Elevator System Design

课程名： 操作系统

授课老师：王冬青

设计者： 杜书杨

学号：1452981

设计时间：2017年4月22日－2017年5月2日

1. 项目描述及需求

基本任务：

某一栋楼有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。

具体目标：

所有电梯初始均处于第一层，在没有命令时均处于静止状态。设计的电梯系统除1层和20层外每层都具有上下按钮（1层只有“上”按钮，20层只有“下”按钮），每部电梯内也有1-20的数字按钮，以及一个报警按钮，按动后将使该电梯停止运行并处于故障状态无法接受任务。开关门按钮因为与实际进程无关故省去，改为在停留的楼层电梯变色来代表开关门上下客。因为界面可以直观显示出电梯运行的情况，故楼层显示也被省略。最后还加入一个恢复运行按钮，用于将设置故障状态的电梯恢复正常状态待命。

2. 解决方案

开发环境：

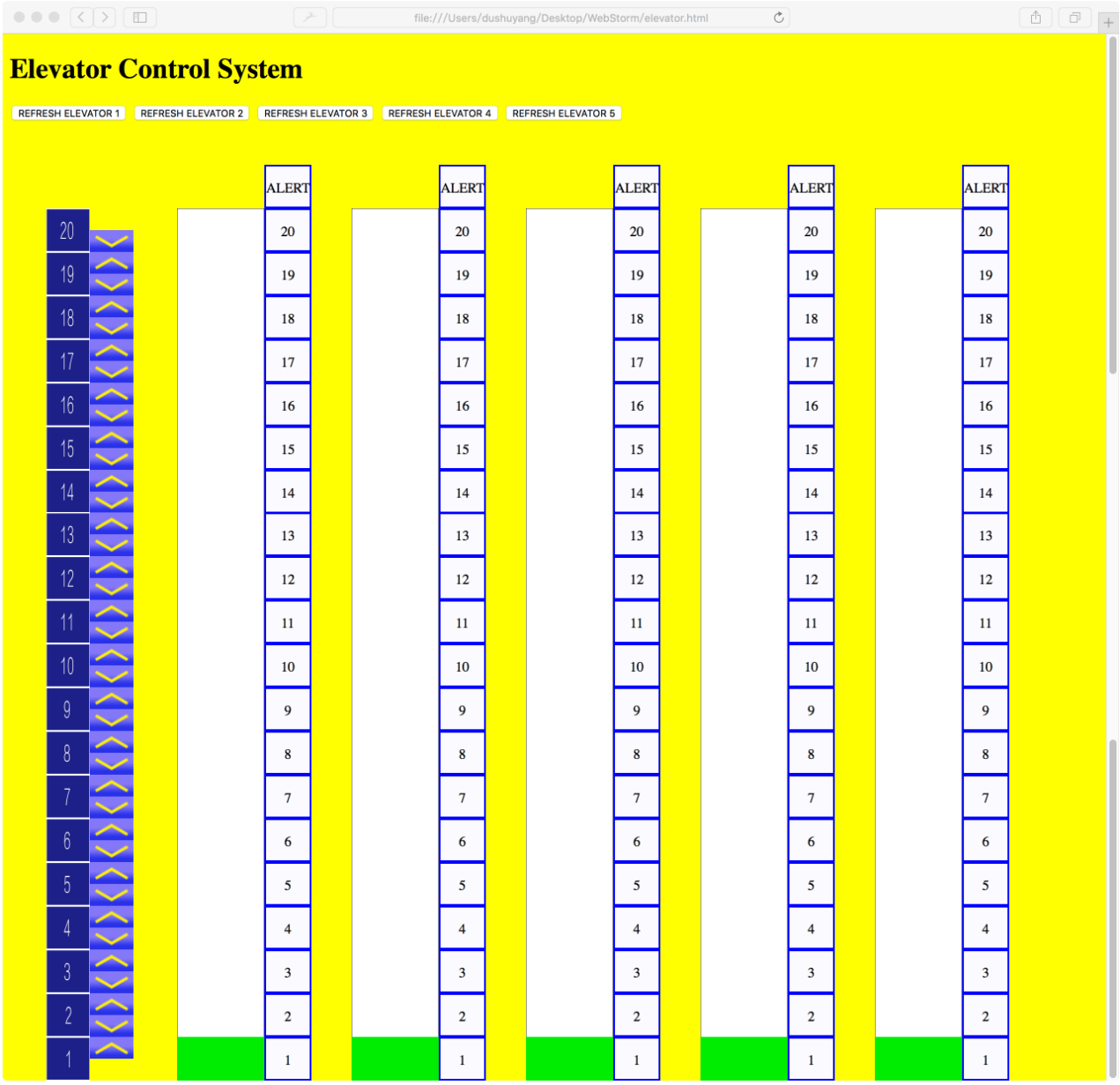
开发工具：WebStorm

开发语言：HTML5 & JavaScript

辅助工具：Adobe Photoshop 用于画按键和一些基本元素，iWork Pages 文档

成品：点击html文件运行

项目大纲：



基本界面如上图所示（由于开始制作界面时没有正确估计大小，导致元素尺寸偏大，使得最终界面长度超过屏幕，可以看到右侧有滚动条，上图为两张截图拼在一起的样子，真实界面需通过滚动条浏览）

界面部分使用HTML5编写，白色的电梯井、左侧的数字和上下按钮为插入的图片（PS制作），电梯中的数字按钮和电梯为自己编写的div块。

为了更直观体现电梯的各个状态，为每个状态都设置了特殊的颜色。电梯空闲时为绿色#00EC00；电梯运行时为浅蓝色#ACD6FF；电梯运行中停靠楼层时为深蓝色#0000C6；电梯处于故障状态时为红色#FF0000。

电梯的运行速度是0.5秒 / 层，停靠楼层的时间为1秒。通过setTimeout函数来实现。

左侧各层的上下按钮为插入的图片，其onclick将调用js文件中的work函数，work函数将调用find函数来寻找空闲的电梯执行任务，若没有空闲电梯则寻找是否有满足顺路条件的电梯，加入其任务列表target_n数组，若没有电梯符合条件则加入公共任务列表mission数组。

每个电梯都有自己的数字按钮和报警按钮。当按下数字按钮时，html将调用js中的transport函数，若此时电梯空闲，则直接执行此任务，若电梯正在执行任务，则将新任务加入该电梯的任务列表，并对任务列表重新排序，调整列表中各任务的优先级。当按下报警按钮时，电梯将停止运行并停在目前的楼层，并在1秒后变为故障状态并设为红色，删除目前该电梯任务列表中的所有任务，并且在故障状态下按该电梯楼层按钮，该电梯也不会执行任务。按下页面上部恢复该电梯按钮时，该电梯会恢复为空闲状态并变为绿色，可以正常接收任务，但不会执行之前的任务。

排序算法：

为了提高电梯运行的效率，在每次加入新任务后都要对任务列表进行重新排序。排序规则如下。若电梯处于上升状态，目前在8层，目标为18层，现依次加入如下楼层任务：17，14，20，7，16，4，8，12. 经排序后，当前目标变为12，任务列表数组变为：14，16，17，18，20，8，7，4. 若电梯处于下降状态，目前在10层，目标为4层，现依次加入如下楼层任务：17，2，20，7，16，14，8，12. 经排序后，当前目标变为8，任务列表数组变为：7，4，2，12，14，16，17，20.

相关代码如下图所示，根据电梯目前的状态，将当前任务楼层和新任务楼层对比确定优先级顺序，优先顺序低的加入任务列表，并对任务列表重新排序。

```
function add_target_1(target)                    // 给1号电梯任务列表添加新任务
{
    if((target > n1 && target < fl1 && status_1 === "up") ||        // 若新任务优先级超过当前任务,将新任务与当前任务替换
        (target < n1 && target > fl1 && status_1 === "down"))
    {
        var temp = fl1;
        fl1 = target;
        target_1[current1++] = temp;
    }
    else
    {
        target_1[current1++] = target;                // 添加新任务至任务列表
    }
    target_1 = sort(status_1, target_1, n1);          // 对任务列表重新排序
}

```

sort函数即对任务数组进行排序，将原有数组拆分为两个数组，如果电梯处在上升状态则大于目前楼层任务的加入第一数组，小于目前楼层的任务加入第二数组，使用冒泡排序（因为数组较小）对第一数组从小到大排序，对第二数组从大到小排序；如果电梯处在下降状态则小于目前楼层任务的加入第一数组，大于目前楼层的任务加入第二数组，使用冒泡排序（因为数组较小）对第一数组从大到小排序，对第二数组从小到大排序。最后讲两个数组合并，第一数组在前第二数组在后，即为新的任务列表。

任务执行：

```
function move1()                    // 1号电梯的移动函数,调用setTimeout循环执行
{
    elevator_1.style.background="#ACD6FF";          // 变为浅蓝色,表示移动
    if(n1 < fl1)                                    // 上升
    {
        status_1="up";
        elevator_1.style.top=elevator_1.offsetTop-50+"px";
        n1++;
        setTimeout("move1()", 500);
    }
    else if(n1 > fl1)                               // 下降
    {
        status_1="down";
        elevator_1.style.top=elevator_1.offsetTop+50+"px";
        n1--;
        setTimeout("move1()", 500);
    }
    else
    {
        elevator_1.style.background = "#0000C6";            // 变为深蓝色,表示停顿并上下客
        if(n1!==20){up_button[n1-1].style.border="";}
        if(n1!==1){down_button[n1-2].style.border="";}      // 该层按钮变为初始状态
        E1_button[n1].style.background="#FBFBFF";
        E1_button[n1].style.border="2px solid blue";
        if(is_1_idle())
        {
            if(is_mission_empty())
            {
                status_1="static";
                elevator_1.style.background = "#00EC00";            // 公共任务和私有任务都为空,进入闲置状态
            }
            else
            {
                fl1=get_mission();                  // 私有任务为空,处理公共任务
                setTimeout("move1()", 1000);
            }
        }
        else
        {
            fl1=get_target_1();                     // 处理下一个私有任务
            setTimeout("move1()", 1000);
        }
```

如上图所示，1号电梯在执行任务中的函数代码，其余几部电梯也遵循相同结构的函数。在每一个选择分支结束后将会继续递归调用本身，并设置0.5秒（继续运行）或1秒（停靠该楼层）延迟。若到达目标楼层，函数将恢复该楼层的按钮，并令电梯变色，停靠1秒后继续执行后续任务，若无后续任务，则执行公共任务，若无公共任务，则令电梯处于闲置状态并变为绿色，函数递归结束。

故障与恢复：

当按动alert按钮时，电梯将被设置为故障状态（前面有解释）。再按动界面顶部的reresh按钮时，电梯又将恢复正常状态，不过注意：恢复正常状态后将不会执行之前的任务，因为该电梯任务列表已清空。以下是alert和refresh事件触发函数的代码。

```
816  function alert1()                  // 1号电梯内部报警按键的触发事件
817  {
818      current1 = 0;                  // 停止运行当前任务,并清除私有任务列表中的所有任务
819      fl1 = n1;
820      for(var i=1; i<=20; i++)
821      {
822          E1_button[i].style.background="#FBFBFF";              // 内部按键全部回到初始状态
823          E1_button[i].style.border="2px solid blue";
824      }
825      for(var j=0; j<=19; j++)
826          target_1[j]=0;
827      setTimeout("status_1 = 'fault';", 1001);          // 设置为故障状态
828      setTimeout("elevator_1.style.background='#FF0000';", 1001);          // 变为红色
829  }
```

```
904  function refresh_2()               // 2号电梯恢复按钮触发事件
905  {
906      status_2 = "static";           // 设置为闲置状态
907      for(var i=1; i<=20; i++)
908      {
909          E2_button[i].style.background="#FBFBFF";              // 内部按键全部回到初始状态
910          E2_button[i].style.border="2px solid blue";
911      }
912      elevator_2.style.background="#00EC00";                  // 变为绿色
913      warn_2.style.background="#FBFBFF";
914      warn_2.style.border="2px solid blue";
915  }
```

3. 工作流程

4.22-4.25　PS制作素材，HTML写界面

4.26-4.28　JS写基本的外部回调函数，debug并使外部按钮按动后电梯能运行

4.29-4.30　加入内部数字按钮，并完成其函数，debug并完成先到先得的处理模式

4.31-5.1　调整任务优先级，写排序算法并加入报警及恢复按钮。

5.2　debug，处理报警故障及恢复的一些问题，确保任务优先级正确。

5.4-5.5　项目文档

4. 关于线程

在我的系统中，所谓的线程其实就是move函数，通过不断的递归调用本身一层一层的达到目标楼层，然后继续递归转而执行下一个任务，直到所有任务执行完毕，函数结束，该线程也结束了。而外部的上下按钮按动后的任务如果被判定能加入各个电梯的任务列表（有顺路电梯或空闲）则直接进入该电梯的线程；如果不行则加入公共任务列表，也就是等到某一部电梯做完所有任务闲置后，再执行该公共任务。所以本系统其实是优先满足电梯内按钮的要求。而alert按钮实际上就是强制终结目前的线程，并恢复初始化，refresh便是将电梯设为初始状态（除了楼层不一样），可以创立新的线程。

5. 备注&小问题

1. 为什么选择JavaScript?

原因其实有很多，一是因为寒假学了一点网页的制作，对HTML有使用经验，JS也有接触，所以会熟悉一些。而C++之前用的太多了，所以自己想要换一种语言，增加点经验。不用C#则是由于之前有同学用C#＋wpf写了5000多行还没写完，代码冗余很严重，再加上交互的unity3D项目马上也要用C#也有机会练习，所以没有用。最后由于最近要帮同学做网页前端，所以正好借此机会加深对JS的了解，为后面做铺垫。

2. 备注：因为是第一次用JS写项目，没有很多经验，也没有系统的学习只是看了网上的文字教程，所以写的代码也有很多冗余，很多本来能合在一起的函数却因为一些未知的bug只能分开来写，加大了代码量，最后JS接近1000行，希望通过以后进一步的学习能够进一步提升代码质量。

注：小bug：如果将五部电梯同时设为故障状态后（全红），再按下左侧的上下按钮，然后恢复电梯至正常状态，再给一个任务，电梯仍会执行之前的公共任务即上下的外部按钮赋予的任务。这是因为为了确保部分电梯故障时外部按钮的公共任务也能被剩余电梯执行，所以公共任务列表没有清空。其实这个bug也能解决，只是觉得不影响整体所以没有修改。还有就是如果已经点击的外部按钮连续重复点，会持续增加任务给所有电梯，请不要这么做（虽然也不会崩）；如果已经点击的内部数字按钮连续重复点则会使最后的停留时间不断的+1s（想膜就点吧）。如果在正常运行时点refresh则会出现错误（不显示任务目标楼层，但仍执行任务）。目前我就发现了这些问题，不过都不会对系统造成很大影响。

最后说一句老师助教辛苦了！

有问题可以邮箱联系我：dushuyang@126.com