# **电梯调度程序设计项目报告**

2351289周慧星

## **一、项目概述**

### **（一）项目介绍**

本项目致力于模拟一个具备 20 层楼、五部电梯的运行环境。通过精心设计电梯调度算法，构建一个高效的电梯调度系统，以此满足用户多样化的乘梯需求。该系统允许用户自定义建筑物的楼层数量和电梯数量，同时提供了多种电梯调度算法，如 LOOK 算法、最短剩余时间优先算法、分组调度算法等，用户可以在各楼层请求电梯，系统会根据所选的调度算法自动调度合适的电梯响应请求。系统通过直观的用户界面实时显示电梯的运行状态和当前所在楼层，方便用户观察和操作。

### **（二）项目设计**

1. 功能设计

* 用户能够自定义建筑物的楼层数量和电梯数量。
* 提供多种电梯调度算法供用户选择。
* 实时显示电梯的运行状态（上升、下降、暂停）和当前所在楼层。
* 用户可以在各楼层发出电梯请求，系统能自动调度电梯响应。
* 具备运行和停止功能，方便控制整个系统。

2. 非功能设计（界面设计）

* 系统界面友好，操作简单易懂。
* 系统具有较高的稳定性和可靠性，能长时间稳定运行。

## **二、系统设计**

### **（一）整体架构**

系统主要由电梯、楼层和调度算法三大部分构成。电梯部分负责处理电梯内部的各类操作，诸如开关门、报警以及到达楼层等；楼层部分主要承担接收用户请求，并将这些请求准确发送给调度算法的任务；调度算法则依据不同的策略，精准选择合适的电梯来响应请求。

### **（二）模块设计**

1. **电梯模块**：涵盖电梯的基本属性，如编号、当前楼层、运行状态、门状态等，同时具备电梯的操作方法，像开门、关门、移动楼层、接收请求等。电梯内部配备数字键、开门键、关门键、报警键等，用户可通过这些按钮操控电梯的运行并改变其状态。此外，还设有数码显示器，用于实时显示当前楼层以及电梯的运行状态。
2. **楼层模块**：负责记录楼层信息，包括楼层数。每层楼均设有上行按钮、下行按钮和数码显示器。当用户按下楼层按钮时，楼层模块会迅速将请求发送给调度算法。
3. **调度算法模块**：依据不同的调度算法，如 LOOK 调度算法、最短剩余时间调度算法、分组调度算法、FCFS 算法 + 评分最优、FCFS 算法 + 任务最少、FCFS 算法 + 顺序分配等，对楼层的请求进行有效处理，进而选择合适的电梯来响应请求。

### **（三）类设计**

1. **qt\_elevator\_os类**：作为主窗口类，负责系统的初始化和用户界面的显示。处理运行和停止按钮的点击事件，管理电梯和建筑物对象的创建与销毁。
2. **elevator类**：表示单个电梯，处理电梯的运行逻辑，包括开门、关门、移动到目标楼层等。提供接收外部请求和计算剩余任务时间的功能。
3. **building类**：表示建筑物，包含多个电梯对象。负责处理电梯的调度算法，根据用户选择的调度模式为用户请求分配合适的电梯。

## **三、功能设计**

### **（一）电梯功能**

1. **按钮操作**：用户可借助电梯内部的按钮自如控制电梯运行。按下数字键能够选择目标楼层；按下开门键，在电梯静止且门关闭或正在关闭时，可打开电梯门；按下关门键，可关闭电梯门（前提是门未关闭或正在关闭）；按下报警键则可发出警报。
2. **状态显示**：电梯内部的数码显示器能够实时显示当前楼层，同时通过不同图标直观展示电梯的运行状态（如上升、下降、停止）以及门状态（如已开门、已关门、关门中、开门中）。
3. **请求处理**：电梯能够接收来自楼层的请求，并依据请求灵活调整运行方向和目标楼层。当电梯抵达目标楼层时，会暂停并执行相应操作，如开门、关门等。

### **（二）楼层功能**

1. **请求发送**：当用户在楼层按下上行或下行按钮时，楼层模块会即刻将请求信息（包括请求方向和楼层）发送给调度算法。
2. **状态显示**：楼层的数码显示器会显示当前电梯的运行状态和所在楼层，便于用户及时了解电梯位置。

### **（三）调度算法功能（6大算法）**

1. **LOOK 调度算法**：该算法优先选择与请求楼层距离最近且运行方向与请求方向一致的电梯。在LOOKAlgorithm函数中，初始化最近电梯索引closestElevatorIndex为 -1，最小距离minDistance为最大整数值INT\_MAX。遍历eles电梯向量，对于每个电梯：若电梯处于空闲状态（ele->status == 0），计算其与请求楼层的距离，若小于当前最小距离，则更新最小距离和最近电梯索引；若电梯正在运行，且运行方向与请求方向一致（向上请求且电梯向上运行且当前楼层小于等于请求楼层，或向下请求且电梯向下运行且当前楼层大于等于请求楼层），同样计算距离并更新最小距离和最近电梯索引。找到合适电梯后，向其发送请求并显示调度信息；若未找到，则强制调度第一部电梯。

**// LOOK调度算法实现函数**

**void building::LOOKAlgorithm(bool up, int floor) {**

**int closestElevatorIndex = -1;**

**int minDistance = INT\_MAX;**

**for (int i = 0; i < ELE\_NUM; ++i) {**

**elevator\* ele = eles[i];**

**if (ele->status == 0) {**

**int distance = abs(ele->currentFloor - floor);**

**if (distance < minDistance) {**

**minDistance = distance;**

**closestElevatorIndex = i;**

**}**

**}**

**else if ((up && ele->status == 1 && ele->currentFloor <= floor) || (!up && ele->status == 2 && ele->currentFloor >= floor)) {**

**int distance = abs(ele->currentFloor - floor);**

**if (distance < minDistance) {**

**minDistance = distance;**

**closestElevatorIndex = i;**

**}**

**}**

**}**

**}**

1. **最短剩余时间调度算法**：根据电梯完成当前任务队列中剩余任务所需时间来选择电梯。在ShortestRemainingTimeFirst函数中，创建elevatorTimes向量用于存储每个电梯的索引及其剩余时间。遍历eles电梯向量，调用eles[i]->calculateRemainingTime()函数计算每部电梯的剩余时间，并将电梯索引和剩余时间存入elevatorTimes。然后对elevatorTimes按剩余时间从小到大排序。按照排序顺序，尝试向剩余时间最短的电梯发送请求，若成功则显示调度信息，若所有电梯都拒绝请求，则强制调度剩余时间最短的电梯。
2. **分组调度算法**：将楼层分为高低层两组，根据请求楼层所在组选择电梯。在GroupScheduling函数中，先计算中间楼层middleFloor = FLOOR\_NUM / 2。根据请求楼层与中间楼层的关系确定目标电梯组：若请求楼层在中层及以下，则将当前楼层在中层及以下的电梯加入目标组；若请求楼层在中层以上，则将当前楼层在中层以上的电梯加入目标组。遍历目标组电梯，尝试向其发送请求，若成功则显示调度信息。若目标组内无电梯响应请求，则遍历所有电梯再次尝试发送请求。若所有电梯都无响应，则强制调度第一部电梯。
3. **FCFS 算法 + 任务最少**：此算法基于 FCFS 原则，优先将请求分配给任务最少的电梯。在ALGORITHM2函数中，先清空ele\_dest\_num向量，用于记录每部电梯的任务数量。遍历eles电梯向量，将每部电梯的编号和其目标楼层数（eles[i]->dests\_size()）存入ele\_dest\_num。然后对ele\_dest\_num按任务数量从小到大排序。按照排序顺序，尝试向任务最少的电梯发送请求，逻辑与 FCFS 算法 + 评分最优类似，即若成功则显示调度信息，若所有电梯都拒绝请求，则强制调度任务最少的电梯。
4. **FCFS 算法 + 顺序分配**：按照 FCFS 原则，顺序尝试将请求分配给电梯。在ALGORITHM3函数中，通过遍历eles电梯向量，依次尝试向每部电梯发送请求。若有电梯接收请求，则显示调度信息；若所有电梯都拒绝请求，则强制调度第一部电梯。
5. **FCFS 算法 + 评分最优**：该算法遵循先来先服务（FCFS）原则，并结合电梯评分来选择响应请求的电梯。在代码中，ele\_select\_send函数依据ELE\_SELECT\_MODE判断是否采用此算法。进入ALGORITHM1函数后，首先清空eleRatings向量，用于存储每部电梯的评分信息。通过遍历eles电梯向量，调用ele\_rate函数计算每部电梯的评分。ele\_rate函数综合考量电梯负载、与请求楼层的距离和运行状态：
6. 电梯负载评分loadRating通过当前电梯目标楼层数（eles[ele\_number]->dests\_size()）与总楼层数和电梯数乘积（ELE\_NUM \* FLOOR\_NUM）的比例得出。
7. 距离评分distanceRating利用指数函数std::pow(1.1, std::abs(eleFloor - reqFloor)) / std::pow(1.1, FLOOR\_NUM)加大距离影响，且当电梯处于空闲状态（eleStatus == 0）时，该评分乘以 3。
8. 状态评分statusRating根据电梯运行状态和请求方向确定。例如，请求向上且电梯向上运行，若电梯当前楼层小于请求楼层，则状态评分为 1.0；否则为 0.2 等。
9. ****综合各项评分100.0 \* (distanceRating \* 0.3 + statusRating \* 0.4 + loadRating \* 0.3)得到最终评分。计算完所有电梯评分后，对eleRatings向量按评分从小到大排序。接着尝试向评分最优的电梯发送请求，若成功，则在状态标签显示调度信息；若所有电梯都拒绝请求，则强制调度评分最低的电梯。

## **四、界面设计**

### **（一）电梯界面**

1. **整体布局**：电梯界面划分为两个区域，顶部区域展示电梯编号和按钮标志；中间区域为目的地楼层选择区域，通过按钮供用户选择目标楼层，以及操作按钮（开门、关门、报警）；底部区域显示电梯的运行状态、门状态和当前楼层。
2. **颜色与样式**：界面整体采用浅蓝色和淡绿色为主色调，按钮和标签的颜色依据功能进行区分，如开门按钮、关门按钮、报警按钮以图片形式展示，以此增强视觉辨识度。



### **（二）楼层界面**

1. **整体布局**：楼层界面分为两个区域，顶部区域显示电梯编号，以及数码显示器，用于显示电梯状态和所在楼层；底部区域为楼层按钮区域，包含每层的上行按钮、下行按钮。
2. **颜色与样式**：界面整体采用淡绿色为主色调，按钮和标签的颜色与电梯界面保持一致，使整个系统的界面风格统一协调。

### **（三）主控制界面**

1. **整体布局**：主控制界面分为两个主要区域，上方区域用于设置电梯数量、楼层数量和选择调度算法；下边区域为操作按钮区域，包括开始、暂停和退出按钮。
2. **颜色与样式**：界面整体采用浅灰色为主色调，按钮和标签的颜色根据功能进行区分，如开始按钮为绿色，暂停按钮为橙色，退出按钮为红色，方便用户操作。

## **五、开发环境与技术选型**

### **（一）开发环境**

1. **操作系统**：Windows 11
2. **开发工具**：vs2022+Qt
3. **编程语言**：C++

### **（二）技术选型**

1. **Qt 库**：用于开发图形用户界面，提供丰富的 UI 组件和便捷的开发方式。
2. **多线程技术**：利用 Qt 的 QThread 类实现电梯的多线程运行，确保电梯的独立运行和高效响应。
3. **信号与槽机制**：通过 Qt 的信号与槽机制实现各个模块之间的通信和事件处理，如按钮点击事件、电梯状态变化事件等。

## **六、项目总结**

1. **项目成果**

* 实现了一个功能完整的电梯操作系统，支持多种电梯调度算法和用户自定义功能。
* 具有直观的用户界面，方便用户操作和观察电梯的运行状态。
* 系统具有较高的稳定性和可靠性，能够长时间稳定运行。

1. **不足之处**

* 部分调度算法的实现可能不够优化，在高并发情况下可能会出现性能问题。
* 系统的可扩展性有待提高，如添加新的调度算法或功能可能需要对代码进行较大的修改。

1. **改进方向**

* 对调度算法进行进一步优化，提高系统的性能和响应速度。
* 采用更灵活的设计模式，增加系统的可扩展性，方便添加新的调度算法和功能。
* 进一步完善用户界面，提高用户体验。