进程管理——电梯调度项目设计报告

一、项目背景

1.1 项目背景

操作系统是计算机系统中管理硬件资源和协调应用程序运行的核心软件，进程调度、并发控制和资源管理是操作系统设计的重点与难点。为了加深对进程调度、线程通信、事件驱动机制等操作系统核心概念的理解，本项目设计并实现了一个简化的电梯调度系统，模拟多任务环境下的进程调度与资源协作过程。

1.2 项目目的

本项目通过 C++ 和 Qt 开发图形化界面，模拟实现一个多楼层电梯调度系统，旨在掌握：

- 操作系统中进程/线程调度策略
- 事件驱动机制的应用
- 并发请求的同步与互斥处理
- 信号与槽机制实现进程通信模型

通过项目实践，将理论知识与实际程序开发相结合，提升操作系统核心概念的应用能力。

二、开发环境

2.1 硬件和软件环境

操作系统：Windows 11

开发工具：Visual Studio 2022

编程语言：C++

图形界面框架：Qt，用于跨平台的图形用户界面开发，支持事件驱动模型和界面控件。

2.2 主要依赖库

2.2.1 C++ 标准库

包含 iostream、thread、climits、vector 等。

2.2.2 Qt 相关库

包含 QMutexLocker、QDebug、QGraphicsDropShadowEffect、QVBoxLayout、QLabel、QFrame 等。

三、系统需求分析

3.1 功能需求

- 实现电梯上行、下行、停靠、开关门功能

- 支持电梯内部楼层按钮请求
- 支持楼层外部上行、下行按钮请求
- 实时显示电梯当前楼层和运行状态
- 电梯日志栏显示电梯调度和运行信息

3.2 非功能需求

- 界面美观，操作直观
- 系统响应及时，运行稳定

四、系统设计

4.1 系统架构

本电梯控制系统采用面向对象的设计模式，基于 Qt 框架开发，包含多个模块来实现电梯的调度与控制。系统的主要功能是模拟电梯在不同楼层间的运行，根据用户请求调度电梯并显示电梯的运行状态。系统的核心模块包括调度器（Dispatcher）、电梯控制（Elevator）、图形界面（ElevatorWidget）、楼层按钮控件（FloorButtonWidget）、日志面板（LogPanel）以及主界面（MainWindow）。

系统采用了多线程模型，确保电梯调度、运行和界面显示可以并行处理。各个模块通过信号和槽机制进行通信和协作。

4.2 系统逻辑

4.2.1 用户请求

用户通过点击界面中的楼层按钮（FloorButtonWidget），发起对电梯的请求。每个楼层的按钮都绑定了一个信号，当按钮被点击时，会触发信号，通知调度器（Dispatcher）进行处理。

4.2.2 调度器处理

调度器（Dispatcher）是系统的核心，它管理所有楼层的请求，并根据电梯当前的状态决定哪个电梯执行任务。调度器通过计算电梯的当前位置和目标楼层，选择最合适的电梯进行调度。如果多个电梯可以执行任务，调度器会根据设定的策略（如最近优先、空闲电梯优先）进行调度。

4.2.3 电梯执行任务

每个电梯模块（Elevator）都运行在独立的线程中，负责处理来自调度器（Dispatcher）的任务请求。每个电梯可以接收多个任务，并将这些任务组织成一个任务队列，依次完成每个任务。

任务接收与排队：电梯接收到一个任务时，会将该任务添加到任务队列中。每个任务包

括目标楼层和任务类型（如上行、下行、到达后停留等）。

任务执行：电梯模块的线程会不断从任务队列中获取任务并执行。电梯会根据任务的要求调整其状态（如开始上行、下行等），并模拟电梯的运行过程。

任务顺序执行：电梯会按照任务队列的顺序依次执行任务。完成一个任务后，电梯会从队列中移除该任务，继续执行下一个任务。电梯的每次任务完成后，电梯模块会通过信号通知图形界面（ElevatorWidget）更新显示。

任务完成与待命：当所有任务都执行完毕后，电梯模块会返回待命状态，等待新的任务请求。

每个电梯的线程独立运行，能够同时处理多个任务，并且确保按照顺序执行。这样可以有效模拟电梯的并行操作和任务管理。

4.2.4 电梯界面

电梯界面（ElevatorWidget）负责显示电梯的位置、楼层按钮的状态和电梯的运行状态。它根据电梯模块的信号实时更新电梯的图形位置，模拟电梯在不同楼层之间的移动。同时，界面上还会展示电梯当前的状态（如上行、下行、停止等）以及日志信息。

4.2.5 日志面板

每当电梯开始和完成一次任务（如到达目标楼层或发生异常），日志面板（LogPanel）会记录下这次操作，并显示在界面上。日志面板用于系统监控，帮助开发者跟踪电梯的运行过程，方便调试和分析。

4.2.6 主窗口界面

主窗口（MainWindow）作为整个应用程序的入口，管理所有其他控件和模块的显示与交互。它负责初始化并显示界面，管理电梯的调度和用户输入的响应。

4.3 模块间的交互

4.3.1 FloorButtonWidget -> Dispatcher

用户点击楼层按钮时，FloorButtonWidget 发出信号通知 Dispatcher 进行电梯调度。

4.3.2 Dispatcher -> Elevator

Dispatcher 根据楼层请求和电梯状态，选择适合的 Elevator 来执行任务，发出控制目标楼层信号。

4.3.3 Elevator -> ElevatorWidget

电梯位置更新时，Elevator 发出信号通知 ElevatorWidget 更新图形界面，模拟电梯的移动。

4.3.4 Elevator -> LogPanel

当电梯加入任务和完成任务时，Elevator 通过信号通知 LogPanel 记录操作日志。

4.3.5 MainWindow -> 其他模块

MainWindow 负责协调各个模块的初始化和展示，确保系统正常启动并运行。

五、核心功能实现

5.1 电梯控制模块（Elevator）

5.1.1 核心数据结构

①Direction 枚举

IDLE：电梯处于待命状态。

UP：电梯正在上行。

DOWN：电梯正在下行。

②QSet<int>

up_tasks：电梯上行任务队列，用于存储电梯需要前往的更高楼层。

down_tasks：电梯下行任务队列，用于存储电梯需要前往的更低楼层。

5.1.2 多线程思想

电梯控制模块运行在独立的线程中，以便多个电梯能够并行运行。每个电梯在独立线程中执行任务，确保电梯的任务可以并行处理。

5.1.3 任务队列

①up_tasks 和 down_tasks 的作用

这两个任务队列确保电梯能根据请求按顺序处理目标楼层。up_tasks 存储上行任务，down_tasks 存储下行任务。当电梯接收到任务请求时，它会根据楼层的大小判断任务应该添加到哪个队列。

电梯通过任务队列进行任务调度，从而避免了任务冲突和不必要的重复执行。例如，如果电梯正在上行，它会优先处理上行任务队列中的楼层请求，直到所有上行任务完成，才会转向处理下行任务。

②方向更新和任务调度

电梯在每次完成任务后，会检查当前方向和任务队列。如果电梯当前的任务队列为空，它会改变方向，转向下一个非空任务队列。当电梯任务队列为空时，电梯进入待命（IDLE）状态，停止运动。

5.2 电梯图形化模块（Elevator Widget）

5.2.1 核心数据结构

① **QWidget**

ElevatorWidget 继承自 QWidget，是电梯的图形界面组件，用于显示电梯的当前状态、楼层及运动方向。该组件通过与 Elevator 控制模块的信号连接，实时更新电梯状态。

② QLabel 和 QPainter

QLabel 用于显示电梯的当前楼层、状态等信息。

QPainter 用于绘制电梯的运动轨迹和图形界面中的动态效果，如电梯位置、楼层指示等。

5.2.2 图形界面设计思想

电梯图形化模块通过 QWidget 类和 Qt 的事件处理机制,在 GUI 界面上呈现电梯的动态变化。设计上采用响应式布局，确保界面能够根据实际电梯数量和楼层数进行自适应调整。

5.2.3 多线程与事件驱动

①多线程支持：
电梯图形化模块与电梯控制模块分离，电梯的控制逻辑运行在独立线程中，而图形化模块则通过 Qt 的信号槽机制与控制模块交互，保证界面与后台逻辑的实时同步。

②事件驱动更新：
ElevatorWidget 通过监听电梯控制模块发出的信号（如 floorChanged、arrivedAtFloor）来动态更新电梯的界面显示。例如，当电梯到达某楼层时，界面自动刷新电梯位置，确保图形界面的实时性和响应速度。

5.2.4 任务与状态更新

①楼层更新：每当电梯完成一次任务或改变楼层时，ElevatorWidget 会收到 floorChanged 信号，从而更新电梯当前楼层。

②状态更新：电梯到达楼层后，图形界面会通过 arrivedAtFloor 信号更新状态，显示电梯到达的楼层，并调整电梯的方向图标。

③动态显示：电梯每次更新楼层时，会重新绘制电梯的位置，使得电梯的上下移动在图形界面上直观呈现。当电梯完成任务后，会触发开门/关门事件，图形界面相应显示开关门状态，增加交互感。

5.3 用户请求模块(FloorButtonsWidget)

5.3.1 核心功能

FloorButtonsWidget 提供了一个图形化的用户界面，允许用户通过点击相应楼层的“上升”或“下降”按钮来请求电梯。每个楼层都具有相应的按钮，用户通过点击这些按钮发起电梯的请求。

楼层按钮布局：所有楼层的按钮被排列在一个垂直布局中，楼层按降序排列，从 20 楼到 1 楼。每层有两个按钮：

上升按钮：除去 20 楼，其他楼层都有“上升”按钮，表示用户希望从当前楼层向上呼叫电梯。

下降按钮：除去 1 楼，其他楼层都有“下降”按钮，表示用户希望从当前楼层向下呼叫电梯。

5.3.2 信号与槽机制

在用户点击按钮后，信号会被发射，用于通知电梯控制模块以响应请求：

floorRequested 信号：每当用户点击上升或下降按钮时，相应的信号会携带楼层号及方向信息，通知电梯模块执行操作。

上升按钮会发射 floorRequested(i, true) 信号，表示请求上行电梯。

下降按钮会发射 floorRequested(i, false) 信号，表示请求下行电梯。

5.4 调度模块(Dispatcher)

Dispatcher 模块是整个电梯系统中的任务调度中心，主要负责根据用户的请求来合理选择并分配任务给各个电梯。它并不直接控制电梯的运动，而是根据任务的特点（如任务的方向、楼层等）选择最合适的电梯进行任务分配，具体的任务执行则由电梯模块自己负责。该模块确保任务高效、准确地分配到最适合的电梯，并根据电梯的状态（如是否空闲、是否顺路等）做出合理的决策。

5.4.1. 功能概述

Dispatcher 模块的核心功能是通过判断电梯的当前状态和任务请求的特点，决定将任务分配给哪个电梯。任务的分配逻辑主要考虑以下几点：

顺路电梯优先：优先选择与任务方向一致的电梯。

空置电梯优先：若没有顺路电梯，则选择空闲电梯进行任务分配。

逆行电梯分配：若没有顺路或空置电梯，则选择逆行的电梯。

调度模块的设计确保任务能快速响应，并尽量减少电梯空驶，提升系统整体效率。

5.4.2 任务分配逻辑

Dispatcher 模块的任务分配是根据以下几个原则进行的：

①顺路电梯优先

首先，调度器会遍历所有电梯，寻找与当前请求任务方向一致的电梯。电梯的方向判断基于任务的方向与电梯当前的运行方向是否匹配。电梯正在上行时，会优先分配给请求上行的楼层；同理，电梯正在下行时，会优先分配给请求下行的楼层。

②空置电梯次优

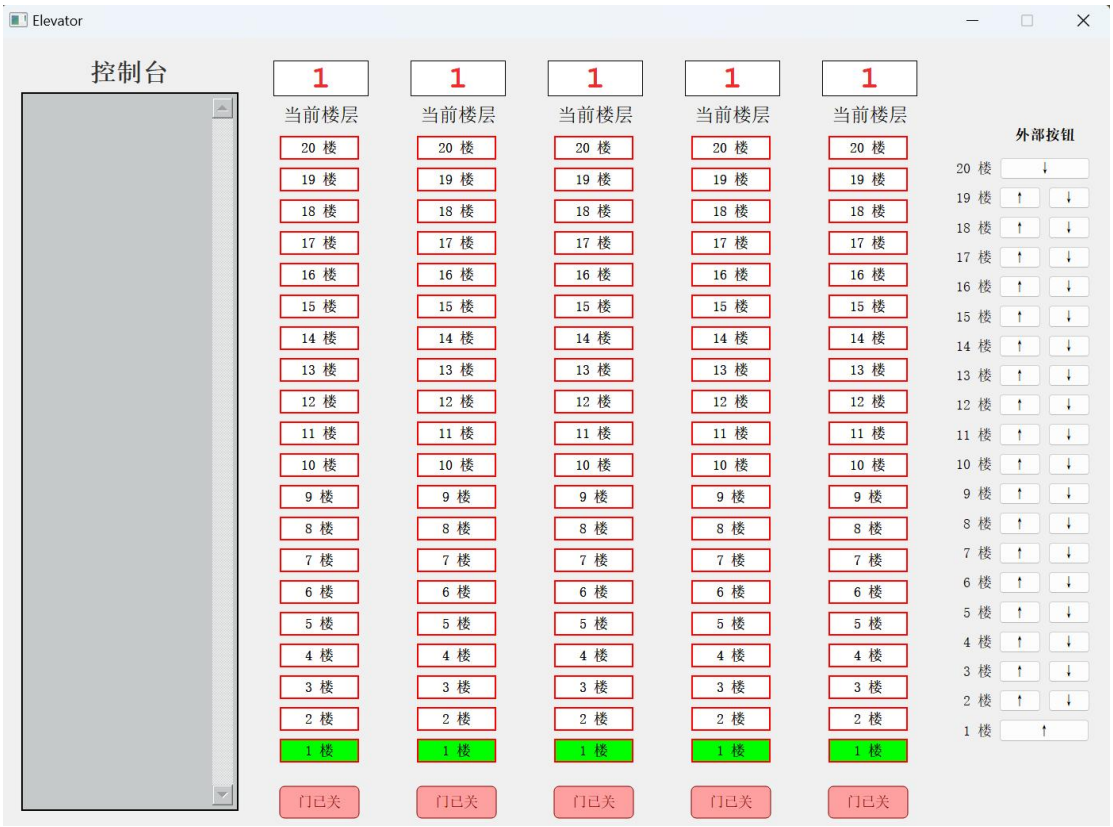
如果没有合适的顺路电梯，调度器会优先选择当前处于待命状态（即空闲）的电梯。这些电梯通常是可以随时接收新任务并迅速响应，因此是任务分配中的次优选择。

③逆行电梯分配

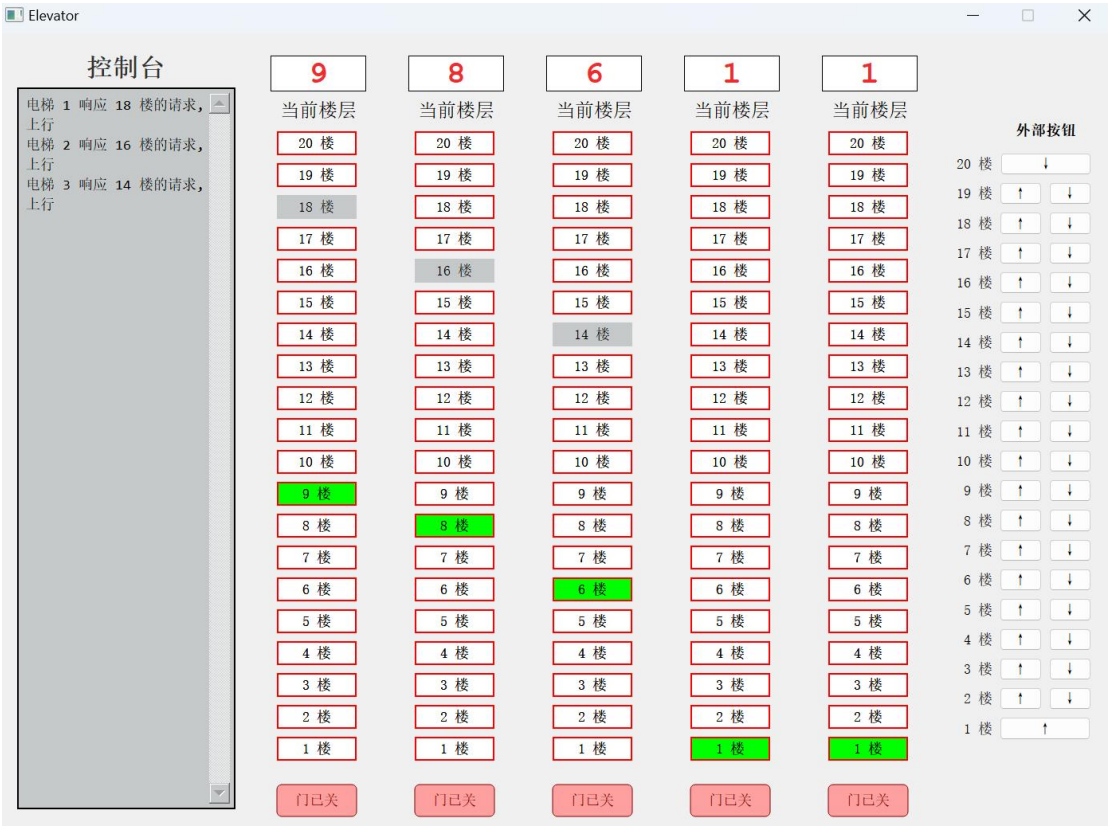
如果没有顺路电梯和空置电梯可用，则调度器会选择逆行电梯。逆行电梯是指电梯的行驶方向与任务的方向相反。虽然这种选择可能会导致一些额外的空驶时间，但它保证了即使在电梯繁忙的情况下，任务也能被处理。多辆逆行电梯时，选择与当前楼层差值的绝对值最小的那辆电梯。

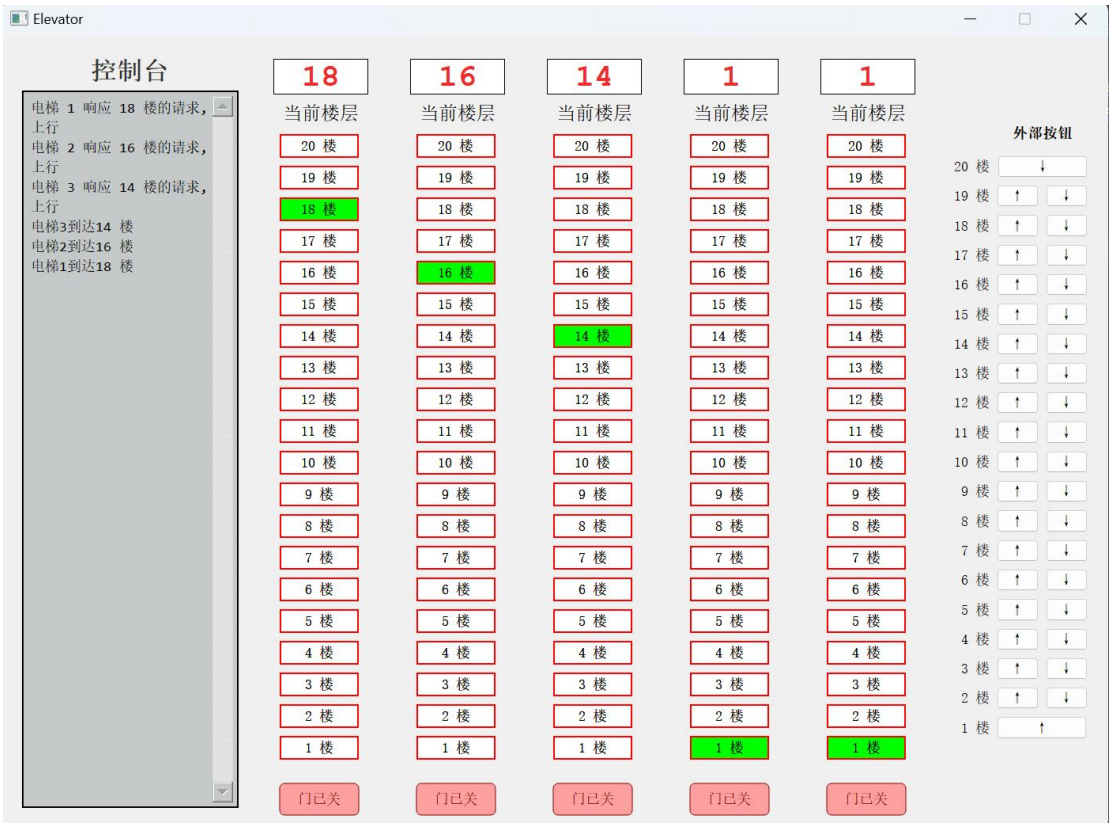
六、实现效果

6.1 初始 ui 界面



6.2 电梯运行界面


Elevator
控制台
电梯 1 响应 18 楼的请求，上行
电梯 2 响应 16 楼的请求，上行
电梯 3 响应 14 楼的请求，上行
9
8
6
1
1
当前楼层
20 楼
19 楼
18 楼
17 楼
16 楼
15 楼
14 楼
13 楼
12 楼
11 楼
10 楼
9 楼
8 楼
7 楼
6 楼
5 楼
4 楼
3 楼
2 楼
1 楼
门已关
外部按钮


Elevator
控制台
电梯 1 响应 18 楼的请求，上行
电梯 2 响应 16 楼的请求，上行
电梯 3 响应 14 楼的请求，上行
电梯3到达14 楼
电梯2到达16 楼
电梯1到达18 楼
18
16
14
1
1
当前楼层
20 楼
19 楼
18 楼
17 楼
16 楼
15 楼
14 楼
13 楼
12 楼
11 楼
10 楼
9 楼
8 楼
7 楼
6 楼
5 楼
4 楼
3 楼
2 楼
1 楼
门已关
外部按钮