

**操作系统进程管理项目**

题目 电梯调度系统

姓 名 张翔

学 号 2352985

学 院 计算机科学与技术学院

专 业 软件工程

教 师 张惠娟

二〇二五 年 五 月 十七 日

电梯调度系统

张翔 2352985 软件工程

**一、项目介绍**

（一）项目需求

某一层楼20层，有五部互联的电梯。基于线程思想，编写一个电梯调度程序。具体需求包括：

（1）用户界面：每部电梯都配有一套按键，包括数字键（1-20层），开门键、关门键、上行键、下行键以及报警键。每层楼以及每部电梯门口装有上行和下行按钮，以及用于显示电梯当前状态的数码显示器。

（2）调度算法：所有电梯在初始状态时均位于第一层。电梯在没有接收到请求时应保持当前位置静止。调度算法需要优化电梯的响应时间和运行效率，将任务合理分配给当前最合适执行该任务的电梯，以提高运行效率。

（3）电梯相互连结：五部电梯相互联结，即当一个电梯按钮按下去时，其它电梯相应按钮同时点亮，表示也按下去了。

（二）项目目的

（1）操作系统调度模拟： 本项目通过调度多台电梯来模拟操作系统中的任务调度机制。电梯的运行过程与操作系统中的进程调度具有高度相似性，每部电梯都可以视为一个进程，都需要在多个任务之间合理分配有限资源。通过模拟电梯的请求、响应和运动，可以直观地理解操作系统调度器的设计思路和工作原理。

（2）学习多线程编程方法：本项目需要实现多部电梯的并发运行，因此在编程过程中需要使用多线程方法，多线程编程是操作系统广泛使用的技术，理解和掌握这一技术可以更好地理解现代操作系统的并发和并行处理能力。还需要解决用户界面线程与工作线程之间的通信问题，确保界面响应的实时性和流畅性。

（3）学习调度算法：调度算法为本项目的核心内容，通过系统性地研究和实现多种经典的调度算法，可以提供关于如何处理多个请求并优化资源分配的直观理解。

（三）开发环境

操作系统：Windows 11

开发工具：JetBrains Rider 2024.3.5

编程语言：C# (.NET 8.0)

UI框架：WPF (Windows Presentation Foundation)

**二、项目设计**

（一）Elevator类（电梯类）

这个 Elevator 类是电梯调度系统中的核心模型类，继承自ObservableObject以实现数据绑定和属性变更通知。它完整封装了电梯的运行时状态和行为控制，主要包含以下功能：

（1）状态管理：使用 \_currentFloor 记录电梯的实时位置，通过 \_state 枚举维护运行状态（上行/下行/空闲等），采用线程安全的 ObservableCollection<int> 存储待处理的楼层请求队列。

（2）线程控制：通过 CancellationTokenSource 实现异步任务的取消机制，使用 \_queueLock 对象锁确保队列操作的线程安全，\_isProcessingQueue 标志位防止重复处理任务队列。

（3）命令接口：提供完整的操作命令集：AddFloorCommand（楼层调度）、OpenCommand（开门）、CloseCommand（关门）和 EmergencyCommand（紧急暂停），采用 RelayCommand 实现 MVVM 模式的命令绑定。



（二）ElevatorSystem类（电梯系统类）

ElevatorSystem 类是电梯调度系统的核心控制类，继承自 ObservableObject 实现数据绑定和属性变更通知，负责管理多部电梯的协同调度。其主要功能包括：

（1）电梯集群管理：维护 \_elevators 数组管理5部电梯实例，提供 Elevators 只读属性对外暴露电梯状态，通过事件监听实时跟踪各电梯的楼层变化（PropertyChanged）。

（2）智能任务调度：采用 FindBestElevator 算法实现动态调度。

（3）任务生命周期管理：使用 ObservableCollection<ElevatorTask> 维护任务队列，自动检测任务完成状态（CheckTaskCompletion），防止重复请求（canDispatch 校验）。

该类通过精细的线程安全设计和响应式编程模式，实现了高效的多电梯协同调度系统，调度算法在保证响应速度的同时优化了整体效率。



（三）MainWindow类（主要窗口类）

这是用户交互的界面, 其中初始化了一个ElevatorSystem类的实例，并将该实例作为整个窗口的DataContext（上下文），方便在前端调用该实例中的各自方法。它可以展示电梯的当前状态，接收用户的输入（如电梯楼层按钮的点击），并显示电梯的实时运动。



（四）其他工具类

以上的类要么注重于界面展示，要么注重于系统逻辑。项目中还需要其他一些工具类来实现前后端的绑定，以及界面的实时更新。

Converter类：转换器类用于在数据绑定过程中对数据进行转换，使界面显示的数据格式与后端逻辑数据分离。具体使用包括：将电梯的ElevatorState（如MovingUp、Idle）转换成对应的颜色或文字；将楼层数字转换成更友好的显示格式（如1 → "1F"）。

ICommand类：命令类用于在MVVM架构中处理用户交互（如按钮点击），使界面事件与业务逻辑解耦。在可视化UI界面中的所有按钮均绑定了一个Command，用来作为点击按钮后触发的事件。具体使用包括：处理楼层按钮点击，触发电梯调度逻辑；控制电梯的开门、关门操作模拟紧急停止功能。

ObservableObject类：可观察对象类实现INotifyPropertyChanged接口，使数据变化能自动通知UI更新。在本项目中的具体使用场景为：当电梯的CurrentFloor变化时，UI上的电梯位置实时移动；当电梯状态变化时，UI动画自动触发，实现显示界面的流畅自然。

**三、用户交互界面设计**

（一）界面设计概述

整体界面设计区块包括：电梯运行区、电梯运行区、紧急暂停、开门关门按钮区域、上下行按钮区。在整体设计上遵循以下理念：

（1）清晰的布局设计：功能区块划分明确，间距合理，重要操作区域位于界面黄金位置，层级分明；

（2）UI一致性与实时性：统一设计视觉风格，所有交互元素提供即时视觉反馈，能够实时更新电梯位置和运行数据；

（3）直观的楼层导航：每部电梯旁边的楼层按钮清晰标注，配有上下行箭头按钮，使用户可以轻松设置电梯的目的楼层。减少了操作的复杂性，提升了界面的用户友好度。

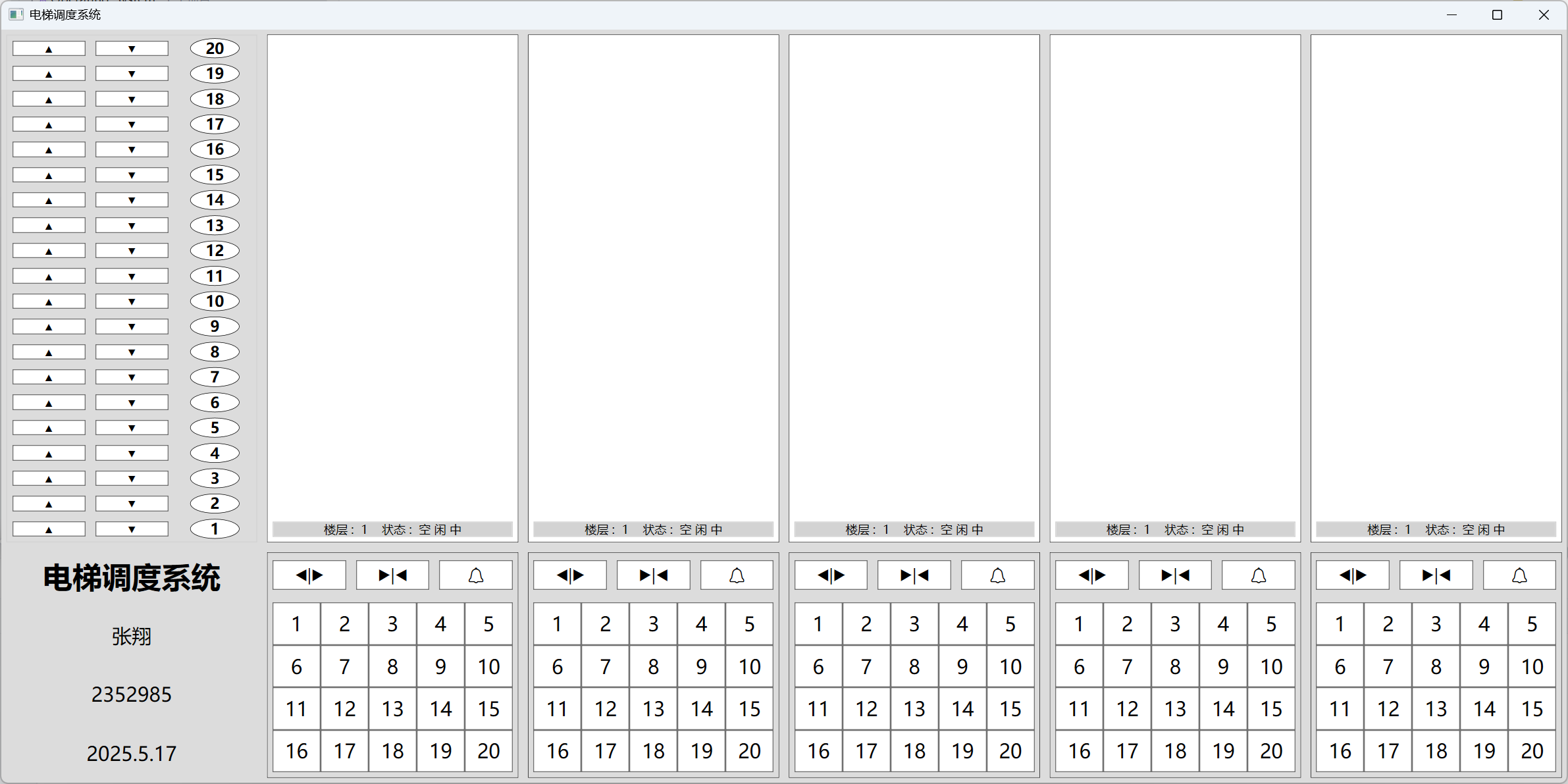


图1.设计界面示例图

（二）界面设计亮点

电梯运行区：每个电梯均被分配了一个可视化的电梯井用来观察当前电梯的运行情况，电梯运行位置与左侧楼层上下行按钮处一一对应，可以清晰观察到目前电梯所在楼层，此外电梯内部也会显示电梯的具体参数，包括所在楼层和运行状态（上行/下行/空闲等等），此外不同状态还对应电梯的不同显示颜色（空闲对应浅灰色，上行对应浅蓝色，下行对应浅黄色），方便清晰的观察电梯状态的变化。

电梯运行区：每台电梯都配有楼层按钮来选择想要到达的楼层。当点击对应按钮时，按钮会变为不可用状态。当到达指定楼层（即当前任务完成）时按钮状态会恢复。

紧急暂停、开门关门按钮区域：每台电梯均提供紧急按钮，以便在电梯出现故障时进行模拟和处理，紧急暂停后该电梯不再能够执行命令。开关门可以延长或减少已经停靠在某楼层等待中的时间（不能在电梯上下行时按开关门按钮）

上下行按钮区：除起始层和顶层分别只有上行键和下行键，每个楼层都配有上下行键。用户可以通过点击这些按钮发出上行和下行请求来调用合适的电梯响应。

**四、核心调度算法介绍**

（一）算法流程

本项目第一部分调度算法的核心流程是通过计算每部电梯的综合得分（考虑运行方向、当前位置、任务队列等因素）来选择最优电梯并分配任务。

具体流程为：遍历五台电梯，跳过当前状态为故障的电梯，然后根据电梯当前状态（上行、下行或空闲）计算分数：同方向且在路径上的电梯计算其位置和目标楼层差值的绝对值，同方向但需折返的计算目标楼层和任务列表中最远位置的绝对差值（考虑折返带来的），状态为空闲的电梯计算其位置和目标楼层差值的绝对值，反方向电梯惩罚性加分。

然后根据当前电梯任务列表的长度作为负载因子，防止任务分配过于不均衡。全部计算完成后，选择总分最低的电梯响应请求。

本项目中的调度算法除了电梯系统在接收到任务然后分配给合适的电梯外，还包括每台电梯在面对任务列表中被分配的多项任务时的执行顺序，电梯的默认逻辑是会依次按顺序执行任务列表中的每项任务，然后在执行完后将其删除。

然而在每一次添加新任务时，会执行一次任务排序算法，用来更新任务执行顺序。首先设置标志位isAdded初始为false，用于判断新楼层是否已插入队列，遍历当前电梯的楼层队列\_floorQueue中的每个目标楼层，然后根据电梯运行状态执行不同逻辑：

（1）当电梯处于上行状态（MovingUp）时：

a) 如果队列中的目标楼层在当前楼层下方：检查新楼层是否大于该目标楼层，是则插入当前位置。

b) 如果队列中的目标楼层在当前楼层上方：检查新楼层是否小于该目标楼层且大于当前楼层，是则插入当前位置。

（2）当电梯处于下行状态（MovingDown）时：

a) 如果队列中的目标楼层在当前楼层上方：检查新楼层是否小于该目标楼层，是则插入当前位置。

b) 如果队列中的目标楼层在当前楼层下方：检查新楼层是否大于该目标楼层且小于当前楼层，是则插入当前位置。

如果遍历完整个队列仍未找到合适位置，则将新楼层添加到队列末尾。

这两部分调度算法的共同作用实现了五台电梯互不干扰的流畅运行。

（二）算法优势

（1）方向匹配时优先路径上的最近电梯，减少等待时间；

（2）空闲电梯灵活调度，提高资源利用率；

（3）任务队列加权避免单一电梯过载，实现负载均衡；

**五、项目总结**

（一）多线程编程

本项目通过电梯调度系统的开发，深入实践了C#多线程编程技术。系统采用Task Parallel Library(TPL)实现多部电梯的并发运行，每部电梯作为独立线程运行，模拟真实场景中的并行调度。

在开发过程中，重点解决了线程同步问题，通过lock关键字确保共享资源的安全访问，避免出现竞态条件。同时，运用async/await异步编程模型处理IO密集型操作，确保UI线程不被阻塞。项目还实现了基于事件的线程间通信机制，使各电梯线程能协同响应楼层请求，这些实践加深了对线程生命周期管理、线程池等核心概念的理解。

（二）调度算法学习

在开发本项目的过程中我系统性地学习并尝试实现了多种电梯调度算法，包括基础的FCFS（先来先服务）、高效的SSTF（最短寻道时间优先）以及更复杂的SCAN和LOOK算法。加深了对操作系统中常用调度算法的理解和认识。

（三）线程管理

本项目通过精心设计的进程管理策略，实现了复杂的业务需求，即多电梯的调度和管理。进程管理主要涉及以下几个方面：

控制系统进程根据各电梯的当前状态和楼层位置，动态地为电梯分配上行或下行的任务。这种动态调度策略提高了电梯运行的效率，并减少了等待时间。电梯进程负责实时更新其状态，如当前楼层和电梯门的状态，以及响应控制系统进程的调度命令。

**六、项目展示**

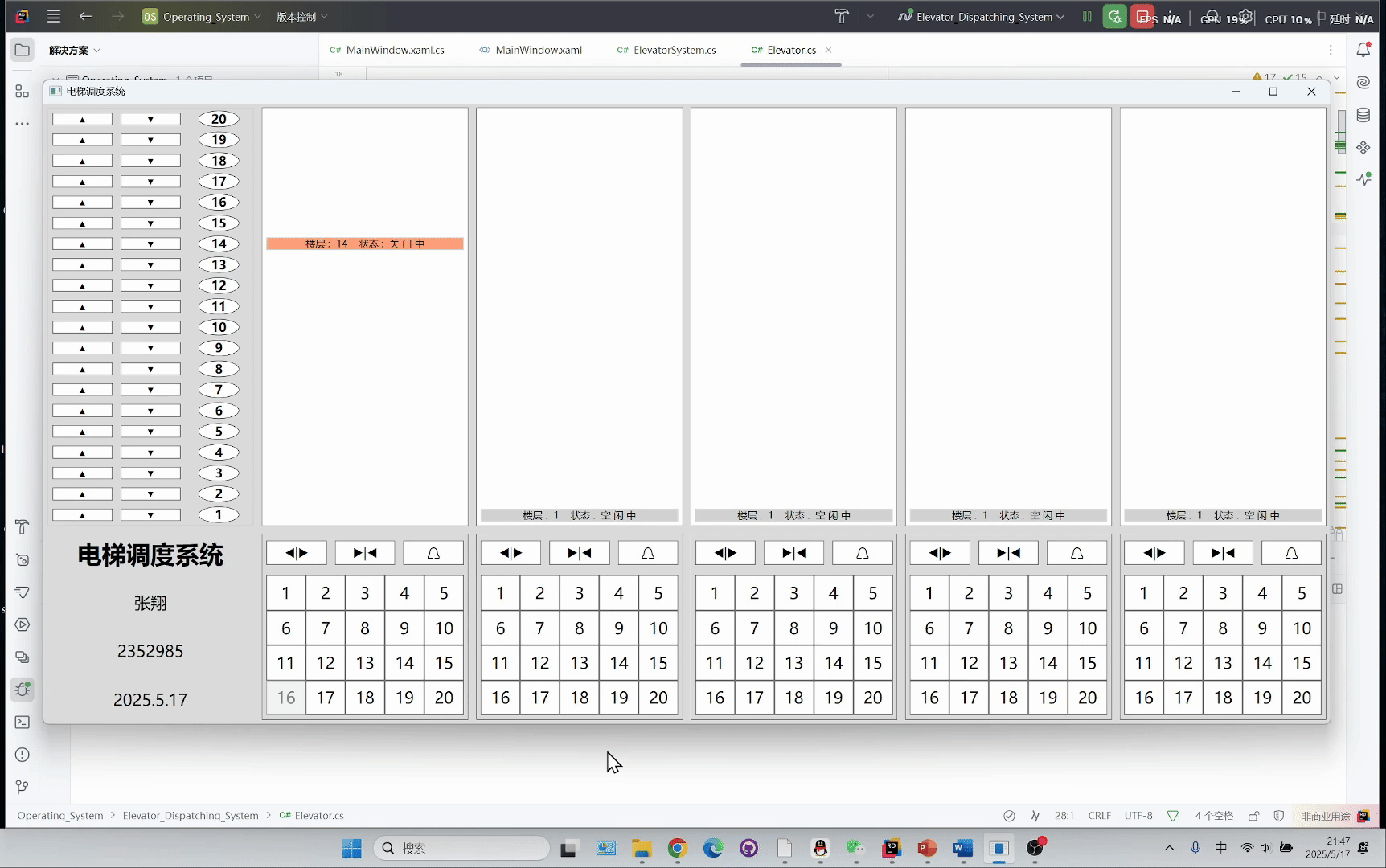


图2.单部电梯运行示例

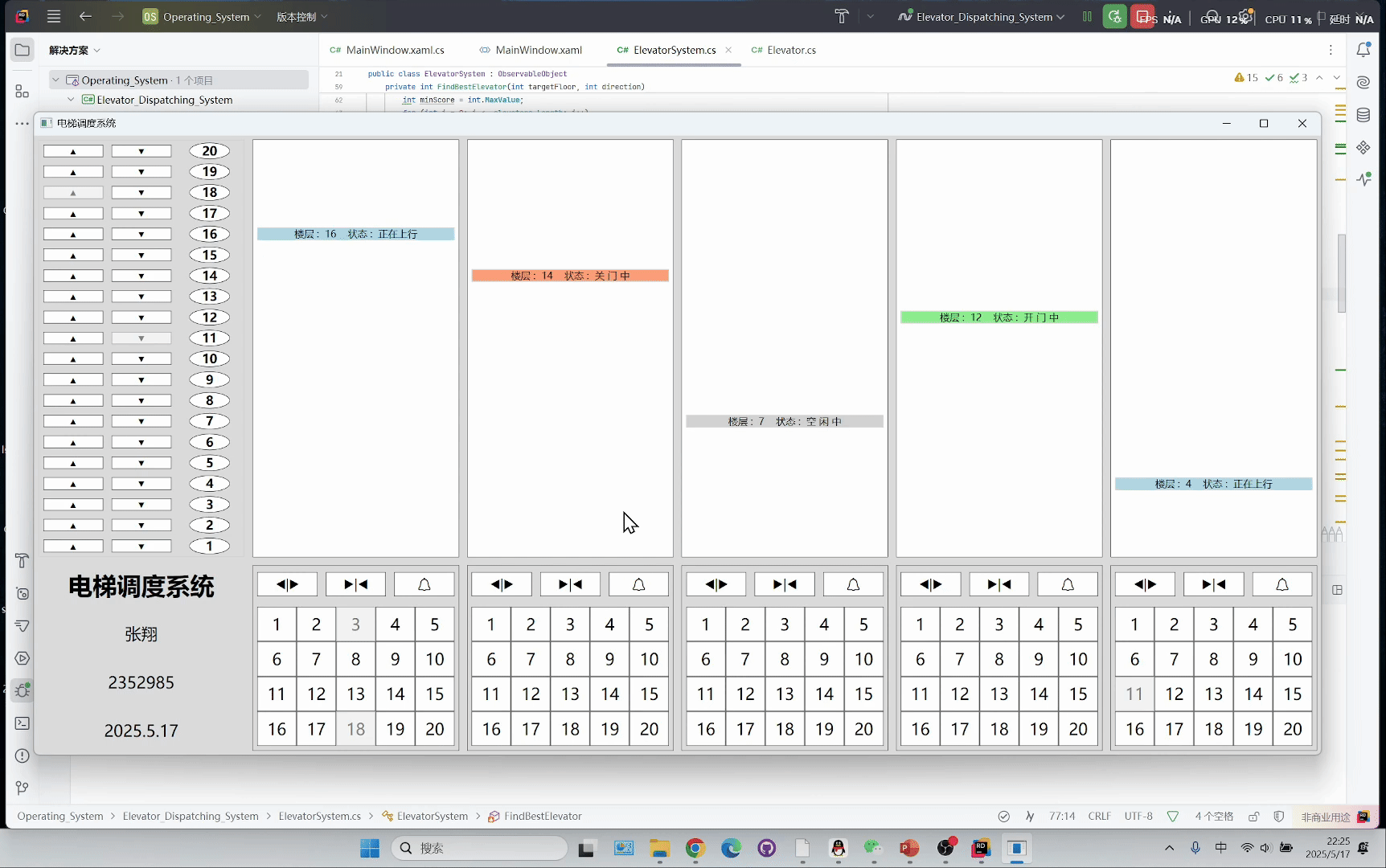


图3.多部电梯调度运行示例