操作系统项目说明文档

——处理机项目之电梯调度

作 者 姓 名： 马威

学 号： 2151294

指 导 教 师： 王冬青

学院、 专业： 软件学院 软件工程

****

**目 录**

[1.项目分析 1](#_Toc26249)

[1.1 项目背景 1](#_Toc21459)

[1.2 项目要求 1](#_Toc17663)

[1.3 项目需求分析 1](#_Toc20880)

[2.项目设计 2](#_Toc10766)

[2.1 开发环境选择 2](#_Toc2736)

[2.2 界面设计 2](#_Toc15463)

[2.3 调度算法设计 3](#_Toc13979)

[2.3.1 算法思路 3](#_Toc15204)

[2.3.2 算法实现 3](#_Toc11919)

[2.3.2.1 信息存储 3](#_Toc32108)

[2.3.2.2 按下电梯内数字键的情况 5](#_Toc20719)

[2.3.2.3 按下楼道内按键的情况 5](#_Toc10887)

[2.3.2.4 按下报警键的情况 7](#_Toc9227)

[2.3.2.5 线程设计 9](#_Toc23687)

[2.3.2.6 常规维护槽函数设计 10](#_Toc23595)

[2.3.2.7 事件的处理 12](#_Toc22890)

[2.3.3 算法总结 12](#_Toc18140)

3.项目测试 13

4.项目评价 13

[4.1 可改进的地方 13](#_Toc485)

[4.2 解决了的特殊问题 14](#_Toc28122)

**1.项目分析**

1.1 项目背景

现实生活中，一栋楼需要有多部电梯同时运行，为了使其提高效率又能满足人们的需求，他们的运行通常是相互独立而又有关联的，与操作系统中线程的思想比较类似。因此，本项目通过多线程的思想，编写程序模拟现实中的电梯调度。

1.2 项目要求

①某一层楼20层，有五部互联的电梯。

②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。

③有数码显示器指示当前电梯状态。

④每层楼、每部电梯门口，有上行、下行按钮、数码显示。

⑤五部电梯相互联结，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。

⑥电梯调度算法：所有电梯初始状态都在第一层；每个电梯没有相应请求情况下，则应该在原地保持不动；电梯调度算法自行设计。

1.3 项目需求分析

对于电梯调度模拟系统，需考虑以下需求：

**·正确性**

程序应当能够在设计者能力范围内尽可能模仿现实中的电梯调度。尽可能正确的模拟出电梯的运行情况。

**·健壮性**

程序应当对电梯出现警报的特殊情况进行合适的处理。

**·交互性**

程序要以系统的形式出现，涉及到按按键等功能，需要给用户提供足够的提示与说明，使用方式要友好。

**2.项目设计**

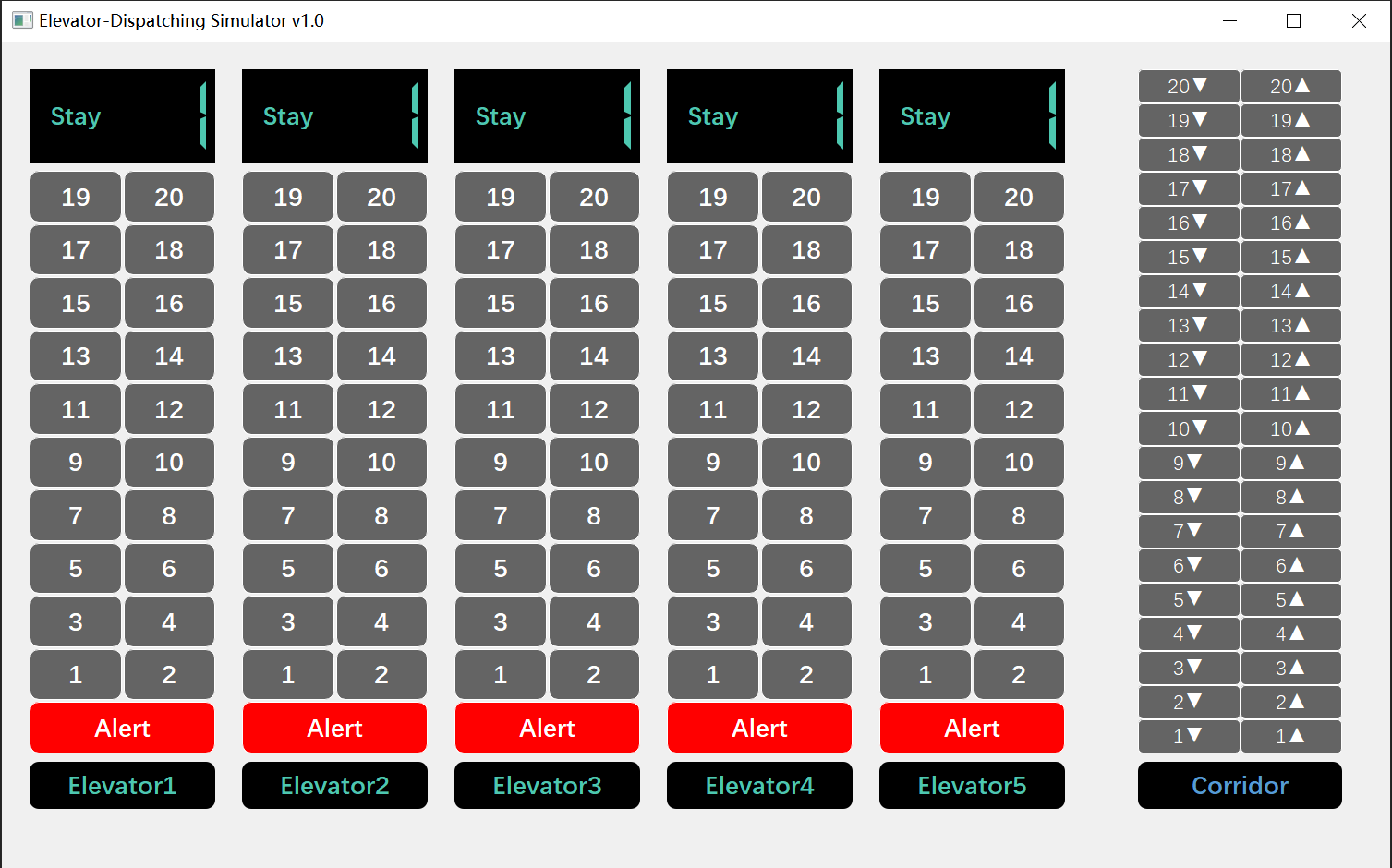
2.1 开发环境选择

本项目的开发环境：

操作系统：Windows10 编程语言：python 3.9 界面设计：PyQt5

2.2 界面设计

本项目模拟了五台电梯，每台电梯有20个数字键、一个报警键、一个电梯名称的标签、一个表示电梯状态的标签和一个显示电梯当前楼层的数码显示管。楼道内每层楼设有上下两个按键。由于数码显示管在每层楼的显示都是一致的，每一层楼的上键、下键分别状态一致，为避免冗余，以上部件仅设计一个。



2.3 调度算法设计

**2.3.1 算法思路**

给每一步电梯分配一个线程，用数组、集合等结构存储当前个电梯的状态，所有线程对部分信息进行共享。维护方面，用永真循环进行，分特殊情况和常规维护，特殊情况单独处理，常规维护每一次循环都在进行，使得电梯状态有现实意义、满足需求。

**2.3.2 算法实现**

**2.3.2.1 信息存储**

**elevator\_goal：list[set]** 用来存储五部电梯的目标楼层，下标代表电梯编号。

**should\_open:list[bool]** 用来标志五部电梯是否需要开门，由于开门是一种特殊情况，且运行中多种情况都有可能导致开门发生（且不一定是事件触发，无法使用挂接），因此使用列表进行维护。

**state:list[int]** 表示五部电梯的状态，0表示停止，1表示向上运行，-1表示向下运行。

**alert:list[bool]** 表示五部电梯是否按下了报警键

**floor:list[int]** 表示五部电梯的所在楼层

**corridor\_goal:set[int]** 表示楼道内发出的所有请求所在的楼层

代码如下：

# 各电梯目标楼层的集合  
*elevator\_goal* = []  
**for** *i* **in range**(5):  
 *elevator\_goal*.append(**set**([]))  
  
# 各电梯是否开门的标志  
*should\_open* = []  
**for** *i* **in range**(5):  
 *should\_open*.append(**False**)  
  
# 各电梯的状态  
*state* = []  
**for** *i* **in range**(5):  
 *state*.append(0)  
  
# 各电梯是否按下了报警键  
*alert* = []  
**for** *i* **in range**(5):  
 *alert*.append(**False**)  
  
# 各电梯的当前楼层  
*floor* = []  
**for** *i* **in range**(5):  
 *floor*.append(1)  
  
# 楼道里的请求集合  
*corridor\_goal* = **set**([])

**2.3.2.2 按下电梯内数字键的情况**

若当前电梯已在目标楼层，则不做任何事，否则将目标楼层加入当前电梯的目标楼层集合中。按下效果如下：

屏幕截图 2023-05-07 212734

代码如下：

**def** set\_elev\_goal(*elev\_num*, *goal\_floor*):  
 **if** *floor*[*elev\_num* - 1] == *goal\_floor*: # 若当前电梯已在目标楼层，则不做任何事  
 **return** *mainWindow*.findChild(QPushButton, f"E{*elev\_num*}F{*goal\_floor*}").setStyleSheet(*numButtonClickedStyle*) # 设置对应数字键样式  
 set\_elev\_label(*elev\_num*) # 更新状态标签样式  
 *elevator\_goal*[*elev\_num* - 1].add(*goal\_floor*) # 将目标楼层添加到当前电梯的目标队列中  
 **print**(f"{*elev\_num*}号电梯要去{*goal\_floor*}层")

**2.3.2.3 按下楼道内按键的情况**

设有人按下了goal层的按键：若有电梯elev在goal层，则elev开门，此外不做任何操作；若都没有电梯在goal层，则选出与当前楼层最近的电梯，将goal加入其目标楼层集合。

按下效果如下：

屏幕截图 2023-05-07 213040

按下下楼按键代码如下：

**def** set\_corridor\_goal\_up(*goal\_floor*):  
 **for** *elev* **in range**(5): # 检查五部电梯，若有停在当前楼层的，使其中一部开门  
 **if** *floor*[*elev*] == *goal\_floor* **and** *state*[*elev*] == 0:  
 **print**(f"电梯{*elev* + 1}已在{*goal\_floor*}层，无需分配任务")  
 *should\_open*[*elev*] = **True  
 return** *mainWindow*.findChild(QPushButton, f"Up{*goal\_floor*}").setStyleSheet(*corridorButtonClickedStyle*) # 设置对应外部按键样式  
  
 *floor\_gap* = [] # 计算五部电梯当前楼层和目标楼层的差值  
 **for** *f* **in range**(5):  
 *floor\_gap*.append(2147483647 **if** *alert*[*f*] **else abs**(*floor*[*f*] - *goal\_floor*)) # 报警状态的电梯离目标楼层无限远  
  
 *elev\_num* = *floor\_gap*.index(**min**(*floor\_gap*)) # 找出距离目标楼层最近的电梯，将该楼层添加到该电梯的目标中  
 *elevator\_goal*[*elev\_num*].add(*goal\_floor*)  
 *corridor\_goal*.add(*goal\_floor*)  
 **print**(f"{*elev\_num* + 1}号电梯被分配到{*goal\_floor*}层的任务")

按下上楼按键代码如下：

**def** set\_corridor\_goal\_down(*goal\_floor*):  
 **for** *elev* **in range**(5): # 检查五部电梯，若有停在当前楼层的，使其中一部开门  
 **if** *floor*[*elev*] == *goal\_floor* **and** *state*[*elev*] == 0:  
 **print**(f"电梯{*elev* + 1}已在{*goal\_floor*}层，无需分配任务")  
 *should\_open*[*elev*] = **True  
 return** *mainWindow*.findChild(QPushButton, f"Down{*goal\_floor*}").setStyleSheet(*corridorButtonClickedStyle*) # 设置对应外部按键样式  
  
 *floor\_gap* = [] # 计算五部电梯当前楼层和目标楼层的差值  
 **for** *f* **in range**(5):  
 *floor\_gap*.append(2147483647 **if** *alert*[*f*] **else abs**(*floor*[*f*] - *goal\_floor*)) # 报警状态的电梯离目标楼层无限远  
  
 *elev\_num* = *floor\_gap*.index(**min**(*floor\_gap*)) # 找出距离目标楼层最近的电梯，将该楼层添加到该电梯的目标中  
 *elevator\_goal*[*elev\_num*].add(*goal\_floor*)  
 *corridor\_goal*.add(*goal\_floor*)  
 **print**(f"{*elev\_num* + 1}号电梯被分配到{*goal\_floor*}层的任务")

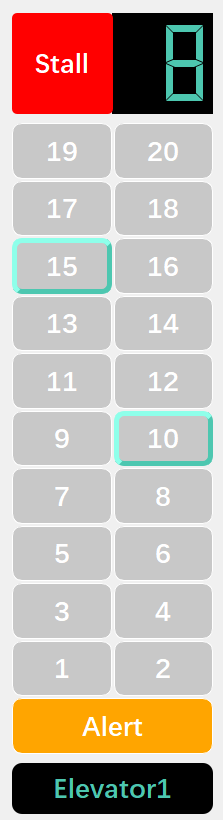
**2.3.2.4 按下报警键的情况**

设电梯elev的报警键被按下了，显然如果elev有目标楼层，则一定来源于两个方面：**电梯内的数字键、楼道内的请求**。

若报警键原先没有按下，则检查elev的每一个目标楼层的来源，若它**来源于楼道内的请求**，则将该目标分配给其他电梯（原则和2.3.3.3中相同，不同点是计算与目标楼层的距离时，按下报警键的电梯对应的距离为无穷大）。来源于楼道内的请求分配给其他电梯后，若elev的目标楼层为空，则将其状态设为静止。

若报警键原先已被按下，则恢复先前的状态。若elev的目标楼层不为空，则一定**来源于电梯内的数字键**，这是电梯内部的目标。电梯会继续完成这些目标。

按下效果如下：



代码如下：

**def** elev\_alert(*elev\_num*):  
 # 找到对应电梯的报警键和状态标签  
 *button* = *mainWindow*.findChild(QPushButton, f"E{*elev\_num*}Alert")  
 *label* = *mainWindow*.findChild(QLabel, f"ElevState{*elev\_num*}")  
  
 # 未处在报警状态，则设置为报警状态并停止  
 **if not** *alert*[*elev\_num* - 1]:  
 *alert*[*elev\_num* - 1] = **True** # 报警状态标志设为真  
 **for** *i* **in range**(20):  
 *mainWindow*.findChild(QPushButton, f"E{*elev\_num*}F{*i* + 1}").setDisabled(**True**) # 将数字键设为Disabled状态  
 *button*.setStyleSheet(*alertButtonClickedStyle*) # 设置报警键样式  
 *label*.setFont(*fontBig*) # 设置状态标签字体  
 *label*.setStyleSheet(*stateAlertLabelStyle*) # 设置状态标签样式  
 *label*.setText("Stall") # 设置状态标签文本  
 **print**(f"\n{*elev\_num*}号电梯发生异常！\n"  
 f"正在检索其目标楼层......{*elevator\_goal*[*elev\_num* - 1]}\n"  
 f"正在检索当前楼道目标......{*corridor\_goal*}")  
  
 *discard\_level* = **set**([]) # 存放要分配给其他电梯的目标楼层  
 **for** *goal* **in** *elevator\_goal*[*elev\_num* - 1]: # 检查当前电梯所有目标楼层，若其来源于楼道内请求，则分配给其他电梯  
 **if** *goal* **in** *corridor\_goal*:  
 *floor\_gap* = [] # 计算五部电梯当前楼层和目标楼层的差值  
 **for** *f* **in range**(5):  
 *floor\_gap*.append(2147483647 **if** *alert*[*f*] **else abs**(*floor*[*f*] - *goal*)) # 报警状态的电梯离目标楼层无限远  
  
 *new\_elev* = *floor\_gap*.index(**min**(*floor\_gap*)) # 找出距离目标楼层最近的电梯，将该楼层添加到该电梯的目标中  
 *discard\_level*.add(*goal*)  
 *elevator\_goal*[*new\_elev*].add(*goal*)  
 **print**(f"{*elev\_num*}号电梯本要去{*goal*}楼的任务因中断而分配给了{*new\_elev* + 1}号电梯")  
 **print**()  
  
 **for** *level* **in** *discard\_level*: # 将要分配给其他电梯的目标楼层从当前电梯的目标楼层集合中删除  
 *elevator\_goal*[*elev\_num* - 1].discard(*level*)  
 **if len**(*elevator\_goal*[*elev\_num* - 1]) == 0: # 若删完后当前电梯没有目标楼层了，则运行状态设为静止  
 *state*[*elev\_num* - 1] = 0  
  
 # 处在报警状态，则设置为未报警状态并解除停止  
 **else**:  
 *alert*[*elev\_num* - 1] = **False** # 报警状态标志设为假  
 **for** *i* **in range**(20):  
 *mainWindow*.findChild(QPushButton, f"E{*elev\_num*}F{*i* + 1}").setDisabled(**False**) # 将数字键设为Enabled状态  
 *button*.setStyleSheet(*alertButtonStyle*) # 设置报警键样式  
 *label*.setStyleSheet(*stateLabelStyle*) # 设置状态标签样式  
 set\_elev\_label(*elev\_num*) # 更新状态标签样式  
 **print**(f"{*elev\_num*}号电梯已解除报警状态")

**2.3.2.5 线程设计**

为了配合PyQt界面实现多线程操作，本项目使用QThread类进行线程设计。

成员有：电梯编号（int）、信号对象（pyqtSignal(int)）。

初始化时接收一个int型参数，即电梯的编号，同时将信号对象与常规维护的槽函数连接。

运行时处在一个永真循环中，每一秒进行一次循环。首先访问should\_open列表和alert列表检查电梯是否要开门，若是则完成开门的动作并将对应楼层的所有按键熄灭。其次将电梯编号作为参数发射至槽函数进行常规维护。

线程设计代码如下：

**class** ElevThread(QThread):  
 signal = pyqtSignal(**int**) # 信号对象实例化，带一个整型参数  
  
 **def \_\_init\_\_**(**self**, *num*):  
 **super**(ElevThread, **self**).**\_\_init\_\_**()  
 **self**.elev\_num = *num* # 接收的参数转为成员：电梯编号  
 **self**.signal.connect(update\_state) # 信号与槽函数update\_state连接  
  
 **def** run(**self**):  
 **while** 1: # 线程运行过程处在永真循环中  
 **if not** *alert*[**self**.elev\_num - 1] **and** *should\_open*[**self**.elev\_num - 1]: # 若该电梯没有报警且应当开门  
 # 找到电梯所在楼层的楼道按钮、数字键和对应电梯的状态标签  
 *up\_button* = *mainWindow*.findChild(QPushButton, f"Up{*floor*[**self**.elev\_num - 1]}")  
 *down\_button* = *mainWindow*.findChild(QPushButton, f"Down{*floor*[**self**.elev\_num - 1]}")  
 *num\_button* = *mainWindow*.findChild(QPushButton, f"E{**self**.elev\_num}F{*floor*[**self**.elev\_num - 1]}")  
 *label* = *mainWindow*.findChild(QLabel, f"ElevState{**self**.elev\_num}")  
  
 # 设置开门效果  
 *up\_button*.setStyleSheet(*corridorButtonStyle*) # 楼道上键恢复状态  
 *down\_button*.setStyleSheet(*corridorButtonStyle*) # 楼道下键恢复状态  
 *num\_button*.setStyleSheet(*numButtonStyle*) # 电梯内数字键恢复状态  
 *label*.setFont(*fontBig*) # 设置电梯标签字体  
 *label*.setText("Open") # 设置电梯标签文本  
 *label*.setStyleSheet(*stateOpenLabelStyle*) # 设置电梯标签样式  
  
 # 将开门效果延时  
 time.sleep(2)  
  
 # 消除开门效果  
 *label*.setStyleSheet(*stateLabelStyle*) # 设置电梯标签样式  
 *label*.setText("Stay") # 设置电梯标签文本  
 *should\_open*[**self**.elev\_num - 1] = **False** # 将开门标志设为假  
  
 **self**.signal.emit(**self**.elev\_num) # 信号发射参数给槽函数update\_state，通知其进行常规维护  
 time.sleep(1)

线程实例化代码如下：

*elevators* = []  
**for** *i* **in range**(5):  
 *elevators*.append(ElevThread(*i* + 1))

**2.3.2.6 常规维护槽函数设计**

若当前电梯处在报警状态，则不能进行状态的改变；否则进行以下操作：

①根据当前状态改变电梯所在楼层，静止则不操作，上升则增加一层，下降则减少一层。

②检查电梯当前楼层，若是电梯目标楼层或者是楼道内请求的楼层则开门，同时将该层楼的目标删除。

③改变电梯状态。检查电梯相关信息，设目标楼层个数为goal\_length，最大目标楼层为max\_goal，最小目标为min\_goal。若goal\_length = 0则状态设为静止。否则分三种情况讨论：若电梯向下且min\_goal大于当前楼层，则状态变为上升；若电梯向上且max\_goal小于当前楼层，则状态变为下降；若电梯静止，如果max\_goal比当前楼层高，状态设为上升，如果min\_goal比当前楼层低，状态设为下降。

第③点中的状态切换条件与现实情况符合，在电梯运行时，它可以让电梯在完成当前方向的任务后再调转运行方向去完成别的任务。

代码如下：

**def** update\_state(*elev\_num*):  
 **if not** *alert*[*elev\_num* - 1]: # 只有电梯在运行才能改变电梯的状态  
  
 # 一、改变电梯楼层  
 **if** *state*[*elev\_num* - 1] == 0:  
 **pass  
 elif** *state*[*elev\_num* - 1] == -1:  
 *floor*[*elev\_num* - 1] -= 1  
 **else**:  
 *floor*[*elev\_num* - 1] += 1  
 *mainWindow*.findChild(QLCDNumber, f"ElevLCD{*elev\_num*}").display(*floor*[*elev\_num* - 1]) # 数码显示更新  
  
 # 二、电梯状态标签更新  
 set\_elev\_label(*elev\_num*)  
  
 # 三、从各目标集合中移除该层  
 **if** (*floor*[*elev\_num* - 1] **in** *elevator\_goal*[*elev\_num* - 1]) **or** (*floor*[*elev\_num* - 1] **in** *corridor\_goal*):  
 *should\_open*[*elev\_num* - 1] = 1 # 达到目标，设开门标志为真，使线程进行特殊情况的维护  
 *corridor\_goal*.discard(*floor*[*elev\_num* - 1]) # 从楼道里的请求中删除当前楼层  
  
 *elevator\_goal*[*elev\_num* - 1].discard(*floor*[*elev\_num* - 1]) # 从电梯的目标楼层集合中删除当前楼层  
  
 # 四、改变电梯的状态  
 *goal\_length* = **len**(*elevator\_goal*[*elev\_num* - 1])  
 *max\_goal* = -1 **if** *goal\_length* == 0 **else max**(*elevator\_goal*[*elev\_num* - 1])  
 *min\_goal* = 100 **if** *goal\_length* == 0 **else min**(*elevator\_goal*[*elev\_num* - 1])  
  
 # 1、如果当前状态是向下  
 **if** *state*[*elev\_num* - 1] == -1:  
 **if** *goal\_length* == 0:  
 *state*[*elev\_num* - 1] = 0 # 电梯没有目标可去，状态设为停止  
 **elif** *min\_goal* > *floor*[*elev\_num* - 1]:  
 *state*[*elev\_num* - 1] = 1 # 电梯最小的目标层数比当前层数还高，状态设为上升  
  
 # 2、如果当前状态是向上  
 **if** *state*[*elev\_num* - 1] == 1:  
 **if** *goal\_length* == 0:  
 *state*[*elev\_num* - 1] = 0 # 电梯没有目标可去，状态设为停止  
 **elif** *max\_goal* < *floor*[*elev\_num* - 1]:  
 *state*[*elev\_num* - 1] = -1 # 电梯最大的目标层数比当前层数还低，状态设为下降  
  
 # 3、如果当前状态是静止且有目标层数  
 **if** *state*[*elev\_num* - 1] == 0 **and** *goal\_length* != 0:  
 **if** *max\_goal* > *floor*[*elev\_num* - 1]:  
 *state*[*elev\_num* - 1] = 1 # 电梯最大的目标层数比当前层数高，状态设为上升  
 **if** *min\_goal* < *floor*[*elev\_num* - 1]:  
 *state*[*elev\_num* - 1] = -1 # 电梯最小的目标层数比当前层数低，状态设为下降

**2.3.2.7 事件的处理**

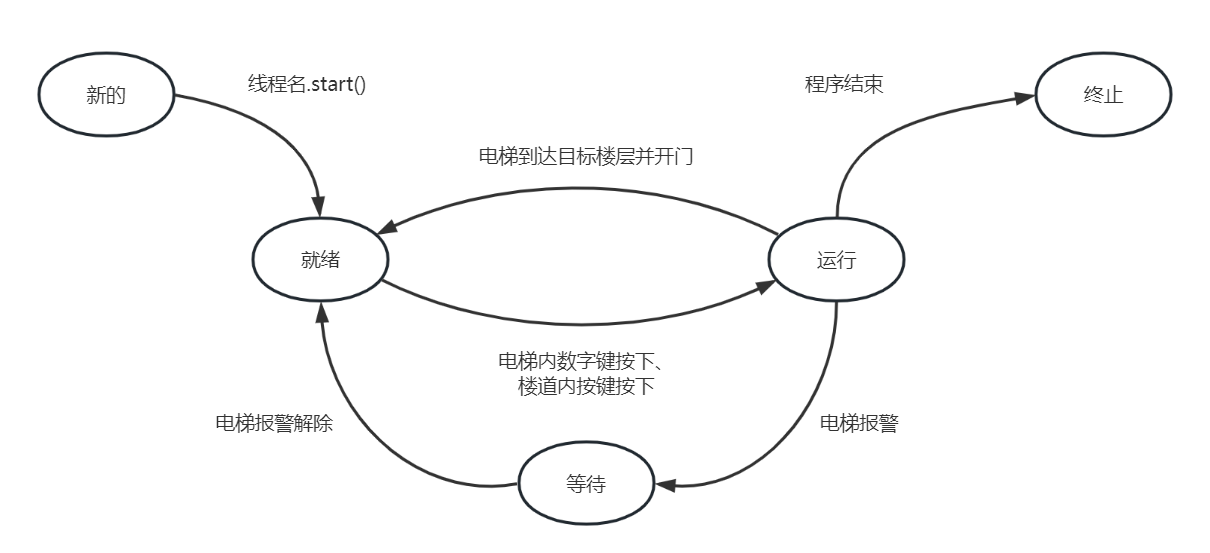
电梯内数字键的点击挂接了2.3.2.2的对应函数。

楼道内按键的点击挂接了2.3.2.3的对应函数。

报警键的点击挂接了2.3.2.4的对应函数。

**2.3.3 算法总结**

根据以上算法，线程的状态变化如下：



**3.项目测试**

****

**4.项目评价**

4.1 可改进的地方

①考虑以下情况：某部电梯内10层按键被按下，同时也被分配到了楼道内8↑、8↓的任务。

现实情况下，楼道内按键的请求是分上下楼的，到达8楼时只有8↑会熄灭，8↓却不会，且8↓的请求仍保留。而本项目中的模拟系统会将两个按键都熄灭。

②考虑以下情况：某部电梯内8层按键被按下，此后来到了8楼。根据本项目的实现，应当是电梯内8层按键、楼道内8↑、8↓都熄灭（若点亮的话）。然而有时8↓的按键熄灭后没有回到原来的效果，将鼠标悬停在8↓后又恢复正常了。

其他楼层也会用同样的bug，仔细检查代码后没有问题。推测可能是对于线程访问共享区域没有设限制而导致的错误。

③当所有的电梯都处在报警状态，则所有的任务都分配给了报警的电梯。在这个情况下除非有电梯解除报警，否则所有请求都总是无法得到满足。本项目未对该“死锁”情况进行处理。

④按键较多，界面设计上可以再改进为更简洁、友好。

4.2 解决了的特殊问题

①实现了电梯的开门效果

②实现了电梯的报警效果和报警之后的无法更改状态

③当电梯报警时，能将其当前的目标楼层分配给其他电梯，同时不允许再对这部电梯内的数字键做任何操作

④处理了电梯所在楼层和按下数字键对应楼层一致的情况

⑤处理了电梯所在楼层和楼道内请求所在楼层一致的情况