****

**Tongji University**

**操作系统课程项目实验报告**

**001-电梯调度**

专 业： 软 件 工 程

指导教师： 张 慧 娟

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **项目及基本要求概述**
   1. **项目概览**
      1. **项目目的**
2. 通过控制电梯调度，实现操作系统调度过程；
3. 学习特定环境下多线程编程方法；
4. 学习调度算法。
   * 1. **项目要求**
5. 基本任务：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。
6. 功能描述：
7. 电梯应有一些按键，如：数字键、关门键、开门键、上行键、下行键、报警键等；
8. 有数码显示器指示当前电梯状态；
9. 每层楼、每部电梯门口，有上行、下行按钮、数码显示。
10. 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。
11. 电梯调度算法：
12. 所有电梯初始状态都在第一层；
13. 每个电梯没有相应请求情况下，则应该在原地保持不动；
14. 电梯调度算法自行设计。
    1. **项目环境**
       1. **图形化界面**

本次项目需要对界面进行图形化设计，因此选择当下热门的QT作为图形化设计的工具。

* + 1. **编程语言**

本次项目选择使用python作为项目的编程语言，因此使用qt的python适配库pyqt5作为图形化设计的工具。其中，python版本为python3.10，添加pyqt5等库内容，并在pycharm编辑器上进行代码的书写，Debug和打包运行。

* + 1. **操作系统平台**

本次项目选择使用Windows作为项目操作系统平台，所有程序均在Windows10系统下运行。

1. **整体设计思路**

本次项目整体设计包括四个部分，即项目界面设计，电梯功能设计，多线程设计和调度算法的实现。

* 1. **项目界面设计**

项目界面主要包括四个模块：

1. 顶部显示屏：顶部显示屏需要能够显示电梯当前所在的楼层数，同时也需要在电梯运动时不断更新。
2. 楼层及电梯位置显示：共有20层楼，共5个电梯井。需要设置5\*20个小矩形组成方阵，同时用不同的颜色标明电梯当前所在的矩形空间（即A电梯当前在B楼层 = 矩形（A,B）与其他矩形颜色不同）。令用户可以直观看到电梯的位置。
3. 电梯内部按钮设置：需要设置电梯内部按钮，包括
4. 在每个楼层旁边设置数字按钮，表示这是电梯内部的楼层按钮；
5. 在每台电梯下面设置开门，关门和报警三个按钮。
6. 电梯外部按钮设置：要在每个楼层旁边设置呼叫电梯的按钮，应为上行/下行成对出现。
   1. **电梯功能设计**

电梯功能主要面向用户考虑：

1. 电梯需要在用户按下呼叫按钮后到达用户所在的楼层。
2. 用户可以在电梯内部点击各个楼层按钮控制电梯的移动；
3. 用户能够通过内部的开关门按钮实现电梯门开关；
4. 按下报警按钮后，电梯停止工作，一切任务分流给能够正常运行的电梯；
   1. **多线程设计**

多线程设计主要需要从现实意义和代码逻辑两个方面考虑，总共可以分为五个要点：

1. 状态同步：需要确保不同线程之间的状态是同步的，如电梯是否开门、电梯是否在运行等。
2. 任务调度：需要考虑如何调度任务，如何安排电梯的运行路线等。
3. 界面更新：需要确保电梯的运行状态能够实时反映在界面上，如当前楼层、电梯状态等。
4. 安全性考虑：需要确保电梯在运行过程中不会发生意外，如碰撞、坠落等。这通常需要考虑各种情况下的安全措施，如报警系统等。
5. 代码可读性和可维护性：需要考虑代码的可读性和可维护性，如封装、代码复用等。这有助于减少代码的bug，并方便后期的维护和升级。
   1. **调度算法设计**

电梯调度算法主要有几种实现方式，其优缺点和适合的场景各不相同：

1. FCFS（先来先服务）算法：按照请求的顺序进行服务，没有优先级的考虑，存在“饥饿”问题（多个线程竞争同一资源的情况下，如果某些线程总是优先于其他线程获取到该资源，那么其他线程就会饿死，无法执行自己的任务）。
2. SSTF（最短寻找时间）算法：每次选择距离当前电梯位置最近的楼层进行服务，在某些情况下能够获得最短的调度时间，但是存在“优先级反转”问题（低优先级的线程获取了高优先级线程所需要的资源，导致高优先级的线程处于等待状态）。
3. SCAN（扫描）算法：按照一个方向扫描所有请求，直到到达一个极限位置后改变方向，解决了“饥饿”问题，但存在“优先级反转”问题。
4. C-SCAN（循环扫描）算法：和SCAN算法类似，但到达一个极限位置后直接返回另一个极限位置，解决了“优先级反转”问题。
5. LOOK算法：类似于SCAN算法，但只扫描当前有请求的楼层，而不是整个电梯轨道，解决了“优先级反转”问题。
6. C-LOOK算法：类似于C-SCAN算法，但只扫描当前有请求的楼层，而不是整个电梯轨道，解决了“优先级反转”问题。

在实际设计中，可以根据不同的需求和场景选择不同的算法，例如：对于高峰期来说，优先考虑响应时间的短暂性，可以采用SSTF或LOOK算法；对于低峰期来说，优先考虑公平性，可以采用FCFS算法；对于楼层密集的地区，可以采用SCAN或C-SCAN算法等等。

本次设计考虑到时间和空间效率，选择基于LOOK算法开发电梯调度算法。

1. **项目功能实现**
   1. **项目界面实现**
      1. **界面绘制**

项目界面利用pyqt5进行代码化绘制，主要通过设置多个水平布局对象和竖直布局对象的嵌套来进行按钮等控件的呈现。以下为实现方法说明：

1. 在画布上设置底层水平布局对象QVBoxLayout()，用来存放各个楼层、显示屏和外部按钮的布局对象；
2. 在底层布局对象中添加次底层水平布局对象，其负责存放五个电梯及显示屏、内部楼层按钮和开关门按钮，报警按钮等控件；
3. 在次底层水平布局对象中添加五个竖直布局对象，分别对应五台电梯及其相应的控件。
4. 其中每个竖直布局对象包括一个显示屏，二十个楼层标识，二十个内部楼层按钮和内部开关门以及故障按钮控件；

其中白色的部分表示普通的楼层，灰色的部分表示电梯当前所在位置。

1. 在底层布局对象中添加竖直布局对象，其中存放二十对外部呼叫按钮，每对各为一个水平布局对象存放在次底层竖直布局对象中；需要注意，第一层不能有向下的按钮，第二十层不能有向上的按钮。旁边设置提示框方便用户理解与使用；

用户在与不同按钮交互后，按钮的颜色慧发生改变。

1. 调节窗口大小，将窗口抬头设置为OS\_Lab1\_Elevator;
   * 1. **信号槽函数实现**

为了建立起UI界面和内部程序的联系，需要设置信号槽函数：

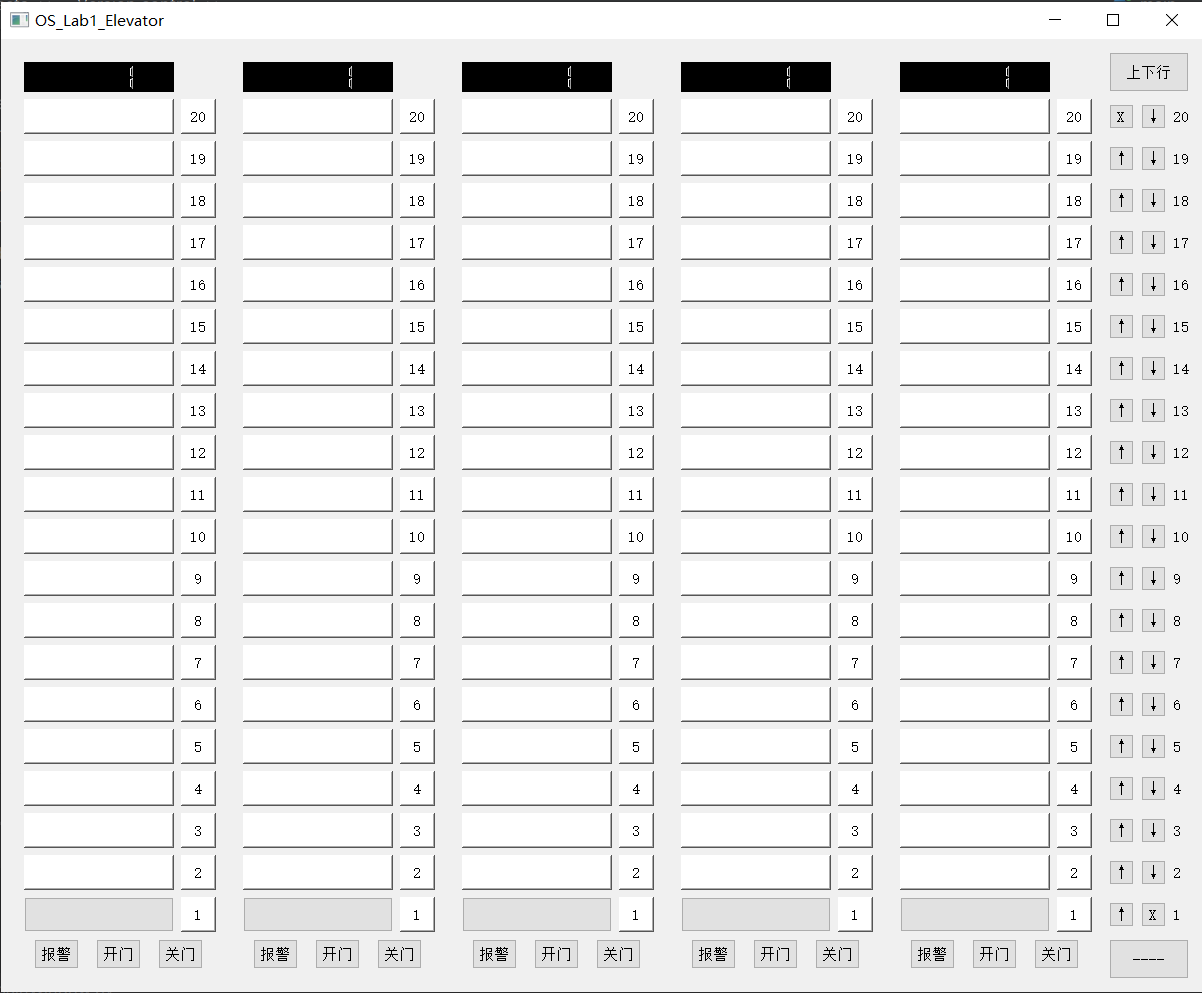
1. 设置内部楼层点击函数inner\_floor\_mark\_clicked：若用户按下电梯内部的楼层按钮，则此按钮变为黄色，标识目标楼层，同时更改内部全局变量，向电梯分配任务，电梯到达用户指定楼层后按钮恢复原状。
2. 设置外部呼叫点击函数outer\_direction\_btn\_clicked：若用户按下外部呼叫按钮，则此按钮变为黄色，直到电梯来到用户所在楼层后按钮恢复原状。函数内部向存储电梯任务的全局变量添加新任务，指导电梯来到任务楼层接引乘客。
3. 设置内部故障点击函数inner\_fault\_btn\_clicked：若用户按下内部故障按钮，如果当前电梯处于正常运行状态，则此按钮变为红色，标识电梯故障；如果此时电梯处于故障状态，则按钮恢复原状，标识电梯可以正常运行。
   * 1. **UI更新函数的视线**

多线程设计需要对UI进行实时的更新，则需要设置函数不断刷新UI。

这一函数需要更新四个模块：

1. 显示屏内容更新
2. 开关门按钮更新
3. 电梯位置更新
4. 外部按钮更新

整体界面实现效果如下：



* 1. **电梯功能实现**

电梯功能主要分为内部和外部两个部分。

* + 1. **电梯内部功能实现**

电梯内部功能主要需要实现楼层的转移，故障和开关门。

1. 用户在进入电梯后可以按下电梯内部的楼层按钮（界面中为每个楼层旁边的数字小方块按钮），作为用户希望去往的目标楼层给电梯下达指令。电梯得到指令后会移动到目标楼层，主要依托灰色矩形的移动作为可视化实现。

代码中设置InnerCmd(QThread)类作为存放内部功能函数的类。

其中One\_Floor函数能够使电梯向指定方向移动一层，run函数能够根据用户的指令需求控制电梯移动到目标楼层，这一功能主要通过run函数循环嵌套调用One\_Floor函数实现。

1. 电梯故障功能的实现：电梯故障时，需要将电梯的所有状态都初始化，并将电梯标识为不可用状态。此时电梯正在执行的任务都重新设置为未分配状态，并汇入任务流重新分配给其他电梯。
2. 电梯开关门功能：电梯开关门功能主要通过颜色改变来表明电梯门的运行状态。在每次到达楼层或者用户有需求时进行。
   * 1. **电梯外部功能的实现**

电梯外部功能主要在于获取用户的外部命令后，将其转化为代码逻辑，运用调度算法计算出分配方案，并交给各个电梯进行执行。执行过程中依旧是使用内部的电梯移动函数进行移动。

* 1. **多线程设计**

Python下的多线程设计主要通过继承QThread类来实现。项目中为了进一步确保安全性，减少饥饿问题等风险，还设置了mutex互斥锁来保障程序运行。

* + 1. **QThread线程类**

1. QThread是Qt框架中提供的一个线程类，可以用于多线程编程。
2. 将项目中InnerCmd类作为线程类的子类，通过重写run()方法，实现线程的逻辑。
3. 在主函数main中通过start()方法启动线程，线程启动后就会进入run()方法执行，直到线程结束。
4. 线程结束后，QThread对象可以被再次启动或被销毁。
   * 1. **Mutex互斥锁**
5. QMutex 是 Qt 框架中的一个类，用于保护共享资源，在多线程环境下防止多个线程同时访问和修改共享资源，从而造成数据竞争和其他问题。
6. 在多线程程序中，可能会出现线程饥饿（starvation）的问题，即某些线程一直无法获得所需要的资源，而一直处于等待状态，无法执行任务。这通常是由于某些线程占用了共享资源并长时间不释放，导致其他线程无法访问该资源而一直处于等待状态。
7. 另外一个常见的问题是优先级反转（priority inversion）问题，当多个线程共享同一个资源时，高优先级线程等待低优先级线程释放资源，而低优先级线程又在等待高优先级线程释放资源，导致优先级反转，使得高优先级线程无法及时执行。
8. 互斥锁主要用于解决这两个问题，对于后者，通过使用优先级继承（priority inheritance）的方法，即当一个低优先级线程占用了某个资源时，将其优先级提升到高优先级线程的优先级，从而避免优先级反转问题。
   1. **调度算法的实现**

本次项目的调度算法选择使用LOOK算法。LOOK算法是一种磁盘调度算法，它是SCAN算法的改进版。与SCAN算法不同的是，LOOK算法不会在到达磁盘最外侧或最内侧时立即改变方向，而是在到达最外侧或最内侧时检查是否有等待服务的请求，如果没有，就改变方向；如果有，就继续服务这些请求，直到没有请求时再改变方向。LOOK算法的优点是平均寻道时间较短，且不会出现像SCAN算法那样的饥饿现象；缺点是容易产生优先级反转问题。然而，对于普通的电梯问题，并不存在过多的优先级问题（除了部分特殊场景需要设置电梯运行优先级），因此使用LOOK算法是当前情况下空间，时间利用性价比较高的选择。

程序调度算法依据LOOK算法的思想，在确保有能够正常运行的电梯后，会根据需求与电梯运行方向是否相同，需求是否在电梯上方/下方，电梯是否有任务这三个“是否”，共分为2\*2\*2=8种情况。然而具体实现时并不需要这么多分类：

1. 第一种情况，若电梯空闲，则可以直接调度绝对距离最小的电梯即可；
2. 第二种情况，电梯运行方向和需求与电梯的相对方向相同，则直接去往需求楼层即可（可以理解为“顺路”）；
3. 其余情况，计算最远任务楼层到目标楼层的绝对值和最远楼层到当前电梯楼层的绝对值之和。

而根据LOOK算法，将电梯设置为到达最上方或最下方时检测是否有等待的请求，如果没有，就改变方向；如果有，就继续给予服务，直到没有请求时再改变方向。

1. **运行方法**

本次项目源文件运行方法如下：

1. 使用python3.10，并添加如下python库

|  |  |  |
| --- | --- | --- |
| PyQt5 | 5.15.9 | 5.15.9 |
| PyQt5-Qt5 | 5.15.2 | 5.15.2 |
| PyQt5-sip | 12.12.1 | 12.12.1 |
| click | 8.1.3 | 8.1.3 |
| colorama | 0.4.6 | 0.4.6 |
| pip | 22.3.1 | 23.1.2 |
| pypiwin32 | 223 | 223 |
| pyqt5-plugins | 5.15.9.2.3 | 5.15.9.2.3 |
| pyqt5-tools | 5.15.9.3.3 | 5.15.9.3.3 |
| python-dotenv | 1.0.0 | 1.0.0 |
| pywin32（非必须） | 306 | 306 |
| qt5-applications | 5.15.2.2.3 | 5.15.2.2.3 |
| qt5-tools | 5.15.2.1.3 | 5.15.2.1.3 |
| setuptools | 65.5.1 | 67.7.2 |
| wheel | 0.38.4 | 0.40.0 |

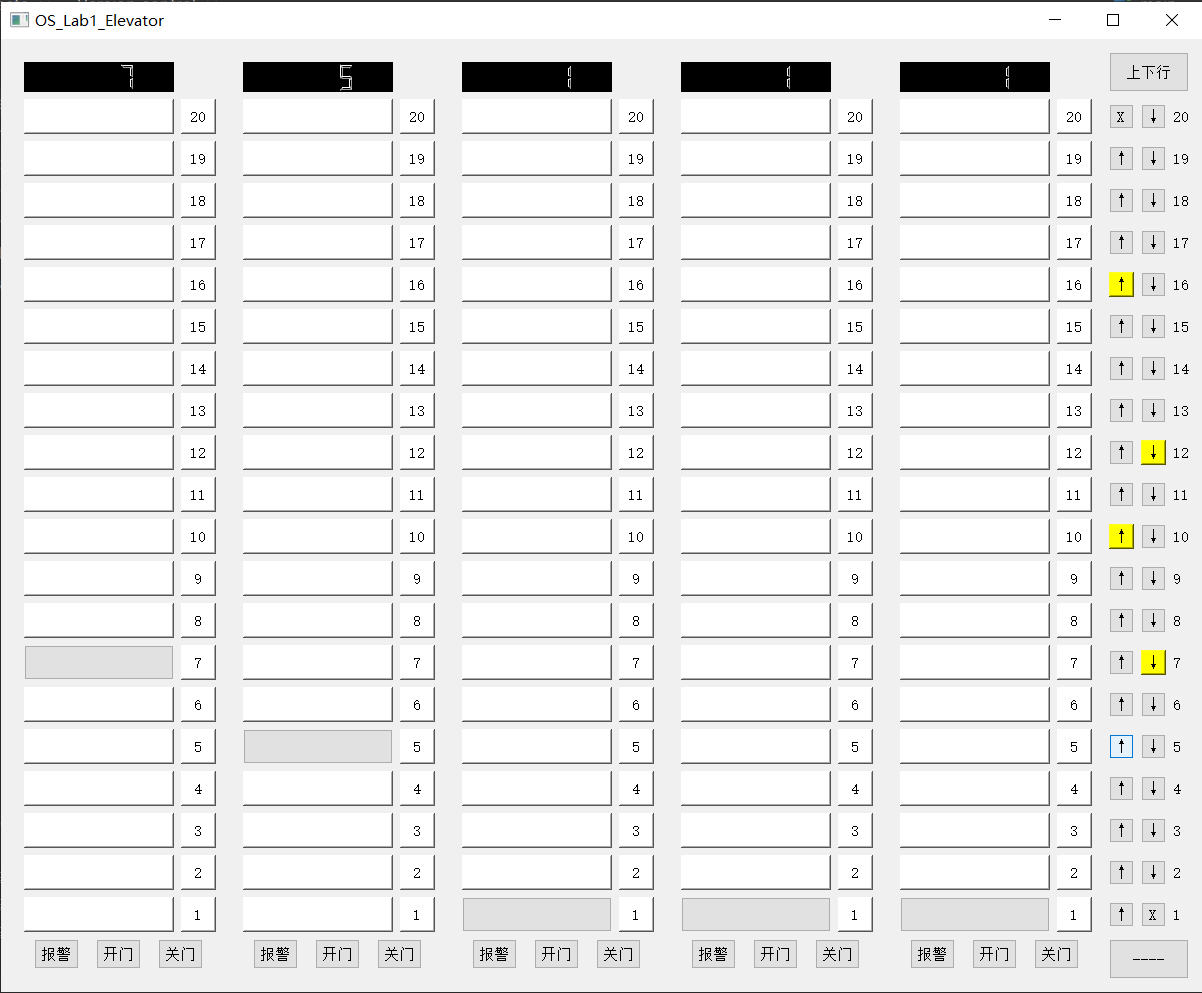
1. 本次项目IDE为pycharm（非必须）；
2. 将代码包中main.py和mainWindow.py放在同一文件目录下；
3. 在Windows环境下运行main.py；
4. 也可直接运行.exe文件，此文件由pyinstaller生成；
5. **心得体会**

本次项目无论是图形化界面，或是多线程编程，再到调度算法，都具有一定的挑战性，在代码编辑和调试过程中发现不少问题：

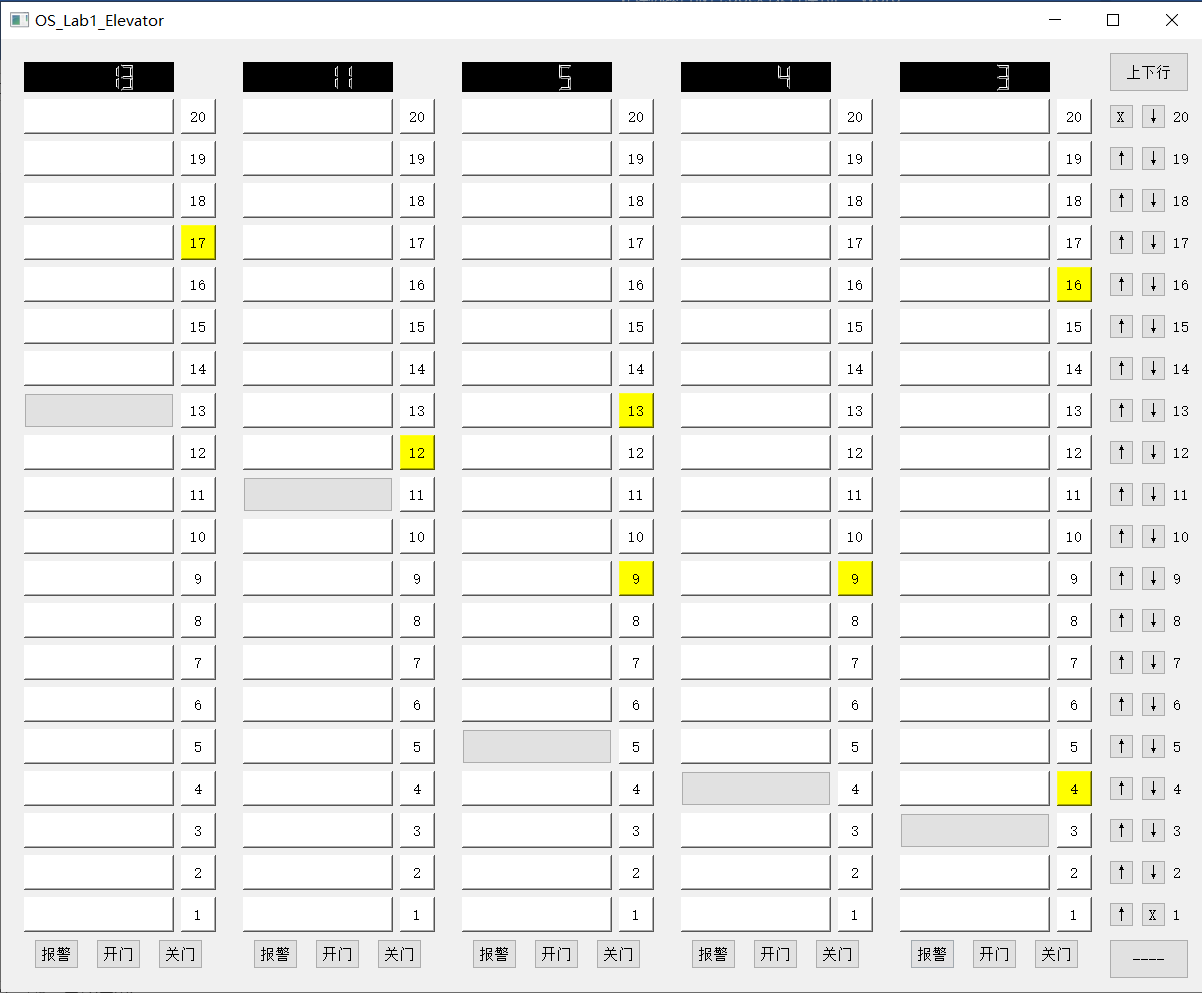
1. 在多线程中贸然使用win32api中的QMessageBox弹窗接口，可能会产生阻塞线程进而导致程序崩溃的问题；
2. 变量命名时建议使用命名法，选取具有代表性的命名，python不必申明变量的同时也使得变量命名和使用过程中可能存在错写漏写的错误，有时会导致严重的程序崩溃和泄露问题。命名法推荐如下：
3. 驼峰命名法（Camel Case）：首字母小写，之后每个单词的首字母大写，例如：firstName、lastName、phoneNumber；
4. 帕斯卡命名法（Pascal Case）：每个单词的首字母都大写，例如：FirstName、LastName、PhoneNumber；
5. 下划线命名法（Snake Case）：单词之间使用下划线连接，例如：first\_name、last\_name、phone\_number；
6. 匈牙利命名法（Hungarian Notation）：变量名前缀使用一个或多个小写字母表示变量类型，例如：iIndex、szName、bSuccess。
7. 很多单线程库函数在多线程中调用时都有可能阻塞线程，导致程序崩溃，如sleep函数等。
8. **程序运行截图**

程序运行截图如下

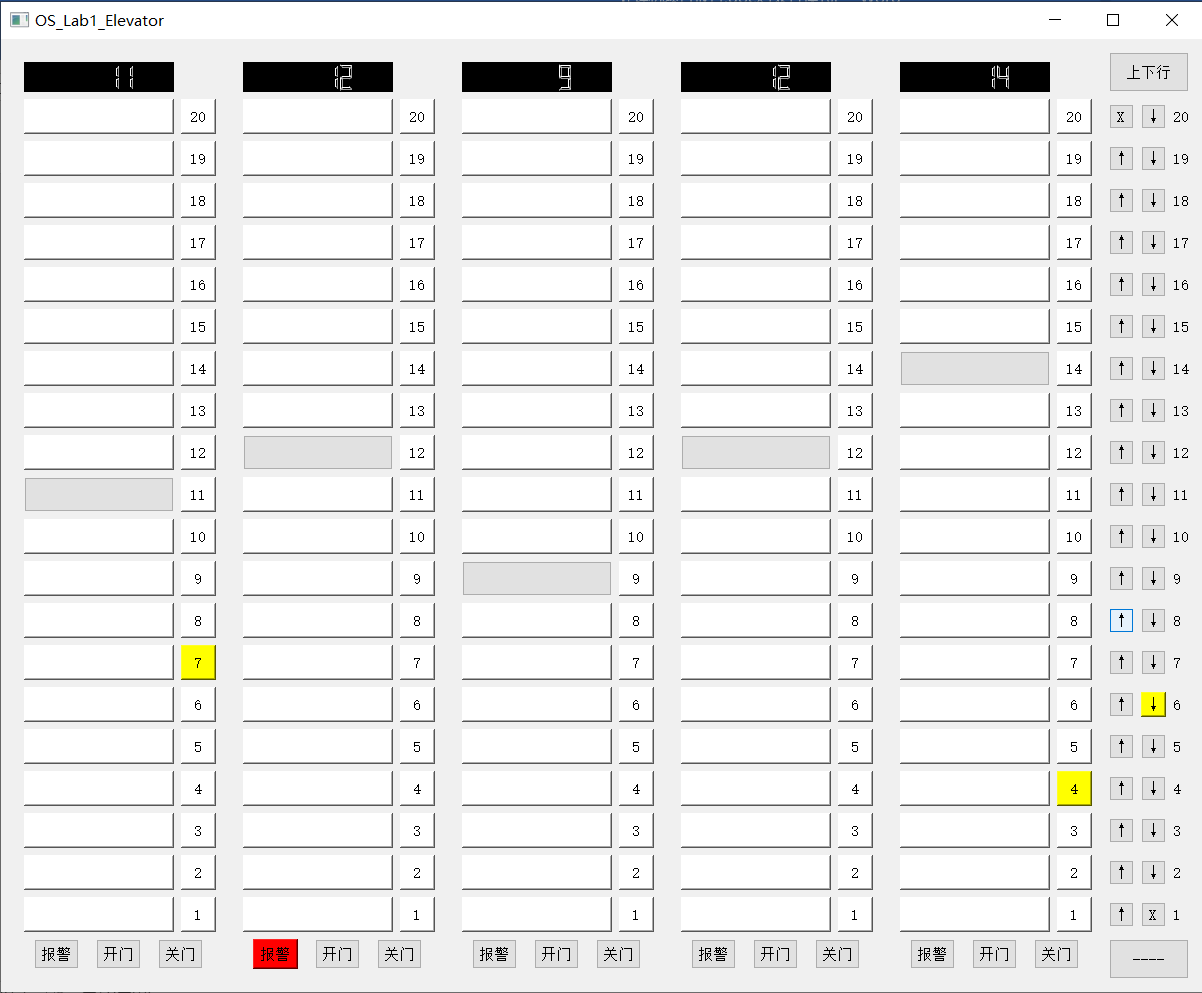
* 1. **外部呼叫按钮**

****

* 1. **内部楼层选择**

****

* 1. **报警及开关门**

****