电梯算法设计

算法主要思路

1. 我们把调度情况分三种模式，分别是上班模式，下班模式和随机模式，然后统计0层和1层开关门前后重量差的近期历史来判断处于哪种模式。每次在0或1层关门时更新历史并判断是否要切换模式，这样电梯能实时自动根据情况来选择比较好的调度策略。
2. 调度主要算法框架是

每个tick

While(changed)

{

changed=false;

1. 对于每个可用电梯i,从可响应请求中找出最好的请求(target[i]=bestFloorToResponse(i,out dir[i]);)

2. 给每个电梯与其上面找出的响应打分（估计的电梯人数和电梯到响应目标楼层的距离之和作为分数）

3. 找出分数最低的可用电梯，设置它的目标楼层为响应楼层，并把此电梯设置成不可用，changed=true;

}

1. bestFloorToResponse中，如果电梯有内部请求或已响应外部请求，则看看当前方向上到当前目标楼层前是否有可以顺路捎带的外部请求。
2. bestFloorToResponse中，如果电梯空闲，由模式决定调度：
3. 如果电梯是空闲的，在上班模式，如果0层或1层有请求时，我们会轮流让2层以上的空闲电梯前往0和1层，避免0和1层某层被饥饿。
4. 如果电梯是空闲的，在下班模式，优先响应较远的请求，而在上班模式和随机模式应该优先响应较近的请求（上班模式下没有0层或1层的请求时）。

在给定的Passenger类中，乘客是根据历史方向决定是否进入电梯的，那么如果一个楼层同时有向上和向下的请求，只能服务与电梯被调度过来的方向一致的请求，这是不合理的。因此我们在Elevator类中开了一个口setCurrentDirection，在电梯开门时修改当前和历史方向，以便让调度器自由决定服务哪个方向的请求。

为便于测试，我们还用C和ncurses库写了一个可视化工具，将Debug输出管道到可视化工具，就能看到电梯动态。还可以调速。（为便于解析，稍微修改了Debug输出格式）

