**C语言电梯模拟程序**

一直以来我对电梯很感兴趣，起初认为用C语言不可能实现电梯模拟，需要多线程的支持，因此一直以来也没有想着做。最近数据结构习题书的这道题引起了我的注意，通过几天的努力终于实现了，先将程序的实现与大家分享出来。

在这个程序关键是处理好电梯运行状态转换与乘客进出的同步进行。好在题目要求每次输入时要输入下一个乘客到来的时间，使得程序变简单了。通过一个模拟时钟，根据模拟时钟判断该运行哪个函数。以下是模拟时钟的代码。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **void** DoTime(){
2. //此函数用于模拟时钟
3. **while**(1){
4. **if**(Time>MaxTime)
5. **return**;
6. TestPeople();//两个始终都会被调用的函数
7. Controler();
8. **struct** Activity\* p=activity.next;
9. **if**(p==NULL){
10. Time=MaxTime;
11. }
12. **if**(p&&Time>=p->time){//取出活动队头的，检测定时是否到了
13. activity.next=p->next;
14. p->fn();
15. free(p);
16. }
17. Time++;
18. }
19. }

在这个先不管TestPeople()、Controler()是什么，其中activity是关键，它是一个链表在链表的头部是计时器时间最小的函数，根据模拟时钟判断是否调用这个函数以及删除这个节点。以下是Activity的具体定义。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **typedef** **struct** Activity{
2. **int** time;
3. **void**(\*fn)(**void**);
4. **struct** Activity\* next;
5. }Activity;

以及全局变量activity

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. Activity activity={0,NULL,NULL};

下面的代码用于将一个函数加入activity链表，这是根据时钟值从小到大插入activity的。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **void** AddAct(**int** time,**void**(\*fn)(**void**)){//将一个活动加入定时器，时间到了会调用这个函数
2. time=Time+time;                 //这个函数参数必须是void，返回值也必须是void
3. **struct** Activity\* act;
4. act=(**struct** Activity\*)malloc(**sizeof**(**struct** Activity));
5. act->next=NULL;
6. act->fn=fn;
7. act->time=time;
8. **struct** Activity\* p=&activity;
9. **while**(p->next!=NULL){
10. **if**(p->next->time>time)
11. **break**;
12. p=p->next;
13. }
14. act->next=p->next;
15. p->next=act;
16. }

一个简单的活动加入计时器。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **void** Input(**void**){//输入人员信息，这个需要手动调用一次，之后就根据定时器调用了
2. Person\* p = (Person\*)malloc(**sizeof**(Person));
3. **int** infloor,outfloor,giveuptime,intertime;
4. **while**(1){
5. printf("请输入用户的起始楼层:");
6. scanf("%d",&infloor);
7. printf("请输入用户的目标的楼层:");
8. scanf("%d",&outfloor);
9. printf("请输入用户的最长容忍时间:");
10. scanf("%d",&giveuptime);
11. printf("请输入下一个用户的到来时间:");
12. scanf("%d",&intertime);
13. **if**(!(infloor<0||infloor>MaxFloor-1||outfloor<0||outfloor>MaxFloor-1)&&(infloor!=outfloor))
14. **break**;
15. printf("错误的用户信息录入!\n");
16. }
17. p->Id=PersonId++;
18. p->GiveupTime=giveuptime+Time;
19. p->next=NULL;
20. p->OutFloor=outfloor;
21. **if**(outfloor>infloor)
22. CallUp[infloor]=1;
23. **else**
24. CallDown[infloor]=1;
25. AddQueue(infloor,p);
26. AddAct(intertime,Input);
27. }

这使得main函数可以写的很简单。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **int** main(){
2. Init();
3. Input();
4. DoTime();
5. **return** 0;
6. }

这样整个电梯程序大体就搭建起来了。下面看Controler()这个函数，它就是判断电梯的运行状态，然后决定电梯的下一个运行状态。AddAct()很重要，这很好的实现了一个需要时间的过程，如电梯的开门，上升，减速。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **void** Controler(**void**){
2. **if**(State==Idle||State==Stop){
3. **if**(CallUp[Floor]||CallDown[Floor]||CallCar[Floor]){
4. //当前层有请求，需要开门进出
5. **if**(CallCar[BaseFloor]==2){
6. CallCar[BaseFloor]=0;
7. State=Idle;
8. printf("现在在%d层,无人请求电梯!\n",BaseFloor);
9. **return**;
10. }
11. State=DoorOpening;
12. AddAct(DoorTime,doopendoor);
13. }
14. **else**{
15. //当前层无请求，判断其他层请求
16. **int** whitch=GetWhere();
17. **if**(whitch==GoingUp){
18. State=SpeedUp;
19. AddAct(Accelerate,domove);
20. }**else** **if**(whitch==GoingDown){
21. State=SpeedDown;
22. AddAct(Accelerate,domove);
23. }**else**{
24. State=Idle;
25. **if**(Floor!=BaseFloor)
26. AddAct(OverTime,tofirst);
27. }
28. }
29. }
30. //否则电梯忙碌
31. **return**;
32. }

下面是一些用到宏定义以及全局变量。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. #define GoingUp 1//匀速上升
2. #define GoingDown 2//匀速下降
3. #define SpeedUp 3//加速上升
4. #define SpeedDown 4//加速下降
5. #define SlowUp 5//减速上升准备停靠
6. #define SlowDown 6//减速下降准备停靠
7. #define Idle 7//空闲
8. #define Stop 8//停止且已关门
9. #define DoorOpen 9//停止且门已打开
10. #define DoorOpening 10
11. #define DoorCloseing 11
13. #define CloseTest 40    //电梯关门测试时间
14. #define OverTime  300   //电梯停候超时时间
15. #define Accelerate 15   //加速时间
16. #define UpTime  51  //上升时间
17. #define DownTime 61 //下降时间
18. #define UpDecelerate 14 //上升减速
19. #define DownDecelerate 23   //下降减速
20. #define DoorTime    20  //开门关门时间
21. #define InOutTime   25  //进出电梯时间
23. #define MaxTime 10000
24. #define MaxFloor 5
25. #define BaseFloor 1
27. //初始每层电梯等待队列和栈
28. #define Init() ({int i;\
29. **for**(i=0;i<MaxFloor;i++){\
30. Stack[i].next=NULL;\
31. Queue[i].next=NULL;\
32. }\
33. activity.next=NULL;\
34. })
35. **typedef** **struct** Person{
36. **int** Id;
37. **int** OutFloor;
38. **int** GiveupTime;
39. **struct** Person\* next;
40. }Person;
42. **typedef** **struct** Activity{
43. **int** time;
44. **void**(\*fn)(**void**);
45. **struct** Activity\* next;
46. }Activity;
48. **typedef** **struct** Person\_Ele{
49. **int** Id;
50. **struct** Person\_Ele\* next;
51. }Person\_Ele;
52. **int** Time=0;
53. **int** CallUp[MaxFloor]={0,};
54. **int** CallDown[MaxFloor]={0,};
55. **int** CallCar[MaxFloor]={0,};
56. **int** Floor=BaseFloor;
57. **int** State=Idle;
58. **int** PersonId=0;
59. Activity activity={0,NULL,NULL};
60. Person\_Ele Stack[5]={0,};
61. Person Queue[5]={0,};

下面是Controler()具体调用细节。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **void** testinout(**void**){//检测有无人进出
2. **if**(Queue[Floor].next||Stack[Floor].next)
3. AddAct(CloseTest,testinout);
4. **else**{
5. State=DoorCloseing;
6. CallUp[Floor]=0;
7. CallDown[Floor]=0;
8. CallCar[Floor]=0;
9. AddAct(DoorTime,doclosedoor);
10. }
11. }
13. **void** doclosedoor(**void**){//电梯门关了
14. printf("电梯门关了\n");
15. State=Stop;
16. }
18. **void** doopendoor(**void**){//打开电梯门
19. printf("电梯门开了\n");
20. State=DoorOpen;//门打开了
21. AddAct(CloseTest,testinout);
22. **if**(Stack[Floor].next)
23. AddAct(InOutTime,doout);
24. **else**{//没人出，就看有没有进的
25. **if**(Queue[Floor].next)
26. AddAct(InOutTime,doin);
27. }
28. }
30. **void** doout(**void**){
31. //根据栈出人，如果没有看是否有人进
32. **if**(Stack[Floor].next){
33. Person\_Ele\* p=Stack[Floor].next;
34. Stack[Floor].next=p->next;
35. ;//显示信息
36. printf("用户%d走出电梯\n",p->Id);
37. free(p);
38. }
39. **if**(Stack[Floor].next){
40. AddAct(InOutTime,doout);
41. }**else**{
42. **if**(Queue[Floor].next)
43. AddAct(InOutTime,doin);
44. }
45. }
47. **void** doin(**void**){//人进入电梯，这里不用关电梯门它会定时关的
48. Person\* p=Queue[Floor].next;
49. **if**(p){
50. Queue[Floor].next=p->next;
51. Person\_Ele\* pe=(Person\_Ele\*)malloc(**sizeof**(Person\_Ele));
52. **int** in=p->OutFloor;
53. CallCar[in]=1;//置位请求
54. pe->next=Stack[in].next;
55. pe->Id=p->Id;
56. Stack[in].next=pe;
57. printf("用户%d走入电梯\n",p->Id);
58. free(p);
59. }
60. **if**(Queue[Floor].next){
61. AddAct(InOutTime,doin);
62. }
63. }
65. **int** GetWhere(**void**){
66. **static** **int** old=0;//保存上一次电梯的方向，保证电梯尽可能在一个方向走
67. **int** isup=0,isdown=0;
68. **int** i;
69. **for**(i=Floor+1;i<MaxFloor;i++){
70. **if**(CallDown[i]||CallUp[i]||CallCar[i])
71. isup=1;
72. }
73. **for**(i=Floor-1;i>=0;i--){
74. **if**(CallDown[i]||CallUp[i]||CallCar[i])
75. isdown=1;
76. }
77. **if**(isup==0&&isdown==0){
78. **return** 0;
79. }
80. **if**(old==0){
81. **if**(isdown) old=GoingDown;
82. **if**(isup) old=GoingUp;
83. **return** old;
84. }
85. **if**(old==GoingUp&&isup)
86. **return** old;
87. **else** **if**(old==GoingDown&&isdown)
88. **return** old;
89. **else** **if**(isdown)
90. old=GoingDown;
91. **else** **if**(isup)
92. old=GoingUp;
93. **else**
94. printf("在选择方向时发生错误!\n");
95. **return** old;
96. }
98. **void** tofirst(**void**){//去第一层
99. **if**(State!=Idle||Floor==BaseFloor)
100. **return**;
101. printf("长时间没人请求电梯!将进入%d层\n",BaseFloor);
102. CallCar[BaseFloor]=2;//给电梯一个虚拟的去1层的请求，这并不会开门
103. }
105. **void** doslow(**void**){//电梯停了
106. printf("电梯停了，当前层是%d\n",Floor);
107. State=Stop;
108. }
110. **void** doup(**void**){
111. Floor++;
112. printf("电梯正在上升!现已到了%d层!\n",Floor);
113. **if**(CallDown[Floor]||CallUp[Floor]||CallCar[Floor]){
114. State=SlowUp;
115. AddAct(UpDecelerate,doslow);
116. }**else**{
117. **if**(Floor==MaxFloor-1){
118. State=SlowUp;
119. AddAct(UpDecelerate,doslow);
120. }**else**{
121. AddAct(UpTime,doup);
122. }
123. }
124. }
126. **void** dodown(**void**){
127. Floor--;
128. printf("电梯正在下降!现已到了%d层!\n",Floor);
129. **if**(CallUp[Floor]||CallDown[Floor]||CallCar[Floor]){
130. State=SlowDown;
131. AddAct(DownDecelerate,doslow);
132. }**else**{
133. **if**(Floor==0){
134. State=SlowDown;
135. AddAct(DownDecelerate,doslow);
136. }**else**{
137. AddAct(DownTime,dodown);
138. }
139. }
140. }
142. **void** domove(**void**){//加速完成，将进入正常速度
143. **if**(State==SpeedUp){
144. printf("电梯已加速上升!\n");
145. State=GoingUp;
146. AddAct(UpTime,doup);
147. }**else**{
148. printf("电梯已加速下降!\n");
149. State=GoingDown;
150. AddAct(DownTime,dodown);
151. }
152. }

他们之间尽管没有直接调用关系，但都是通过AddAct()联系起来的。下面是TestPeople()的实现，我知道DoTime()每次都调用它严重影响效率，但是这需要修改AddAct()，增加复杂性。

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. **void** TestPeople(){//这是检测每层队列是否有人放弃，有人放弃就将他踢出队列
2. **int** i;//这个函数每个时间都会被调用，效率相对较低
3. **for**(i=0;i<MaxFloor;i++){
4. Person\* p=Queue[i].next;
5. Person\* q=&Queue[i];
6. **if**(p==NULL)
7. **continue**;
8. **while**(p!=NULL){
9. **if**(p->GiveupTime<=Time){
10. **if**(Floor=i&&(State>=Idle))
11. **break**;
12. q->next=p->next;
13. printf("用户%d放弃了等待!\n",p->Id);
14. free(p);
15. p=q->next;
16. **continue**;
17. }
18. q=p;
19. p=p->next;
20. }
21. }
22. }

下面是所有的源代码：

Dianti.h

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. #ifndef \_DIANTI\_H\_
2. #define \_DIANTI\_H\_
3. #include <stdio.h>
4. #include <stdlib.h>
6. #define GoingUp 1//匀速上升
7. #define GoingDown 2//匀速下降
8. #define SpeedUp 3//加速上升
9. #define SpeedDown 4//加速下降
10. #define SlowUp 5//减速上升准备停靠
11. #define SlowDown 6//减速下降准备停靠
12. #define Idle 7//空闲
13. #define Stop 8//停止且已关门
14. #define DoorOpen 9//停止且门已打开
15. #define DoorOpening 10
16. #define DoorCloseing 11
18. #define CloseTest 40    //电梯关门测试时间
19. #define OverTime  300   //电梯停候超时时间
20. #define Accelerate 15   //加速时间
21. #define UpTime  51  //上升时间
22. #define DownTime 61 //下降时间
23. #define UpDecelerate 14 //上升减速
24. #define DownDecelerate 23   //下降减速
25. #define DoorTime    20  //开门关门时间
26. #define InOutTime   25  //进出电梯时间
28. #define MaxTime 10000
29. #define MaxFloor 5
30. #define BaseFloor 1
32. //初始每层电梯等待队列和栈
33. #define Init() ({int i;\
34. **for**(i=0;i<MaxFloor;i++){\
35. Stack[i].next=NULL;\
36. Queue[i].next=NULL;\
37. }\
38. activity.next=NULL;\
39. })
40. **typedef** **struct** Person{
41. **int** Id;
42. **int** OutFloor;
43. **int** GiveupTime;
44. **struct** Person\* next;
45. }Person;
47. **typedef** **struct** Activity{
48. **int** time;
49. **void**(\*fn)(**void**);
50. **struct** Activity\* next;
51. }Activity;
53. **typedef** **struct** Person\_Ele{
54. **int** Id;
55. **struct** Person\_Ele\* next;
56. }Person\_Ele;
58. **int** AddQueue(**int** floor,**struct** Person\* p);
59. **void** AddAct(**int** time,**void**(\*fn)(**void**));
60. **void** TestPeople();
61. **void** DoTime();
62. **void** Input(**void**);
63. //以下函数与电梯决策有关
64. **void** testinout(**void**);
65. **void** doclosedoor(**void**);
66. **void** doopendoor(**void**);
67. **void** doout(**void**);
68. **void** doin(**void**);
69. **void** doup(**void**);
70. **void** dodown(**void**);
71. **void** domove(**void**);
72. **void** doslow(**void**);
73. **void** tofirst();
74. **int** GetWhere(**void**);
75. #endif

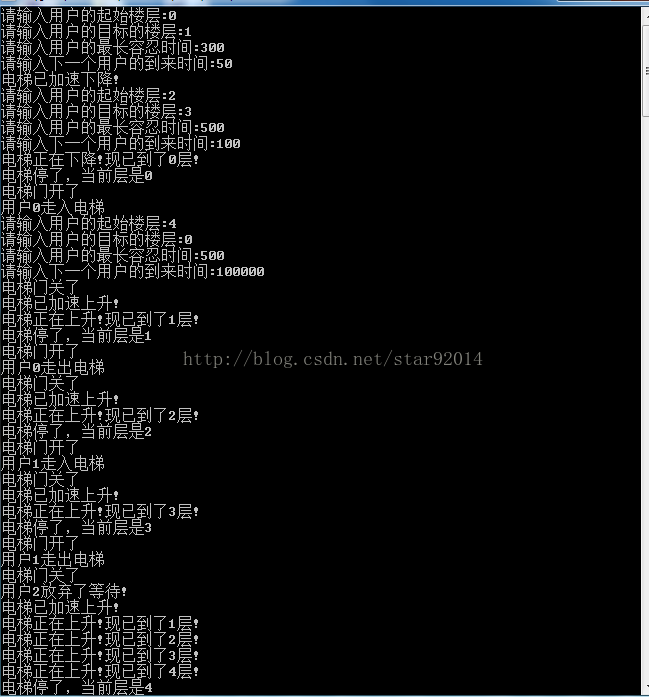
Dianti.c

**[cpp]** [view plaincopy](http://blog.csdn.net/star92014/article/details/46277919)

1. #include "Dianti.h"
3. **int** Time=0;
4. **int** CallUp[MaxFloor]={0,};
5. **int** CallDown[MaxFloor]={0,};
6. **int** CallCar[MaxFloor]={0,};
7. **int** Floor=BaseFloor;
8. **int** State=Idle;
9. **int** PersonId=0;
10. Activity activity={0,NULL,NULL};
11. Person\_Ele Stack[5]={0,};
12. Person Queue[5]={0,};
14. **int** main(){
15. Init();
16. Input();
17. DoTime();
18. **return** 0;
19. }
21. **int** AddQueue(**int** floor,Person\* p){//加入相应层的客户等待队列
22. Person\* tmp=&Queue[floor];//这始终加在链表的最后一位，
23. **while**(tmp->next!=NULL){
24. tmp=tmp->next;
25. }
26. tmp->next=p;
27. **return** 0;
28. }
30. **void** AddAct(**int** time,**void**(\*fn)(**void**)){//将一个活动加入定时器，时间到了会调用这个函数
31. time=Time+time;                 //这个函数参数必须是void，返回值也必须是void
32. **struct** Activity\* act;
33. act=(**struct** Activity\*)malloc(**sizeof**(**struct** Activity));
34. act->next=NULL;
35. act->fn=fn;
36. act->time=time;
37. **struct** Activity\* p=&activity;
38. **while**(p->next!=NULL){
39. **if**(p->next->time>time)
40. **break**;
41. p=p->next;
42. }
43. act->next=p->next;
44. p->next=act;
45. }
47. **void** TestPeople(){//这是检测每层队列是否有人放弃，有人放弃就将他踢出队列
48. **int** i;//这个函数每个时间都会被调用，效率相对较低
49. **for**(i=0;i<MaxFloor;i++){
50. Person\* p=Queue[i].next;
51. Person\* q=&Queue[i];
52. **if**(p==NULL)
53. **continue**;
54. **while**(p!=NULL){
55. **if**(p->GiveupTime<=Time){
56. **if**(Floor=i&&(State>=Idle))
57. **break**;
58. q->next=p->next;
59. printf("用户%d放弃了等待!\n",p->Id);
60. free(p);
61. p=q->next;
62. **continue**;
63. }
64. q=p;
65. p=p->next;
66. }
67. }
68. }


72. **void** Input(**void**){//输入人员信息，这个需要手动调用一次，之后就根据定时器调用了
73. Person\* p = (Person\*)malloc(**sizeof**(Person));
74. **int** infloor,outfloor,giveuptime,intertime;
75. **while**(1){
76. printf("请输入用户的起始楼层:");
77. scanf("%d",&infloor);
78. printf("请输入用户的目标的楼层:");
79. scanf("%d",&outfloor);
80. printf("请输入用户的最长容忍时间:");
81. scanf("%d",&giveuptime);
82. printf("请输入下一个用户的到来时间:");
83. scanf("%d",&intertime);
84. **if**(!(infloor<0||infloor>MaxFloor-1||outfloor<0||outfloor>MaxFloor-1)&&(infloor!=outfloor))
85. **break**;
86. printf("错误的用户信息录入!\n");
87. }
88. p->Id=PersonId++;
89. p->GiveupTime=giveuptime+Time;
90. p->next=NULL;
91. p->OutFloor=outfloor;
92. **if**(outfloor>infloor)
93. CallUp[infloor]=1;
94. **else**
95. CallDown[infloor]=1;
96. AddQueue(infloor,p);
97. AddAct(intertime,Input);
98. }
100. **void** testinout(**void**){//检测有无人进出
101. **if**(Queue[Floor].next||Stack[Floor].next)
102. AddAct(CloseTest,testinout);
103. **else**{
104. State=DoorCloseing;
105. CallUp[Floor]=0;
106. CallDown[Floor]=0;
107. CallCar[Floor]=0;
108. AddAct(DoorTime,doclosedoor);
109. }
110. }
112. **void** doclosedoor(**void**){//电梯门关了
113. printf("电梯门关了\n");
114. State=Stop;
115. }
117. **void** doopendoor(**void**){//打开电梯门
118. printf("电梯门开了\n");
119. State=DoorOpen;//门打开了
120. AddAct(CloseTest,testinout);
121. **if**(Stack[Floor].next)
122. AddAct(InOutTime,doout);
123. **else**{//没人出，就看有没有进的
124. **if**(Queue[Floor].next)
125. AddAct(InOutTime,doin);
126. }
127. }
129. **void** doout(**void**){
130. //根据栈出人，如果没有看是否有人进
131. **if**(Stack[Floor].next){
132. Person\_Ele\* p=Stack[Floor].next;
133. Stack[Floor].next=p->next;
134. ;//显示信息
135. printf("用户%d走出电梯\n",p->Id);
136. free(p);
137. }
138. **if**(Stack[Floor].next){
139. AddAct(InOutTime,doout);
140. }**else**{
141. **if**(Queue[Floor].next)
142. AddAct(InOutTime,doin);
143. }
144. }
146. **void** doin(**void**){//人进入电梯，这里不用关电梯门它会定时关的
147. Person\* p=Queue[Floor].next;
148. **if**(p){
149. Queue[Floor].next=p->next;
150. Person\_Ele\* pe=(Person\_Ele\*)malloc(**sizeof**(Person\_Ele));
151. **int** in=p->OutFloor;
152. CallCar[in]=1;//置位请求
153. pe->next=Stack[in].next;
154. pe->Id=p->Id;
155. Stack[in].next=pe;
156. printf("用户%d走入电梯\n",p->Id);
157. free(p);
158. }
159. **if**(Queue[Floor].next){
160. AddAct(InOutTime,doin);
161. }
162. }
164. **int** GetWhere(**void**){
165. **static** **int** old=0;//保存上一次电梯的方向，保证电梯尽可能在一个方向走
166. **int** isup=0,isdown=0;
167. **int** i;
168. **for**(i=Floor+1;i<MaxFloor;i++){
169. **if**(CallDown[i]||CallUp[i]||CallCar[i])
170. isup=1;
171. }
172. **for**(i=Floor-1;i>=0;i--){
173. **if**(CallDown[i]||CallUp[i]||CallCar[i])
174. isdown=1;
175. }
176. **if**(isup==0&&isdown==0){
177. **return** 0;
178. }
179. **if**(old==0){
180. **if**(isdown) old=GoingDown;
181. **if**(isup) old=GoingUp;
182. **return** old;
183. }
184. **if**(old==GoingUp&&isup)
185. **return** old;
186. **else** **if**(old==GoingDown&&isdown)
187. **return** old;
188. **else** **if**(isdown)
189. old=GoingDown;
190. **else** **if**(isup)
191. old=GoingUp;
192. **else**
193. printf("在选择方向时发生错误!\n");
194. **return** old;
195. }
197. **void** tofirst(**void**){//去第一层
198. **if**(State!=Idle||Floor==BaseFloor)
199. **return**;
200. printf("长时间没人请求电梯!将进入%d层\n",BaseFloor);
201. CallCar[BaseFloor]=2;//给电梯一个虚拟的去1层的请求，这并不会开门
202. }
204. **void** doslow(**void**){//电梯停了
205. printf("电梯停了，当前层是%d\n",Floor);
206. State=Stop;
207. }
209. **void** doup(**void**){
210. Floor++;
211. printf("电梯正在上升!现已到了%d层!\n",Floor);
212. **if**(CallDown[Floor]||CallUp[Floor]||CallCar[Floor]){
213. State=SlowUp;
214. AddAct(UpDecelerate,doslow);
215. }**else**{
216. **if**(Floor==MaxFloor-1){
217. State=SlowUp;
218. AddAct(UpDecelerate,doslow);
219. }**else**{
220. AddAct(UpTime,doup);
221. }
222. }
223. }
225. **void** dodown(**void**){
226. Floor--;
227. printf("电梯正在下降!现已到了%d层!\n",Floor);
228. **if**(CallUp[Floor]||CallDown[Floor]||CallCar[Floor]){
229. State=SlowDown;
230. AddAct(DownDecelerate,doslow);
231. }**else**{
232. **if**(Floor==0){
233. State=SlowDown;
234. AddAct(DownDecelerate,doslow);
235. }**else**{
236. AddAct(DownTime,dodown);
237. }
238. }
239. }
241. **void** domove(**void**){//加速完成，将进入正常速度
242. **if**(State==SpeedUp){
243. printf("电梯已加速上升!\n");
244. State=GoingUp;
245. AddAct(UpTime,doup);
246. }**else**{
247. printf("电梯已加速下降!\n");
248. State=GoingDown;
249. AddAct(DownTime,dodown);
250. }
251. }
253. **void** Controler(**void**){
254. **if**(State==Idle||State==Stop){
255. **if**(CallUp[Floor]||CallDown[Floor]||CallCar[Floor]){
256. //当前层有请求，需要开门进出
257. **if**(CallCar[BaseFloor]==2){
258. CallCar[BaseFloor]=0;
259. State=Idle;
260. printf("现在在%d层,无人请求电梯!\n",BaseFloor);
261. **return**;
262. }
263. State=DoorOpening;
264. AddAct(DoorTime,doopendoor);
265. }
266. **else**{
267. //当前层无请求，判断其他层请求
268. **int** whitch=GetWhere();
269. **if**(whitch==GoingUp){
270. State=SpeedUp;
271. AddAct(Accelerate,domove);
272. }**else** **if**(whitch==GoingDown){
273. State=SpeedDown;
274. AddAct(Accelerate,domove);
275. }**else**{
276. State=Idle;
277. **if**(Floor!=BaseFloor)
278. AddAct(OverTime,tofirst);
279. }
280. }
281. }
282. //否则电梯忙碌
283. **return**;
284. }
286. **void** DoTime(){
287. //此函数用于模拟时钟
288. **while**(1){
289. **if**(Time>MaxTime)
290. **return**;
291. TestPeople();//两个始终都会被调用的函数
292. Controler();
293. **struct** Activity\* p=activity.next;
294. **if**(p==NULL){
295. Time=MaxTime;
296. }
297. **if**(p&&Time>=p->time){//取出活动队头的，检测定时是否到了
298. activity.next=p->next;
299. p->fn();
300. free(p);
301. }
302. Time++;
303. }
304. }

附程序运行结果：





转自：http://blog.csdn.net/star92014/article/details/46277919