موتور تولید ماشین حالت

درود به تمامی دوستان در گروه چیپکده:

در این مقاله مروری میکنیم به استیت ماشین (ماشین حالت)یا FSM و نوعی از پیاده سازی ان‌را با زبان سی مخصوص میکرو انجام خواهیم داد.
ماشین حالت پرکاربرد ترین بلاک کد در سیستم های امبدد است.همانطور که می دونید پیاده سازی ماشین حالت مشکلاتی از قبیل پیچیدگی در طراحی و منحصر به سیستم و در مواقعی که با سیستم پیچیده روبه رو هستیم دبیاگ سخت وزمان بررا دارا می‌باشد.
نگاه بهتر طراحی موتور تولید ماشین حالت است که به طراح نرم افزار کمک کند تا در سریعترین زمان ممکن بتواند سیستمی پایدار تهیه کند. وظیفه طراح نوشتن توابع رویداد (ورودی) و توابع اکشن (خروجی) و همچنین تنظیم جدول حرکت ماشین میباشد و باقی امور پر دردسررا به موتور واگذار میکند.
بدین منظور برای کاهش مشکلات طراحی و بهبود سرعت طراحی روش ویژه ای‌را بررسی میکنیم.

اهداف:

1. ساختن موتور (کرنل یا core) برای تولید ماشین حالت
2. سبک بودن فوت پرینت کرنل
3. قابل حمل بودن کرنل
4. توانایی پشتیبانی از ماشین حالت سلسله مراتبی (hierarchical state machines)
5. اجرای موازی چندین ماشین حالت. بدون تداخل در کار یکدیگر

محیط کار:

1. سیستم عامل: xubuntu 17.4
2. کامپایلر : gcc
3. ادیتور : Vim
4. دیباگر : gdb

در ابتدا پارامتر های اساسی fsm رو تعریف میکنیم. پارامترهایی از قبیل:

1. استیت ها(state)
2. روبدادها (event)
3. توابع گذرنده(transition)
4. توابع اکشن(action)

در زیر به توضیح و پیاده سازی ان می پردازیم

رویدادها:

در این مدل event ها توسط استراکت FSME_TRANS که تبدیل به نوع داده‌ایی شده است. ساخته میشوند.

این استراکت شامل پوینتر به تابعی است که مقداری از نوع bool بر اساس این که شرط گذر اتفاق افتاده است یا نه برمیگرداند. و همچنین دارای عضوی است که استیت بعدی‌را مشخص می‌کند.

اگر تابع Event مقدار true یا یک برگرداند. ماشین به استیت بعدی گذر خواهد کرد. به تصویر زیر دقت کنید.

```
36
37 // function pointer type – for event update functions
38 typedef uint8_t (*FSME_PF_EV) (void);
39 // transition type
40 // describes a transition using the following fields:
41 // Event > a pointer to an event (input) function
42 // NextState > the next state for the transition
43 typedef struct  {
44     FSME_PF_EV Event;
45     uint8_t NextState;
46 } FSME_TRANS;
47
48
```

استیت ها:

هر استیت با استراکت FSME_STATE ساخته میشود. این استراکت شامل اطلاعاتی در مورد تابع اکشن (خروجی) به صورت فانکشن پوینتر و همچنین عضوی که نمایانگر تعداد رویداد ها از همین استیت و پوینتری به جدول گذر است.

همین جا اضافه کنم که این ماشین از نوع moore می‌باشد.

به تصویر زیر دقت کنید:

```
48
49 // function pointer type – for action (output) functions
50 typedef void (*FSME_PF) (void);
51 // state type
52 // describes a FSM state using the following fields:
53 // Action > a pointer to an action (output) function;
54 // the outputs of the FSM that are activated in this state
55 // TransNO > the number of transitions from the state
56 // Trans > an array that contains the actual transitions infromation: pairs (event, next state)
57 typedef  struct {
58     FSME_PF Action;
59     uint8_t TransNO;
60     FSME_TRANS  * Trans;
61 } FSME_STATE;
62
```

نمونه سازی:

استراکت مهم بعدی که مدل ما احتیاج داره رو میتونید تو شکل زیر ببینید. برای هر fsm که طراحی میکنید باید یک نمونه از استراکت زیررا نمونه سازی (instance) کنید. این استراکت دارای فیلدها (اعضای) یا ممبر های زیر است:

1. یک فلگ که تغییین کننده این است ایا fsm فعال یا غیر فعال است
2. یک فلگ که تایین میکند ایا استیت تغییر کرده است یا نه
3. یک فیلد که مشخص کننده استیت فعلی است
4. یک پوینتر به ارایه ایی شامل تمام رویداد ها و استیت های موجود در fsm
5. یک فیلد که برابر با طول ارایه بالا است. به تعداد استیت های موجود در ماشین
6. یک پوینتر به ارایه ایی که شامل تمام رویداد هایی است که استیت فعلی میتواند هندل کند
7. یک فیلد که برابر با با طول ارایه بالا می‌باشد. به تعداد رویداد هایی که در استیت فعلی ماشین میتواند به انها پاسخ دهد.

دو فیلد اخر برای کوتاه کردن حجم کد و همچنین بالا بردن سرعت اجرای ماشین اضافه شده است. گرچه باعث می‌شود مقدار استفاده از رم کمی بیشتر می‌شود.

در تصویر زیر کد استراکت رو میتونید ببینید:

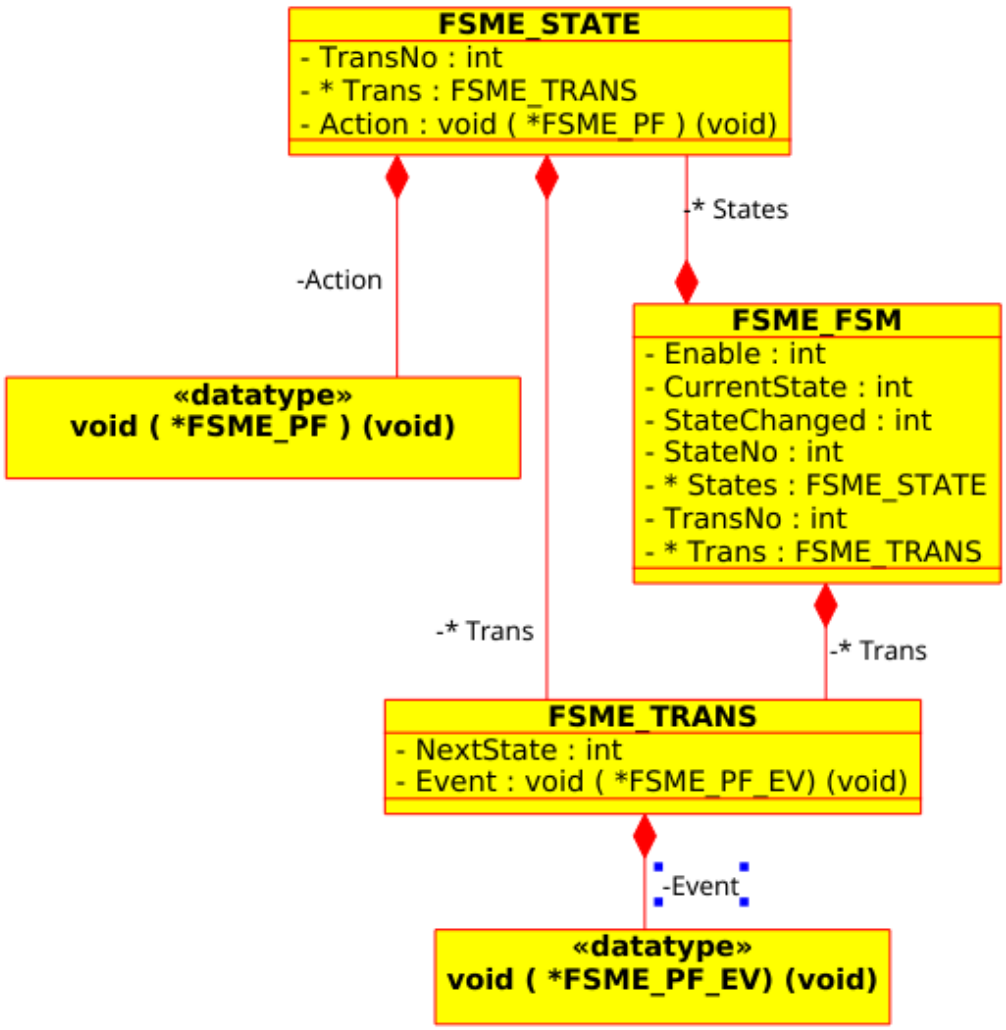
```
63
64 // FSM type
65 // describes a FSM using:
66 // Enable  > a flag that indicates the state of FSM: enabled or disabled
67 // CurrentState > the current state of the FSM
68 // StateChanged >flag that indicates a state change
69 // StatesNO > the number of states of the FSM
70 // States   > an array that contains the states and transitions of the FSM
71 // TransNO > the number of transitions from current  state
72 // Trans > a reference to an array with the transitions form current state
73 typedef struct {
74     uint8_t Enable;
75     uint8_t CurrentState;
76     uint8_t StateChanged;
77     uint8_t StatesNO;
78     FSME_STATE  * States;
79     uint8_t TransNO;
80     FSME_TRANS  * Trans;
81 } FSME_FSM;
```

ساختار هر ماشین از سه جز تشکیل میشود:

1. یک ارایه ایی از استراکت FSME_TRANS که تمام transition های ماشین در هر استیت رو مشخص میکنه.
2. یک ارایه از استراکت FSME_STATE که تمام استیت های ماشین رو مشخص میکنه به علاوه دارای پوینتری به ارایه ایی از transition ها

3. یک استراکت از نوع FSME_FSM که تمام اطلاعات ماشین رو در خودش جا داده. درواقع wrapper ایی است که ما ان‌را به توابع ارسال میکنیم. وتمام تغییرات رو در ان ثبت و نگهداری می کنیم.

در شکل زیر نمودار UML ساختار بالا رو میتونید مشاهده کنید:



توابع داخلی:

در ادامه باید توابعی بنویسیم که وظایف محول شده به ماشین رو انجام بدهد. به سادگی فقط با دو تابع این کار رو انجام میدیم.

```
115
116 static void FSME_Action( FSME_FSM * F )
117 {
118     F->States[F->CurrentState].Action();
119 }
120
```

```
83
84 static FSME_TRANS * _t;
85 static FSME_STATE * _s;
86 static void FSME_UpdateState( FSME_FSM * F )
87 {
88      uint8_t _i = 0;
89      uint8_t _n;
90 // set a variable to point to current transition table
91      _t = F->Trans;
92 // set a variable to current value of transitions count for current state
93      _n = F->TransNO;
94 // loop for all possible transitions from current state
95      for ( ; _i < _n ; _i++ )
96      {
97 // check if the events have occured (conditions are true for a transition to take place)
98           if ( FSME_EVENT_FALSE !=  _t[ _i ].Event())
99           {
100 // update current state
101              F->CurrentState = _t[ _i ].NextState;
102 // get a pointer to current state
103              _s =  & ( F->States[ F->CurrentState ]);
104 // update current transition table according to current state
105              F->Trans = _s->Trans;
106 // update current transitions number according to current state
107              F->TransNO = _s->TransNO;
108 // set state changed flag
109              F->StateChanged = FSME_STATE_CHANGED;
110 // leave the for loop and function
111              break;
112           }
113      }
114 }
115
```

اینترفیس (interface):

برنامه نویس یا طراح برای به کارانداختن ماشین تنها احتیاج به یک تابع دارد. در تصویر زیر میتونید کد این تابع رو ببنید.

```
135
136 void FSM_Run( FSME_FSM * F )
137 {
138      if ( !FSM_Get_Status(F) )
139      {
140 // TODO: may reset the FSM into initial state and deactivate outputs
141           printf("\nfsm is disable");
142           return;
143      }
144      FSME_UpdateState( F );
145      FSME_Action( F );
146 }
147
```

تابع بالا رومیشود در main لوپ بصورت polling صدا زد تا زمانی که رویداد موردنظر رخ داد ماشین خروجی مطلوب رو تولید کنه. همچنین میشود در روال اینتراپت اون رو صدا زد یا اینکه در ترکیب با توابع دیگه از اون استفاده کرد. این مسئله به نیاز نرم افزار بستگی دارد. که در زمان طراحی مشخص میشود.

همچنین ۳ تا تابع کمکی دیگه هم برای افزایش بهروری fsm مینویسیم:

```
121
122 void FSM_Status_Enable( FSME_FSM * F )
123 {
124     F->Enable = FSME_ENABLE;
125 }
126 void FSM_Status_Disable( FSME_FSM * F )
127 {
128     F->Enable = FSME_DISABLE;
129 }
130 uint8_t FSM_Get_Status( FSME_FSM * F )
131 {
132     return F->Enable;
133 }
134
```

فضای مورد نیاز:

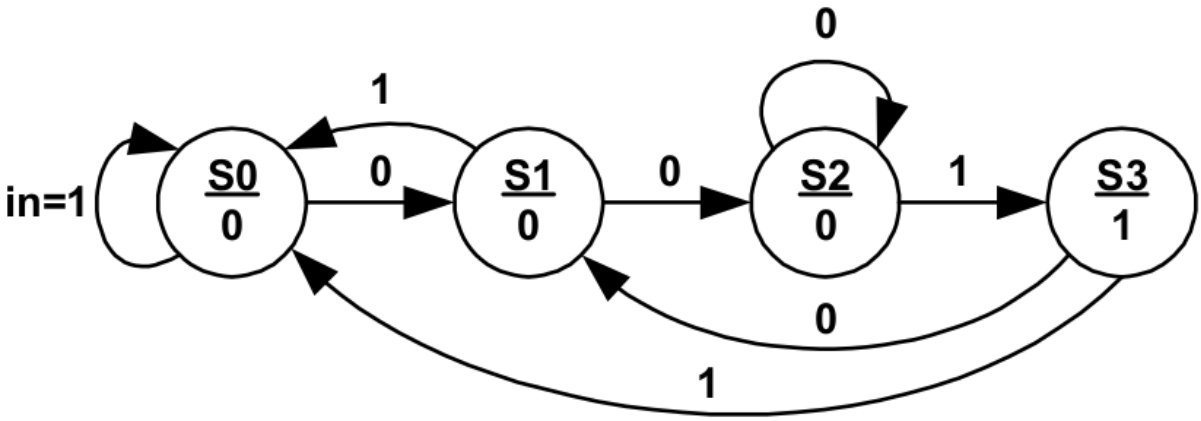
فوت پرینت این مدل ایده ال و به قرار زیر است:

1. هر استیت به ۵ بایت احتیاج دارد
2. هر رویداد ۳ بایت مصرف میکند
3. استراکت اصلی ۹ بایت میخواهد
4. طراحی موتور به ۲۳۰ بایت احتیاج دارد

sequence detector:

حالا که موتور رو طراحی کردیم لازمه که اونو تست کنیم تا از صحت عملکرد اون مطمین بشیم. برای این کار یک ترتیب یاب یا به قولی sequence detector طراحی میکنیم.

بلوک اصلی اون رو میتونید تو شکل زیر ببینید:



ماشین مورد نظر ما به ترتیب ۰۰۱ در ورودی واکنش میده و در نتیجه خروجی رو یک میکنه. درواقع اگر به ماشین استریمی از صفر و یک بدهیم. تنها زمانی خروجی ماشین یک میشود که در ورودی ترتیب ۰۰۱ ظاهر شده باشد.

در تصاویر زیر میتونید کد مورد نظر رو مشاهده کنید:

```
152
153 typedef enum {
154
155     FSM1_S0 = 0,
156     FSM1_S1,
157     FSM1_S2,
158     FSM1_S3,
159     FSM1_MAX_STATE
160
161 }FSM1_STATES;
162
163 int in, out;
164
165
166 /*
167  * inputs (events) prototype
168  */
169 static uint8_t FSM1_EventsUpdate0( void );
170 static uint8_t FSM1_EventsUpdate1( void );
171
172 /*
173  * outputs (actions) prototype
174  */
175 static void FSM1_ActionClr( void );
176 static void FSM1_ActionSet( void );
177
```

```
179
180 // state transitions for each state
181 static FSME_TRANS FSM1_S0_TRANS[] =
182 {
183     { FSM1_EventsUpdate0, FSM1_S1 }
184 };
185
186 static FSME_TRANS FSM1_S1_TRANS[] =
187 {
188     { FSM1_EventsUpdate0, FSM1_S2 },
189     { FSM1_EventsUpdate1, FSM1_S0 }
190 };
191
192 static FSME_TRANS FSM1_S2_TRANS[] =
193 {
194     { FSM1_EventsUpdate1, FSM1_S3 }
195 };
196
197 static FSME_TRANS FSM1_S3_TRANS[] =
198 {
199     { FSM1_EventsUpdate0, FSM1_S1 },
200     { FSM1_EventsUpdate1, FSM1_S0 }
201 };
202
203
204 // state outputs and transitions; entire table
205 static FSME_STATE FSM1_STATES_TABLE[] =
206 {
207     { FSM1_ActionClr, 1, FSM1_S0_TRANS },
208     { FSM1_ActionClr, 2, FSM1_S1_TRANS },
209     { FSM1_ActionClr, 1, FSM1_S2_TRANS },
210     { FSM1_ActionSet, 2, FSM1_S3_TRANS }
211 };
212
```

```
213 /*
214  * instance of FSM
215  * wrap all data which handle by FSM_Run
216  */
217 static FSME_FSM FSM1 =  {
218     .Enable = FSME_ENABLE,
219     .CurrentState = FSM1_S0,
220     .StateChanged = FSME_STATE_CHANGED,
221     .StatesNO = FSM1_MAX_STATE,
222     .States = FSM1_STATES_TABLE,
223     .TransNO = 1,
224     .Trans = FSM1_S0_TRANS
225 };
226
```

```
227
228 /*
229  * inputs and outputs (events and actions) implementation
230  */
231 static uint8_t FSM1_EventsUpdate0( void )
232 {
233     return ( in == 0 );
234 }
235 static uint8_t FSM1_EventsUpdate1( void )
236 {
237     return ( in == 1 );
238 }
239
```

```
239
240 static void FSM1_ActionClr( void )
241 {
242     if ( FSM1.StateChanged == FSME_STATE_CHANGED )
243     {
244         //actions to be executed only once in this state at the state change
245          out = 0;
246 //          printf("\n\t FSM1_ActionClr ----> state change");
247          // reset state changed flag
248          FSM1.StateChanged =
249          FSME_STATE_NOT_CHANGED;
250     }
251     else
252     {
253      // actions to be executed continuously in this state
254 //          printf("\n\t FSM1_ActionClr ----------------> executed continously");
255     }
256 }
257 static void FSM1_ActionSet( void )
258 {
259     if ( FSM1.StateChanged  ==  FSME_STATE_CHANGED )
260     {
261         // actions to be executed only once in this state at the state change
262         out = 1;
263 //          printf("\n\t FSM1_ActionSet ----> state change");
264         // reset state changed flag
265         FSM1.StateChanged =
266         FSME_STATE_NOT_CHANGED;
267     }
268     else
269     {
270         // actions to be executed continoustly in this state
271 //          printf("\n\t FSM1_ActionSet ----------------> executed continously");
272     }
273 }
274
```

```
285 */
286     int
287 main ( int argc, char *argv[] )
288 {
289
290     printf("\n  simple fsm demo \n");
291     int i=0;
292     FSM_Status_Enable(&FSM1);
293
294     printf ("\n\tin: ");
295     for (; i<100; i++){
296
297         in = rand() % 2;
298         FSM_Run(&FSM1);
299         if (out){
300             printf("%d, ", in);
301             printf ("\n\e[1;34m-->out:%d\e[0m", out);
302             printf ("\n\tin: ");
303         }
304         else
305             printf("%d, ", in);
306
307     }
308
309     return EXIT_SUCCESS;
310 }               /* ----------  end of function main  ---------- */
311
```

ورانتها خروجی برنامه:

```
 simple fsm demo

       in: 1, 0, 1, 1, 1, 1, 0, 0, 1,
-->out:1
       in: 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
-->out:1
       in: 0, 1, 1, 0, 0, 0, 1,
-->out:1
       in: 1, 1, 1, 0, 0, 0, 1,
-->out:1
       in: 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1,
-->out:1
       in: 0, 1, 0, 1, 0, 0, 1,
-->out:1
       in: 0, 0, 0, 1,
-->out:1
       in: 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
-->out:1
       in: 0, 1, 0, 0, 0, 0, 0, 1,
-->out:1
       in: 1, 0, 1, 0, 0, 0, 0, 1,
-->out:1
       in: 0, 0, 0, 0, 1,
-->out:1
```

این مقاله و کد ان بصورت رایگان منتشر میشود

مسعود ثمرین
تابستان ۹۶