1091438 郁宸瑋

程式碼說明&如何編譯

1. 標頭黨

```
1 #include <pthread.h>
2 #include <iostream>
3
4 #include <stdio.h>     //random
5 #include <time.h>      //random
6 #include <math.h>      //pow
7 #include <algorithm> // for sort
8 using namespace std;
```

為了使用 Pthread API

為了使用亂數,Ex : srand、rand()

2. 全域變數 (Global Variable)

```
10 bool aircraft[4] = {false,true,false,false};   //Dispatcher,Producer 1~3
11 bool propeller[4] = {false,false,false,false};
12 bool battery[4] = {false,false,false,false};
13 bool gate[3] = {true,true,true}; // To check every Producer has seen the platform
14 int drones[3] = {0,0,0};          //Producer 1~3
15 int item[3] ={0,0,0}; // The total number of items produced , sequence => aircraft,propeller,battery
16 int randomseed = 0;
17 int alldrone = 0; // The count of drone(S)
18
19
20 pthread_mutex_t lock;
```

bool aircraft[4]、bool propeller[4]、bool battery[4](bool 零件[4]):

用來記錄 Dispatcher 將該零件製作出,是否有在平台上面,若放置在平台上且沒有 Producer 索取該零件,**bool 零件[0] = true**。若被 Producer 取走或是尚未製作出並放置到平台上,**bool 零件[0] = false**。而 bool 零件[1~3]則是記錄 Producer1~Producer3 是否擁有該零件,若擁有該零件 = true, 沒有該零件 = false。

bool gate[3]:

用來記錄 Producer1 ~ 3 是否都有進入過 Critical Section(是否都有看過平台上有無它們缺少的零件),false 代表該 Producer 尚未看過平台。

int drones[3]:

用來紀錄 Producer1~3 分別做出了幾架無人機

int item[3]:

用來記錄各個零件(順序為:Aircraft、Propeller、Battery)被 Dispatcher 製造出的數量

int randomseed:

用來記錄第二個輸入的整數參數(初始化程式的亂數)

int alldrone:

紀錄總共造出幾架無人機

pthread_mutex_t t:

創建一個 mutex lock

3. int main(int argc, char *argv[]) &

void *Producer(void *arg) & void *Dispatcher(void *arg)

一開始先創建 4 個子執行緒, 並將輸入的整數參數存放到 randomseed 中

```
254 int main(int argc,char *argv[]){
255         //argv[1] , basic mode
256         pthread_t tid[4];
257         if(pthread_mutex_init(&lock,NULL)!=0){
258                 cout<<"mutex init failed"<<endl;
259                 return 1;
260         }
261         string str = argv[2]; //randomseed
262         int tmp2 = str.size() - 1;
263         for(int i = 0;i <= tmp2;i++){
264                 int tmp1 = str[i] - '0';
265                 randomseed = randomseed + tmp1 * pow(10,tmp2);
266                 tmp2--;
267         }
```

用來暫存第二個輸入的整數參數

將 string 型態的整數參數(第二個輸入), 轉成 int 型態並存到 randomseed 中

pthread_t tid[4]:

有一個 Dispatcher 和三個 Producer, 創造四個子執行緒

第 257~260 行: 初始化 lock, 若初始化失敗, 輸出"mutex init failed"

接下來判斷 argv[1]是否 = 0(第 269 行, 基本模式), 並開始透過 thread 製造無人機:

```
269         if(*argv[1] == '0'){
270                 //Create the thread:Producer1~3 and Dispatcher
271                 pthread_create(&(tid[0]),NULL,&Dispatcher,(void *)"Dispatcher:");
272                 pthread_create(&(tid[1]),NULL,&Producer,(void *)"Producer 1 (aircraft):");
273                 pthread_create(&(tid[2]),NULL,&Producer,(void *)"Producer 2:");
274                 pthread_create(&(tid[3]),NULL,&Producer,(void *)"Producer 3:");
275
276
277                 pthread_join(tid[0],NULL);
278                 pthread_join(tid[1],NULL);
279                 pthread_join(tid[2],NULL);
280                 pthread_join(tid[3],NULL);
```

271~274 行:

建立新 thread(有 4 個) -> Dispatcher 和 Producer(執行緒的入口函數名字), 且帶個自己的名字(Dispatcher、Producer 1…)過去, 並等待子執行緒結束 (277~280 行)

進入到子執行緒,Dispatcher(1 個)負責不斷供應 3 種模組配件到一個供應前台(Critical Section),Producer(3 個)負責不斷製造空拍機,並隨時觀看供應前台是否擁有他們缺少的零件

首先,先說明 Dispatcher 的用處

```
147 void* Dispatcher(void *arg){ // P0
148         srand((unsigned)time(NULL)^randomseed);
149         while(true){
150
151         pthread_mutex_lock(&lock);
152
153         if(alldrone >= 50){
154                 pthread_mutex_unlock(&lock);
155                 break;
156         }
157
158         if(!gate[0] || !gate[1] || !gate[2]){
159                 pthread_mutex_unlock(&lock);
160                 continue; // back to while(True)
161         }
162
163         if(!aircraft[0] && !propeller[0] && !battery[0]){
164                 int ran = rand() % 3;
165                 switch(ran){
166                         case 0:
167                                 aircraft[0] = true;
168                                 item[0]++;
169                                 cout<<(char *)arg<<" aircraft"<<endl;
170                                 break;
171                         case 1:
172                                 propeller[0] = true;
173                                 item[1]++;
174                                 cout<<(char *)arg<<" propeller"<<endl;
175                                 break;
176                         case 2:
177                                 battery[0] = true;
178                                 item[2]++;
179                                 cout<<(char *)arg<<" battery"<<endl;
180                                 break;
181                 }
182         } // All three objects are off the platform
```

初始化程式的亂數

若製作的無人機總數(alldrone)>=50,將 lock 釋放並跳出迴圈<while(true)>

若搶到 Critical Section 使用權的是 Dispatcher,但仍有 Producer 未進入過 CS,則將鎖釋放並重新搶奪 CS 使用權

此處為供應前台沒有全部的模組配件時,Dispatcher 會隨機供應

149 行: 當製作出的無人機數量<50 時,Dispatcher 會持續供應各個零件

第 151 行:
因 153~246 行內均為 Thead 間共用的資料,為了避免 Race condition,將 153~246 行利用 pthread_mutex_lock(&lock)和 pthread_mutex_unlock(&lock)鎖起來,形成 Critical Section,使同一個時間內,只有一個 thread 能更改共用的資料。

第 163~242 行: 為了滿足題目要求
題目: 當供應前台沒有全部的模組配件時,dispatcher 會隨機供應。但 dispatcher 具有智慧判斷系統,因此供應時不會重複提供供應前台已經有的模組配件。

第 163~182 行:
我的設計是:
ran = 0 時,供應 aircraft , ran = 1 時,供應 propeller , ran = 2 時,供應 battery
供應至平台後,將該配件[0] = true(代表平台上有此物件),並將該配件的生產總數+1(item)和輸出(Dispatcher 供應了哪個配件)

```
183         else if(aircraft[0] && !propeller[0] && !battery[0]){
184                 int ran = rand() % 2;
185                 switch(ran){
186                         case 0:
187                                 propeller[0] = true;
188                                 item[1]++;
189                                 cout<<(char *)arg<<" propeller"<<endl;
190                                 break;
191                         case 1:
192                                 battery[0] = true;
193                                 item[2]++;
194                                 cout<<(char *)arg<<" battery"<<endl;
195                                 break;
196                 }
197         }// Platform have aircraft
198         else if(!aircraft[0] && propeller[0] && !battery[0]){
199                 int ran = rand() % 2;
200                 switch(ran){
201                         case 0:
202                                 aircraft[0] = true;
203                                 item[0]++;
204                                 cout<<(char *)arg<<" aircraft"<<endl;
205                                 break;
206                         case 1:
207                                 battery[0] = true;
208                                 item[2]++;
209                                 cout<<(char *)arg<<" battery"<<endl;
210                                 break;
211                 }
212         }// Platform have propeller
213         else if(!aircraft[0] && !propeller[0] && battery[0]){
214                 int ran = rand() % 2;
215                 switch(ran){
216                         case 0:
217                                 aircraft[0] = true;
218                                 item[0]++;
219                                 cout<<(char *)arg<<" aircraft"<<endl;
220                                 break;
221                         case 1:
222                                 propeller[0] = true;
223                                 item[1]++;
224                                 cout<<(char *)arg<<" propeller"<<endl;
225                                 break;
226                 }
227         }// Platform have battery
```

183~227 行:

當供應平台上只有一個模組配件時,Dispatcher 會供應另外兩個模組配件的其中一個,作法與 163~182 行一樣

第 183~197 行:

供應平台上只有 aircraft 時,選擇供應 propeller 或 battery 其中一個

我的設計是:若 ran = 0 時,供應 propeller , ran = 1 時,供應 battery

第 198~212 行:

供應平台上只有 propeller 時,選擇供應 aircraft 或 battery 其中一個

我的設計是:若 ran = 0 時,供應 aircraft , ran = 1 時,供應 battery

第 183~197 行:

供應平台上只有 battery 時,選擇供應 aircraft 或 propeller 其中一個

我的設計是:若 ran = 0 時,供應 aircraft , ran = 1 時,供應 propeller

```
228         else if(!aircraft[0] && propeller[0] && battery[0]){
229                 aircraft[0] = true;
230                 item[0]++;
231                 cout<<(char *)arg<<" aircraft"<<endl;
232         }// Platform have propeller and battery
233         else if(aircraft[0] && !propeller[0] && battery[0]){
234                 propeller[0] = true;
235                 item[1]++;
236                 cout<<(char *)arg<<" propeller"<<endl;
237         }// Platform have aircraft and battery
238         else if(aircraft[0] && propeller[0] && !battery[0]){
239                 battery[0] = true;
240                 item[2]++;
241                 cout<<(char *)arg<<" battery"<<endl;
242         }// Platform have aircraft and propeller
```

第 228~242 行:

當供應平台上已有兩個模組配件時,Dispatcher 會供應剩下的配件(未在供應平台上的),並將該配件[0] = true(台上有該物件),並將該配件的生產總數+1(item)和輸出(Dispatcher 製造了哪個配件) <均與 163-182 行一樣>

```
244         gate[0] = false;
245         gate[1] = false;
246         gate[2] = false;
247
248         pthread_mutex_unlock(&lock);
249
250         }
251         return NULL;
252 }
```

Dispatcher 供應一個配件後(或是不供應,供應前台**滿了**),將 gate[0~2] = false(代表 Producer1~3 尚未查看供應平台上是否有它們缺的配件),並將 lock 釋放,離開 CS

再來,說明 Producer(1~3)的用處

```
22 void* Producer(void *arg){   //P1
23         while(true){
24
25         pthread_mutex_lock(&lock);
26
27         if(alldrone >= 50){
28                 pthread_mutex_unlock(&lock);
29                 break;
30         }
31
32         string str = (string)(char *)arg;
33
34         int count;
35         if(str == "Producer 1 (aircraft):"){
36                 count = 1;
37         }
38         else if(str == "Producer 2:"){
39                 count = 2;
40         }
41         else if(str == "Producer 3:"){
42                 count = 3;
43         }
44         gate[count-1] = true;
45         if(aircraft[0] && !propeller[0] && !battery[0]){
46                 if(!aircraft[count] && count != 1){
47                                 aircraft[count] = true;
48                                 aircraft[0] = false;
49                                 cout<<str<<" get aircraft"<<endl;
50                 }
51
52
53         }// Platform have aircraft,Producer 1 have aircraft forever
54         else if(!aircraft[0] && propeller[0] && !battery[0]){
55                 if(!propeller[count]){
56                                 propeller[count] = true;
57                                 propeller[0] = false;
58                                 cout<<str<<" get propeller"<<endl;
59                 }
60
61         }// Platform have propeller
62         else if(!aircraft[0] && !propeller[0] && battery[0]){
63                 if(!battery[count]){
64                                 battery[count] = true;
65                                 battery[0] = false;
66                                 cout<<str<<" get battery"<<endl;
67                 }
68
69         }// Platform have battery
```

利用 str,判斷目前進入 CS 的是哪一個 Produce,若為 Producer 1,將 count = 1,count 是**用來判斷現在是哪一個 Producer**

第 25 行:
因 26~139 行內均為 Thead 間共用的資料,與 153~246 行的功用一樣(第三頁)

第 44 行:
若此 Producer 已經看過供應平台,將 gate[count-1](Producer1~3 為 gate[0~2],0~2 均為 count-1) = true

第 45~69 行:
供應平台上只有一個模組配件,先透過配件[count]判斷此 Producer 是否已擁有該配件(第 46,55,63 行),若未擁有,將平台上的配件取走(配件[0] = false),且將配件[count] = true(代表此 Producer 已擁有此配件),並輸出(哪個 Producer 拿了哪個配件)
Ps. 第 46 行多了一個判斷(count!=1),因為 Producer1 會先分配 aircraft(已擁有),不需要像供應平台索取 aircraft

```
70          else if(!aircraft[0] && propeller[0] && battery[0]){
71                  if(!propeller[count]){
72                          propeller[count] = true;
73                          propeller[0] = false;
74                          cout<<str<<" get propeller"<<endl;
75                  }
76                  else if (!battery[count]){
77                          battery[count] = true;
78                          battery[0] = false;
79                          cout<<str<<" get battery"<<endl;
80                  }
81
82          }// Platform have propeller and battery
83          else if(aircraft[0] && !propeller[0] && battery[0]){
84                  if(!aircraft[count] && count!=1){
85                          aircraft[count] = true;
86                          aircraft[0] = false;
87                          cout<<str<<" get aircraft"<<endl;
88                  }
89                  else if (!battery[count]){
90                          battery[count] = true;
91                          battery[0] = false;
92                          cout<<str<<" get battery"<<endl;
93                  }
94
95          }// Platform have aircraft and battery
96          else if(aircraft[0] && propeller[0] && !battery[0]){
97
98                  if(!aircraft[count] && count!=1){
99                          aircraft[count] = true;
100                         aircraft[0] = false;
101                         cout<<str<<" get aircraft"<<endl;
102                 }
103                 else if (!propeller[count]){
104                         propeller[count] = true;
105                         propeller[0] = false;
106                         cout<<str<<" get propeller"<<endl;
107                 }
108
109         }// Platform have aircraft and propeller
```

第 70~109 行:
供應平台已有兩個配件,判斷與輸出方式與 45~69 行相同

```
110         else if(aircraft[0] && propeller[0] && battery[0]){
111                 if(!aircraft[count] && count!=1){
112                         aircraft[count] = true;
113                         aircraft[0] = false;
114                         cout<<str<<" get aircraft"<<endl;
115                 }
116                 else if (!propeller[count]){
117                         propeller[count] = true;
118                         propeller[0] = false;
119                         cout<<str<<" get propeller"<<endl;
120                 }
121                 else if (!battery[count]){
122                         battery[2] = true;
123                         battery[0] = false;
124                         cout<<str<<" get battery"<<endl;
125                 }
126
127
128         }// Platform have all items
```

第 110~128 行:

供應平台擁有所有(三個)配件, 判斷與輸出方式與 45~69 行相同

```
130         if(aircraft[count] && propeller[count] && battery[count]){
131                 if(count!=1){
132                     aircraft[count] = false;
133                 }
134                 propeller[count]  = false;
135                 battery[count] = false;
136                 drones[--count]++;
137                 alldrone++;
138                 cout<<str<<" OK, "<<drones[count]<<" drone(s)"<<endl;
139         }
140         pthread_mutex_unlock(&lock);
141
142         }
143         return NULL;
144
145 }
```

第 130~140 行:

若該 Producer 已擁有三個配件, 即可製作無人機(將該 Producer 的所有配件[count] = false, 製作一台無人機會將三個配件消耗掉<除了 Producer1 的 aircraft, Producer1 已升級過, 會一直擁有 aircraft, 第 131 行的判斷>), 並記錄該 Producer 總共生產了幾個無人機(drones[--count]++), 且所有以生產出的無人機數量+1(alldrone++), 最後輸出、將 lock 釋放

製作好 50 架無人機後,回到 main,輸出最終結果

```
282             while(true){
283                     if(alldrone >= 50){
284                             cout<<"50 drones have been completed!!"<<endl<<endl;
285                             cout<<"How many different module accessories are Dispatcher prepared ?"<<endl;
286                             cout<<"Aircraft: "<<item[0]<<", Propeller: "<<item[1]<<" , Battery: "<<item[2]<<endl;
287                             int compare[3] = {drones[0],drones[1],drones[2]};
288                             sort(compare,compare+3,greater<int>());  //sorting,from big to small
289                             int output = 0;
290                             bool outputjudge[3] = {false,false,false};
291                             while(output != 3){
292                                     for(int i = 0; i < 3; i++){
293                                             if(drones[i] == compare[output] && outputjudge[i] == false){
294                                                     cout<<"Producer "<< i+1;
295                                                     if(i == 0){
296                                                             cout<<" (aircraft): ";
297                                                     }
298                                                     else{
299                                                             cout<<": ";
300                                                     }
301                                                     cout<<drones[i]<<" drone(s)"<<endl;
302                                                     outputjudge[i] = true;
303                                             }
304
305                                     }
306                                     output++;
307                             }
308                             break;
309                     }
310             }
311             pthread_mutex_destroy(&lock);
312     }
313     else{
314             cout<<"Advanced mode are not made"<<endl;
315     }
316     return 0;
317 }
```

將 lock 銷毀

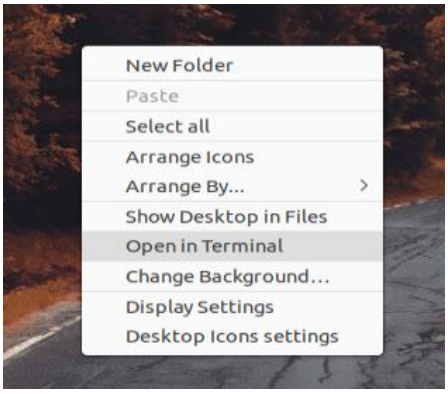
若 argv[1] = 1(進階功能),輸出(我並未做進階功能)

第 287~307 行:

我的設計是:先將 3 個 Producer 生產的無人機總數放入 commpare[3],並透過 sort()進行排序(**由大排到小**),output 是代表目前輸出的是第 count 個的數字(誰生產的無人機越多,就先輸出),而 outputjudge 是我用來判斷哪一個 Producer 已輸出過結果(因為我第 293 行的判斷是透過"數字大小(drones[i])"有沒有與 compare[output]相同,來決定我要不要輸出, 但數字(各個 Producer 生產的無人機數量)有可能一樣,因此我多加一個 outputjudge 來判斷,避免重複輸出),輸出後將 outputjudge[i] = true(表示該 Producer 已輸出過結果)

4.如何編譯

(1)在 Ubuntu 桌面右鍵 - > 選擇 Open Terminal



(2)輸入 g++ /home/yui/Desktop/Progtest3.cpp -o/home/yui/Desktop/Progtest3.out -Wall
(g++ /檔案路徑/檔案名稱.cpp -o/檔案路徑/檔案名稱.out -Wall),並按下 Enter

```
yui@yui-VirtualBox:~/Desktop$ g++ /home/yui/Desktop/Progtest3.cpp -o/home/yui/Desktop/Progtest3.out -Wall
```

(3)輸入 ./Progtest3.out 0 23 (./檔案名稱.out 輸入基本功能(模式)和 Randomseed)
(g++ /檔案路徑/檔案名稱.cpp -o/檔案路徑/檔案名稱.out -Wall),並按下 Enter

```
yui@yui-VirtualBox:~/Desktop$ ./Progtest3.out 0 23
```

(4)觀看結果