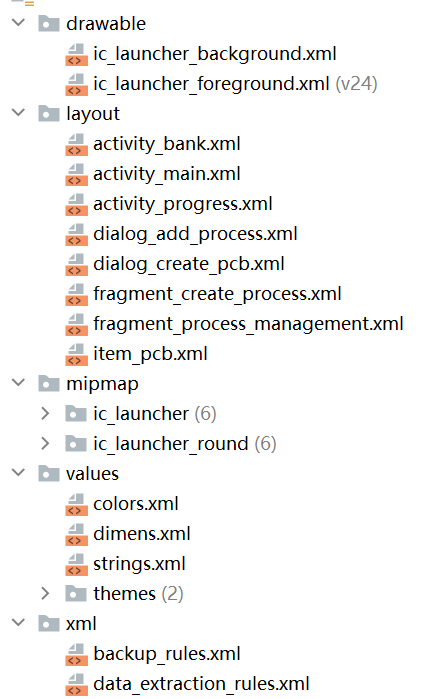
模拟银行家算法的实现，找出是否存在安全序列。

本项目采用Android界面编写

相关类：

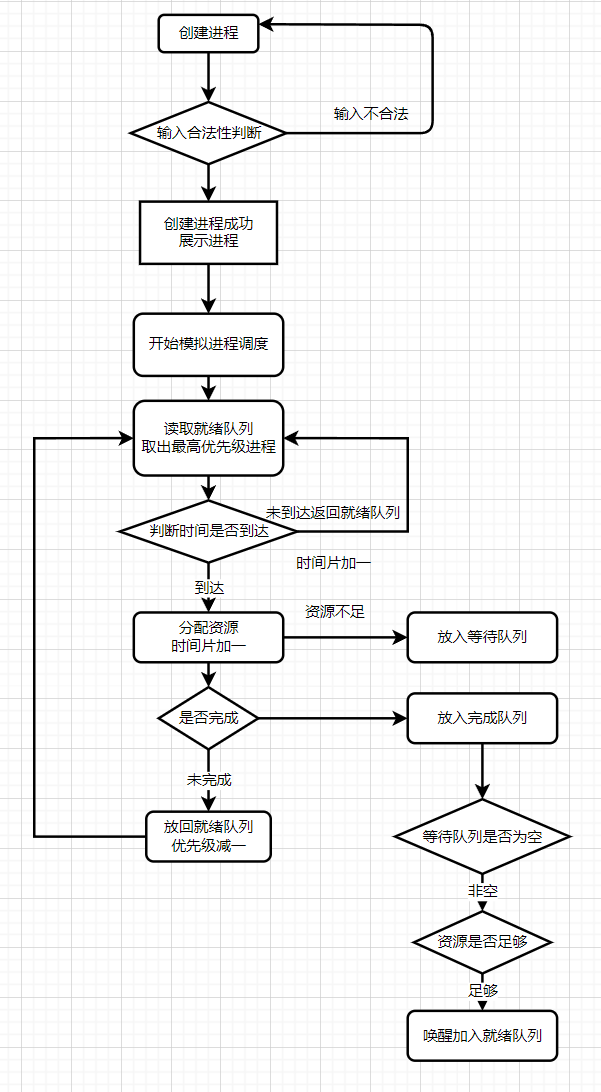


界面布局文件

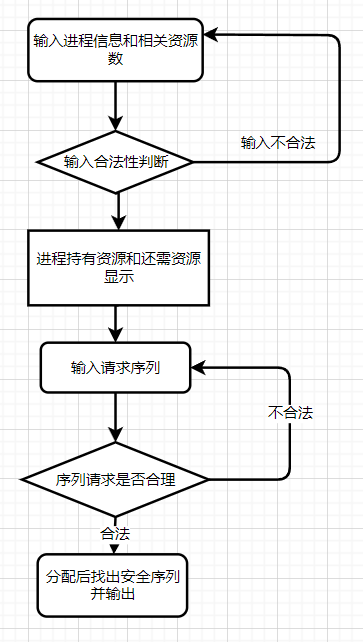


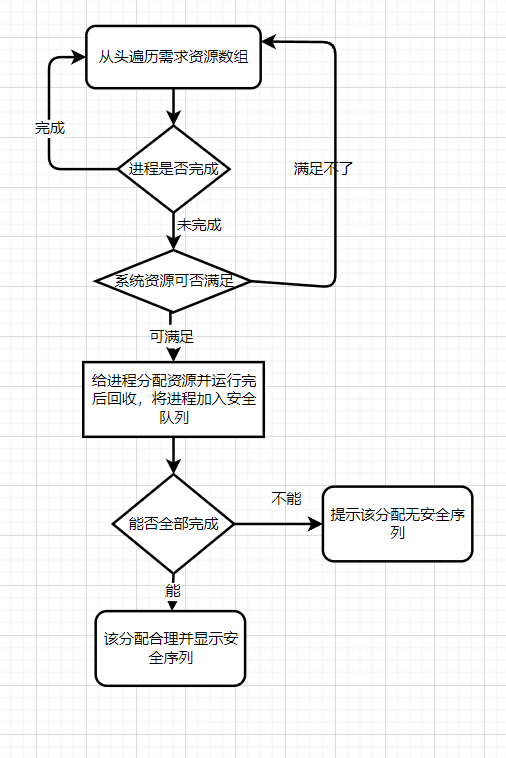
## 主要功能流程

### 3.1进程管理



### 3.2银行家算法





# 系统实现

## 进程管理

### 1.1进程类：

用来存储进程对象的内部信息



### 1.2进程资源存取辅助类



### 1.3调度算法实现模块

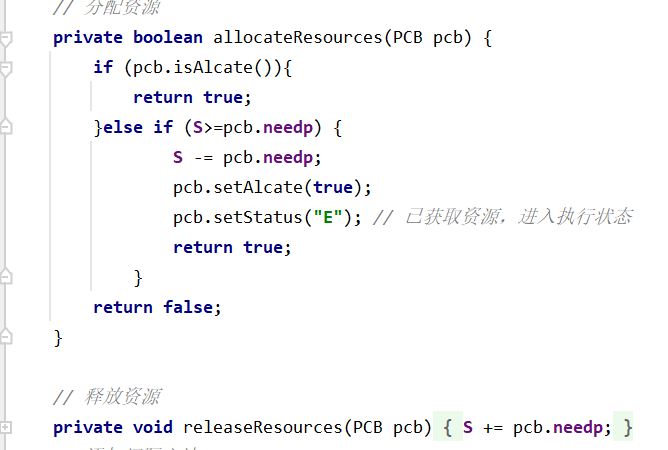


优先级的短作业优先（PSA）调度算法的模拟操作系统进程调度方法。使用了优先级队列来管理就绪队列、等待队列和完成队列。在每次执行任务时，会检查当前时间，处理就绪队列中的进程，并根据进程的状态进行相应的操作，包括资源分配、时间片执行、状态更新等。在处理就绪队列中的进程时，会首先判断当前进程是否到达执行时间，若未到达，则将其重新加入就绪队列，否则进行资源分配和时间片执行。如果进程执行完毕，将其放入完成队列；如果未完成，降低其优先级并将其重新加入就绪队列。此外，它还会尝试唤醒等待队列中的进程，并根据可用资源情况进行资源分配。该方法通过定时任务循环执行，模拟了操作系统进程调度的过程，并在界面上实时显示进程的状态。最后，当所有进程执行完毕时，它会在界面上添加一个“结束模拟”按钮，点击该按钮会删除 PCB 数据文件。

### 1.4资源管理模块

模拟实际进程中部分申请资源的情况。

每次进程请求资源时请求部分资源，部分资源由随机数模拟，当进程因部分资源不足而阻塞时，请求的部分资源不变。



如果申请的资源数可满足，则更新进程资源。

如果不能满足，进入等待态。

### 1.5进程显示更新模块

每次进程调度后更新显示界面



## 2.银行家算法

### 2.1进程信息存储模块



用于存储检测银行家算法的进程的信息

### 2.2初始化模块



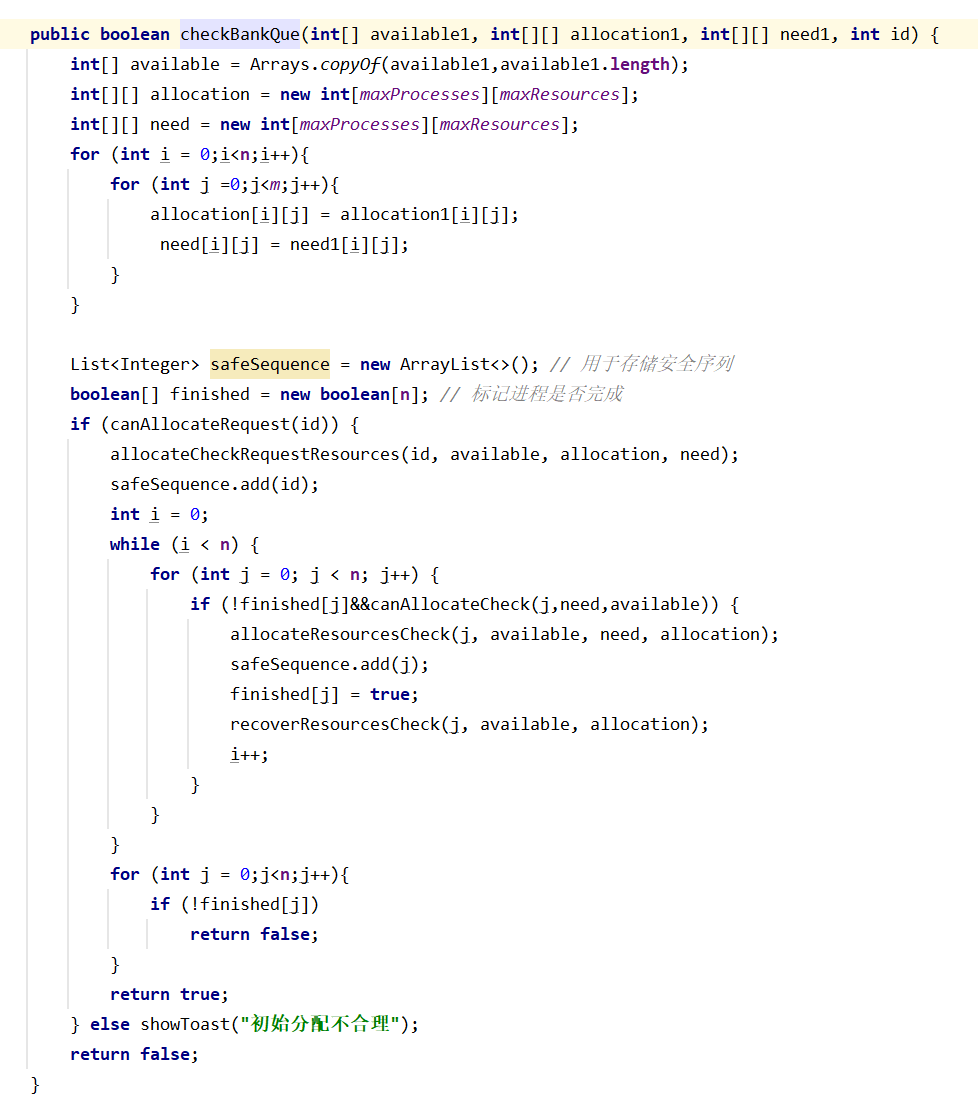
初始化界面和相关组件，检查数据合法性并添加监听器。

### 2.3显示更新模块



将更新的矩阵信息重新显示在表格里。

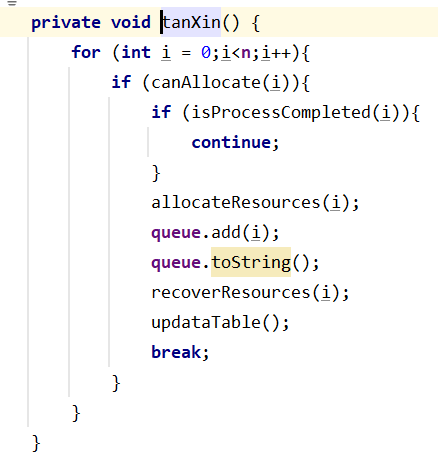
### 2.4安全状态检测和执行模块



银行家算法的资源请求检查函数。接受四个参数：available1 表示当前可用资源数量的数组，allocation1 表示当前分配给各进程的资源数量二维数组，need1 表示各进程尚需资源数量的二维数组，id 表示请求资源的进程编号。

首先，代码复制了输入的资源分配情况，以防止修改原始数据。然后，创建了一个空的安全序列列表和一个布尔数组来标记各进程是否已完成。接着，检查请求是否合理，如果合理，则分配请求资源给进程，并将该进程添加到安全序列中。然后，进入一个循环，尝试分配资源给未完成的进程，直到所有进程都完成为止。在循环中，检查未完成的进程是否可以安全地分配资源，如果可以，则分配资源并标记该进程为已完成。最后，检查是否所有进程都已完成，如果是，则返回 true，否则返回 false。如果请求不合理，则显示一个提示消息并返回 false。

当检测函数返回true时说明该请求有安全序列允许执行，且当执行到最后一步后显示安全队列



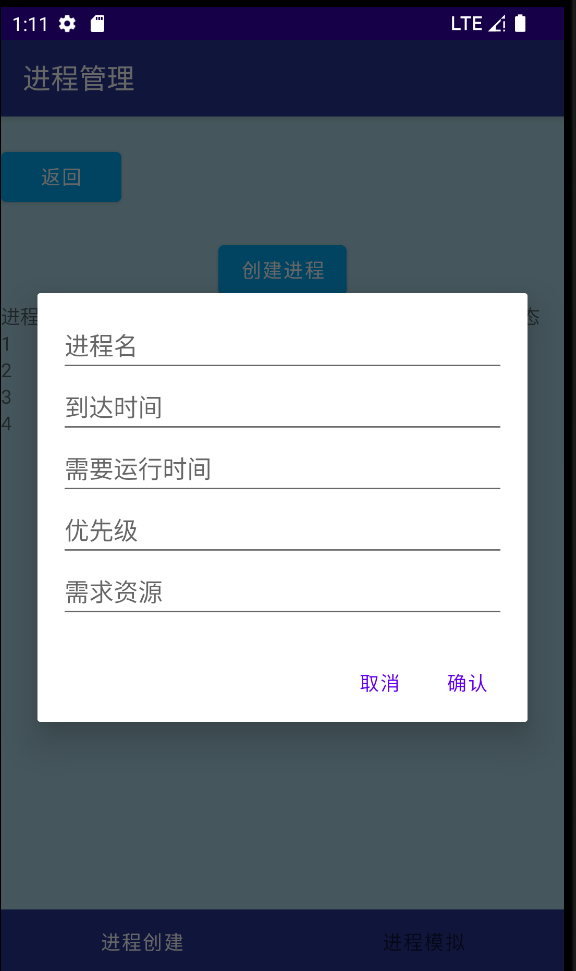
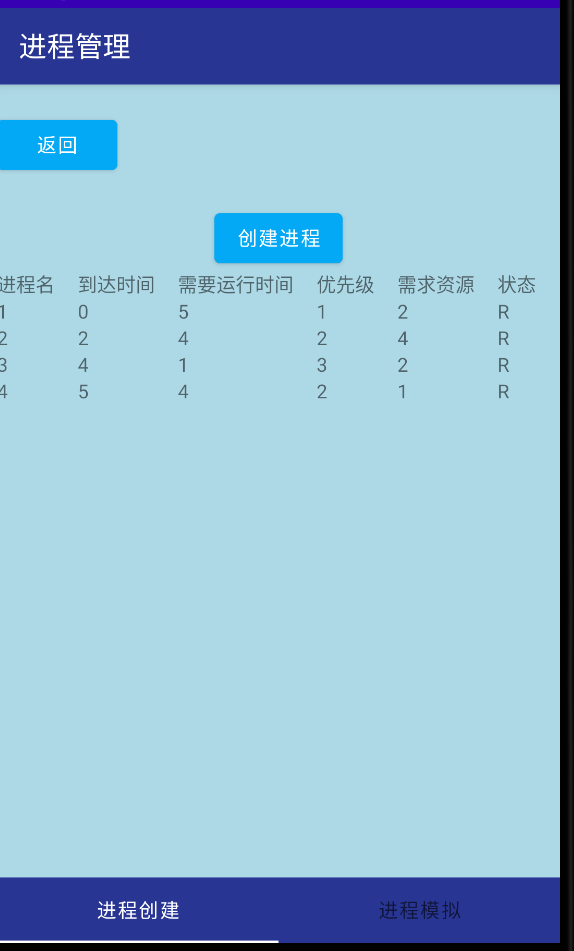
# 系统运行

**总界面：**

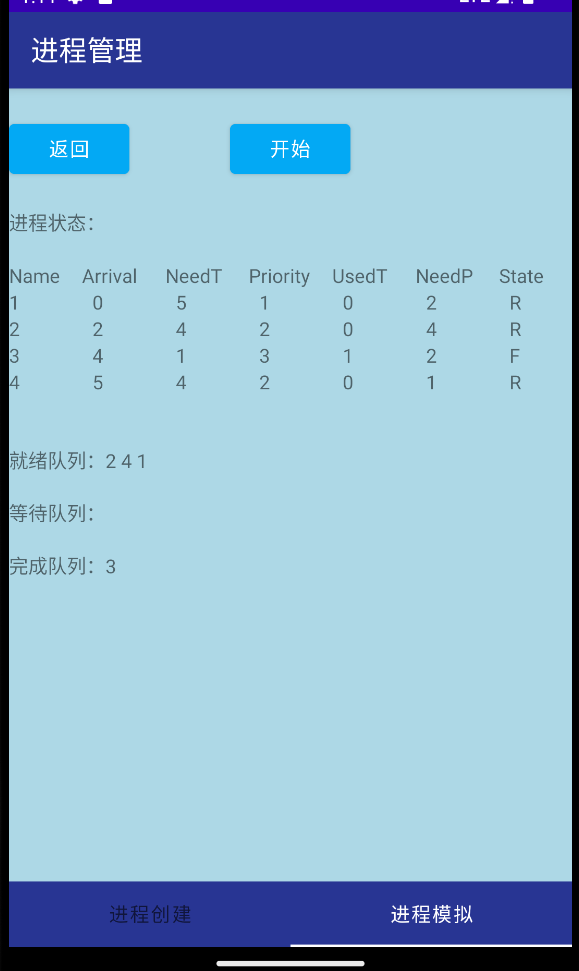
****

## 1.进程管理

### 1.1进程创建



### 1.2开始模拟



### 1.3.执行完毕



## 2.银行家算法

### 2.1主界面



### 2.2 逐步分配



# 总结

进程控制模拟实现：

通过设计与实现优先级调度模拟，我深入理解了进程调度的原理和方法，以及操作系统中进程管理的重要性。通过模拟进程的执行过程，我对进程调度算法的效果和性能有了更直观的认识，并对不同调度算法的优缺点有了更深入的理解。

通过设计与实现优先级调度模拟，我学到了如何设计和实现一个完整的模拟系统，并加深了对操作系统原理和进程调度算法的理解。

经验与体会：

通过设计与实现优先级调度模拟，我体会到了模拟系统设计的复杂性和挑战性。在设计过程中，需要考虑到各种情况和可能性，并设计合理的解决方案。同时，也学会了如何评估模拟系统的性能，并对系统进行优化和改进。

银行家算法的模拟实现：

设计与实现银行家算法需要考虑多方面的因素，包括系统资源管理、进程调度和安全性保障等。在设计过程中，需要对系统的资源状态进行实时监控，并根据进程的请求情况进行动态调整。同时，需要考虑到并发执行的情况，以及可能出现的竞态条件和死锁问题。

通过实现银行家算法，不仅加深了对操作系统资源管理原理的理解，还提高了对并发编程和死锁处理的技能。同时，也体会到了设计算法时需要考虑多种情况和可能性，并寻找合适的解决方案的重要性。

经验与体会：

通过设计与实现银行家算法，我深刻体会到了算法设计的复杂性和挑战性。在解决问题的过程中，需要不断地思考和尝试，寻找最优的解决方案。同时，也学会了如何分析和评估算法的性能，以及如何对算法进行优化和改进。

总的来说，银行家算法的设计与实现过程和进程调度模拟实现是一次宝贵的经验积累，让我对操作系统原理有了更深入的理解，也提高了自己的编程能力和解决问题的能力。