201421130601小组 模拟电梯详细设计

版本号：小组简写-Details-1.0（每次修订时改变编号）

编制时间：2015年05月25日

编制人员：史柠玮

0：概述

【将概要设计中的“4.4 函数说明”内容粘帖到详细设计文档中】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 |  | 何时何条件被何模块调用？  完成什么功能？ |  |  |
| 2 |  |  |  |  |

1. 模块1——In.c

1.1 函数1

函数名称 input\_1

函数参数说明：LPVOID parameter没有用

函数功能：策略一输入用户指令

函数局部变量设计：char key 接收用户的指令

函数算法逻辑：

死循环

getchar()读取入key，fflush(stdin)清除缓冲区

为全局变量ptr的成员next申请一个结构的空间

ptr = ptr->next

将key中的指令赋给ptr的floor成员

1.2 函数2

函数名称 input\_2

函数参数说明：LPVOID parameter没有用

函数功能：策略二输入用户指令

函数局部变量设计：char key 接收用户的指令

函数算法逻辑：

死循环

getchar()读取入key，fflush(stdin)清除缓冲区

key为1-9将innerCmd数组中对应量修改为TRUE

Q-I 将upCmd数组中对应量修改为TRUE

S-L将downCmd数组中对应量修改为TRUE

2. 模块2——State.c

2.1 函数

函数名称 state\_trans

函数参数说明：void

函数功能：一个FSM，为不同状态执行不同的指令，并根据需要改变状态

函数局部变量设计：无

函数算法逻辑：

以全局变量state为准

VACANT：如果aimFloor为零

则修改flag\_1为1

否则

修改flag\_1为0

如果aimFloor > curFloor

修改state为UP

prestate 为 VACANT

否则如果aimFloor < curFloor

修改state为DOWN

prestate 为 VACANT

否则（相等）什么都不做

UP：curFloor增加1

打印出"\*\*\*\*\* Heading for %d \*\*\*\*\*\n"，curFloor

睡两秒钟

如果curFloor 和 aimFloor相等

修改state为PAUSE

prestate 为 UP

修改flag\_1为1

DOWN（与UP雷同）

PAUSE：打印出"\*\*\*\*\* Arrived %d \*\*\*\*\*\n"，curFloor

睡五秒钟

判断如果没有目标楼层

则修改state为VACANT

prestate 为 PAUSE

否则

flag\_1 = 0

判断目标楼层与当前楼层

大则UP

小则DOWN

等则什么都不做

prestate 为 PAUSE

3. 模块3——File.c

3.1 函数

函数名称 outToFile

函数参数说明：FILE \*file 传入的文件

函数功能：在每一次改变状态或者楼层改变向文件中输出一行信息

函数局部变量设计：char sste[7] 储存状态字符串

char dir[5] 储存方向字符串

int i 循环中间量

函数算法逻辑：

利用difftime(curTime,startTime)输出当前时间

用state和sste，dir输出当前状态和方向

curFloor aimFloor输出当前楼层和目标楼层

从0到MAXFLOOR-1循环判断数组upCmd downCmd innerCmd中值为TRUE的元素位置

并输出未响应电梯外向上/向下/内请求

4. 模块4——Control.c

4.1 函数

函数名称 control\_1

函数参数说明：void

函数功能：根据下一命令和当前状态改变目标楼层

函数局部变量设计：struct cmdList \*temp用来储存上一节点便于释放内存

函数算法逻辑：

首先判断如果outPtr->next == NULL且state == VACANT

则打印"The elevator is vacant."

并开始死循环直到outPtr->next != NULL

在flag\_1==1的情况下

如果outPtr->next != NULL

将outPtr赋给temp

在temp不指向head的情况下

free(temp)

如果outPtr != NULL

aimFloor = outPtr->floor

否则

aimFloor = 0

否则

aimFloor = 0

4.2 函数

函数名称 control\_2

函数参数说明：void

函数功能：根据3列指示灯和当前状态改变目标楼层

函数局部变量设计：int tempFloor = 0 暂时储存当前楼层

int flag = 0记录外部同一楼层是否上下都亮

函数算法逻辑：

以全局变量state为准

UP：tempFloor从curFloor开始向上循环到MAXFLOOR

如果upCmd[tempFloor - 1] 和innerCmd[tempFloor - 1] 任一值为 TRUE

aimFloor = tempFloor

DOWN（与UP雷同）

VACANT：如果flag\_2 == 0

打印出"The elevator is vacant."

调用outToFile(file)向文件中打印一行信息

并修改指示灯为假

downCmd[curFloor - 1] = FALSE;

innerCmd[curFloor - 1] = FALSE;

upCmd[curFloor - 1] = FALSE;

flag\_2 = 1表示已经打印过

tempFloor从1循环到MAXFLOOR

如果downCmd[tempFloor-1]， innerCmd[tempFloor-1] ，

upCmd[tempFloor-1]任一为真

aimFloor = tempFloor;

flag\_2 = 0;重置为未打印VACANT

return跳出函数

aimFloor = 0;

PAUSE： innerCmd[curFloor - 1] 内部指示灯置为假

以全局变量prestate为准

UP：如果upCmd[curFloor-1] == TRUE

flag = 1

upCmd[curFloor - 1] 向上指示灯置为假

在curFloor < MAXFLOOR的情况下

tempFloor从curFloor + 1开始向上循环到MAXFLOOR

如innerCmd[tempFloor-1] 和upCmd[tempFloor-1]有真

aimFloor = tempFloor;

return;

在curFloor > 1的情况下

与上面雷同向下循环

如果到这里还没有跳出

aimFloor = 0;

DOWN：与上面雷同

PAUSE： upCmd[curFloor - 1] = FALSE;

downCmd[curFloor - 1] = FALSE;上下全灭

aimFloor = 0;

5. 主模块——main.c

函数名称 main

函数参数说明：主函数

函数功能：将各个模块整合到一起成为一个完整的程序

函数局部变量设计：int policy储存用户键入的策略序号

函数算法逻辑：

为全局变量赋值

curFloor = 1;

state = VACANT;

ptr = &head;

outPtr =&head;

prestate = VACANT;

file = fopen("output.txt", "w");

判断fopen是否成功 若不成功perror并退出程序

fputs("当前时间\t电梯状态\t所在楼层\t目标楼层\t运行方向\t停靠时间\t未响应电梯外向上请求\t未响应电梯外向下请求\t未响应电梯内请求\n", file)向文件中输出表头

system("color 3f");更改背景颜色

打印提示信息

puts("\t\t\tWelcome to 1513-A elevator");

puts("Please select the policy you want to use:");

puts("1:First call first use;\n2:No extra effort;\n3:Fastest response;");

int policy=getchar();键盘中读取策略序号

fflush(stdin);

根据不同的policy以不同的输入函数创建线程

CreateThread(0, 0, input\_1/input\_2, NULL, 0, NULL);

startTime = time(NULL);记录时间

curTime = time(NULL);

死循环

根据policy

调用control\_n 函数

调用state\_trans()

curTime = time(NULL)记录当前时间

如果policy != '2' 或者 state != VACANT

调用 outToFile(file);