```cpp
#include <iostream>
#include <pthread.h>
#include <cstdlib>
#include <ctime>
#include <unistd.h>
#include <string>
#include <cmath>
#include <algorithm>
using namespace std;
```

此為這次作業需要用到的標頭檔

```cpp
bool dis[3] = {false};
bool p1_item[3] = {false,false,true};
bool p2_item[3] = {false};
bool p3_item[3] = {false};
bool check[3] = {true};
int sum =0;
int dis_item[3] = {0};
pthread_mutex_t lock;
int seed;
```

各項全域變數說明:
dis[3]:平台上擁有的零件
p1_item[3]:producer1 擁有的零件
p2_item[3]:producer2 擁有的零件
p3_item[3]:producer3 擁有的零件
以上陣列代表的零件分別為
1:propeller　2:battery　3:aircraft

check[3]:查看 producer1~3 是否都有進入 critical section，是則為 true
sum:紀錄製造出來的空拍機總數
dis_item[3]:紀錄 dispatcher 分配給平台的各零件數量
pthread_mutex_t lock 創建一個 mutex 鎖
seed:亂數種子

```cpp
int trans(string value)
{
    unsigned int deg = value.size() - 1,sum=0;
    for(unsigned int i=0;i<value.size();i++)
    {
        int tmp=(int)value[i]-'0';
        sum = sum+tmp * pow(10,deg);
        deg--;
    }
    return sum;
}
```

這段目的為將一串以 string 形式表達的數字(EX: 23)轉換為 int 的資料型態，由於數字有超過 2 位數以上的可能，因此需要用此方法進行轉換(用於下方第 276 行)

```cpp
int main(int argc ,char* argv[])
{
    string ran;
    char m_t = (*argv[1]);
    int mode = (int)m_t-'0';
    if(mode == 1)
       cout<<"Sorry , i did not make advanced function"<<endl;
    else
    {
       int p1=1,p2=2,p3=3;
       for(int i=2; i<argc;i++)
           ran.push_back(*argv[i]);
       seed = trans(ran);
       pthread_t thr[4];
       pthread_mutex_init(&lock , NULL);
       pthread_create (&thr[0],NULL,dispatcher,(void*)"dispatcher"
       pthread_create (&thr[1],NULL,producer,&p1);
       pthread_create (&thr[2],NULL,producer,&p2);
       pthread_create (&thr[3],NULL,producer,&p3);
       pthread_join(thr[0] , NULL);
       pthread_join(thr[1] , NULL);
       pthread_join(thr[2] , NULL);
       pthread_join(thr[3] , NULL);
```

這段旨在處裡輸入的功能及亂數 seed,由於本次作業我只有做基本功能，因此當輸入 1(進階功能)時，輸出並無製作進階功能

令 p1=1,p2=2,p3=3，方便讀入 producer 函式後判斷為哪一個 producer，取得亂數種子 seed 後並建立 4 個 thread，thr[0]為 dispatcher, thr[1]為 producer1, thr[2]為 producer2, thr[3]為 producer3 建立時將 p1,p2,p3 分別讀入，以利函試判斷為哪一個 producer thread，隨後 pthread_join 等待對應的 thread 執行結束

```
43 void* dispatcher(void* arg)
44 {
45     srand (time(NULL)^seed);
46     while(sum != 50)
47     {
48         pthread_mutex_lock(&lock);
49         if(sum == 50 || check[0]==false || check[1]==false || check[2]==false)
50         {
51             pthread_mutex_unlock(&lock);
52             continue;
53         }
54         int item = 0;
55         while(true)
56         {
57             if(dis[0]==true && dis[1]==true && dis[2]==true)
58                 break;
59             item = rand()%3;
60             if(dis[item] == false)
61             {
62                 dis[item] = true;
63                 break;
64             }
65             else
66                 continue;
67         }

70         switch (item)
71         {
72             case 0:
73                 dis_item[0]++;
74                 cout<<"Dispatcher: propeller"<<endl;
75                 break;
76             case 1:
77                 dis_item[1]++;
78                 cout<<"Dispatcher: battery"<<endl;
79                 break;
80             case 2:
81                 dis_item[2]++;
82                 cout<<"Dispatcher: aircraft"<<endl;
83                 break;
84             default:
85                 break;
86         }
87         check[0]=false;
88         check[1]=false;
89         check[2]=false;
90         pthread_mutex_unlock(&lock);
91     }
92         return NULL;
93 }
```

利用 seed 設定一個隨機亂數種子

這邊寫 while(sum!=50)意義為直到造出 50 台空拍機為止，持續工作。當一進入迴圈後即將 mutex 鎖上，並且判斷如果其他人已順利造出第 50 台或 dispatcher 搶到 critical section 但有 producer 並未進入平台(CS)內查看是否有他們需要的零件時，則將 mutex 釋放，並強制迴圈繼續

定義變數 item，功能為用於存取隨機亂數及記錄抽取的零件，進入 while 迴圈，如果此平台所有零件都有則跳出,如有少零件則隨機抽取一個零件放入平台，寫 while 迴圈的原因在於如果抽到平台上有的零件則需要再抽取一次，直到抽到平台上沒有的零件為止。

由 item 可得知 dispatcher 在平台上放入哪一個零件，因此用 switch case 判斷，並令 dis_item[]++(計算 dispatcher 總共在平台上放個別放多少零件),隨後初始化 check 並將 mutex 釋放

```
21 struct pi_item
22 {
23     string pi;
24     int num = 0;
25 };
26 pi_item pdc[3];
```

```
95 void* producer(void* arg)
96 {
97     int p_num = *(int *)arg;
98     int p1_sum = 0, p2_sum = 0,p3_sum = 0;
99     pdc[0].pi = "producer 1 (aircraft): ";
100    pdc[1].pi = "producer 2: ";
101    pdc[2].pi = "producer 3: ";
```

Producer 內變數介紹:

p_num:將讀入參數轉成 int 型態，作用為判斷讀入的 function 為哪一個 producer

p1_sum~p3_sum:紀錄每一個 producer 製作了幾台空拍機

pdc[0~2].p1:這是一個 struct 型態(配合上方黃框程式碼)，用於記錄每一個 producer 的名稱和製作幾台空拍機(方便之後的排序)

```
while(sum != 50)
{
   if(p_num == 1)
   {
```

與 dispatcher 一樣，直到空拍機總數達 50 前持續執行，if(p_num == 1)為判斷是哪一個 producer，往後的 producer 都有此判斷，以區分每一個 producer

```
      bool take=false;
      int item_num;
      pthread_mutex_lock(&lock);
      check[0] = true;
      if(sum == 50)
      {
         pthread_mutex_unlock(&lock);
         break;
      }
```

take:判斷是否有從平台拿取零件
item_lock:如有拿取零件，紀錄是哪一個零件
將 mutex 鎖上後 check[0]=true，紀錄 producer1 已進入 CS(平台)中查看過

```
      for(int i=0;i<2;i++)
      {
         if(dis[i] == true && p1_item[i] == false)
         {
             dis[i]=false;
             p1_item[i] = true;
             item_num = i;
             take = true;
             break;
         }
         else
             continue;
      }
```

利用迴圈逐一判斷平台內是否有 producer1 需要的零件
判斷方式為 如果 dis(平台)有，但 p1_item(producer1)沒有，代表可拿取。當可拿取時將 dis 和 p1_item 狀態交換，並用 item_num 記錄拿取的零件，再將 take=true 表示已拿取

```
      if(take == true)
      {
         switch (item_num)
         {
            case 0:
               cout<<"Producer 1 (aircraft): get propeller"<<endl;
               break;
            case 1:
               cout<<"Producer 1 (aircraft): get battery"<<endl;
               break;
         }
      }
```

take = true 代表 producer 已從 CS 中拿取，所需零件，因此判斷是哪一個零件後輸出對應的零件

```
      if(p1_item[0]==true && p1_item[1]==true && p1_item[2]==true)
      {
          p1_sum++;
          pdc[0].num++;
          sum++;
          cout<<"Producer 1 (aircraft): OK "<<p1_sum<<" drone(s)"<<endl;
          p1_item[0] = false;
          p1_item[1] = false;
          p1_item[2] = true;
      }
      pthread_mutex_unlock(&lock);
   }
```

在拿完零件之後判斷 p1_item[0~2]是否都有零件，都有代表可生產出一台空拍機了，此時
p1_sum++ (producer1 製作空拍機總數)
pdc[0].num++(producer1 製作空拍機總數)
sum++(所有空拍機總數)
輸出後初始化 p1_item(producer1 的 aircraft 一直都有，因此初始化 p1_item[2]為 true)
隨後將 mutex 釋放

```
153     else if(p_num == 2)
154     {
155        bool take=false;
156        int item_num;
157        pthread_mutex_lock(&lock);
158        check[1] = true;
159        if(sum == 50)
160        {
161           pthread_mutex_unlock(&lock);
162           break;
163        }
164
165        for(int i=0;i<3;i++)
166        {
167           if(dis[i] == true && p2_item[i] == false)
168           {
169               dis[i]=false;
170               p2_item[i] = true;
171               item_num = i;
172               take = true;
173               break;
174           }
175           else
176               continue;
177        }
178
179        if(take == true)
180        {
181           switch (item_num)
182           {
183              case 0:
184                 cout<<"Producer 2: get propeller"<<endl;
185                 break;
186              case 1:
187                 cout<<"Producer 2: get battery"<<endl;
188                 break;
189              case 2:
190                 cout<<"Producer 2: get aircraft"<<endl;
191                 break;
192           }
193        }
194        if(p2_item[0]==true && p2_item[1]==true && p2_item[2]==true)
195        {
196            p2_sum++;
197            sum++;
198            pdc[1].num++;
199            cout<<"Producer 2: OK "<<p2_sum<<" drone(s)"<<endl;
200            p2_item[0] = false;
201            p2_item[1] = false;
202            p2_item[2] = false;
203        }
204        pthread_mutex_unlock(&lock);
205     }
```

Producer 2

```
207     else
208     {
209        bool take=false;
210        int item_num;
211        pthread_mutex_lock(&lock);
212        check[2] = true;;
213        if(sum == 50)
214        {
215           pthread_mutex_unlock(&lock);
216           break;
217        }
218
219        for(int i=0;i<3;i++)
220        {
221           if(dis[i] == true && p3_item[i] == false)
222           {
223               dis[i]=false;
224               p3_item[i] = true;
225               item_num = i;
226               take = true;
227               break;
228           }
229           else
230               continue;
231        }
232
233        if(take == true)
234        {
235           switch (item_num)
236           {
237              case 0:
238                 cout<<"Producer 3: get propeller"<<endl;
239                 break;
240              case 1:
241                 cout<<"Producer 3: get battery"<<endl;
242                 break;
243              case 2:
244                 cout<<"Producer 3: get aircraft"<<endl;
245                 break;
246           }
247        }
248        if(p3_item[0]==true && p3_item[1]==true && p3_item[2]==true)
249        {
250            p3_sum++;
251            sum++;
252            pdc[2].num++;
253            cout<<"Producer 3: OK "<<p3_sum<<" drone(s)"<<endl;
254            p3_item[0] = false;
255            p3_item[1] = false;
256            p3_item[2] = false;
257        }
258        pthread_mutex_unlock(&lock);
259     }
260  }
261  return NULL;
262 }
```

Producer 3

由於 producer 2 和 producer 3 的程式碼與 producer 1 的幾乎相同，因此就不多加進行贅述，producer 2 和 producer 3 的程式碼如上

```
286       pthread_join(thr[3] , NULL);
287
288       cout<<endl<<"Dispatcher has prepared "<<dis_item[0]<<" [Propeller] module accessories"<<endl;
289       cout<<"Dispatcher has prepared "<<dis_item[1]<<" [Battery] module accessories"<<endl;
290       cout<<"Dispatcher has prepared "<<dis_item[2]<<" [Aircraft] module accessories"<<endl<<endl;
291
292       sort(pdc , pdc+3, compare);
293       for(int i=0;i<3;i++)
294          cout<<pdc[i].pi<<pdc[i].num<<" aerial cameras"<<endl;
295
296    }
297 }
```

當 join 等待 thread 執行完後 main thread 輸出 dispatcher 分配給平台的各項零件數

```
27 bool compare(pi_item p1,pi_item p2)
28 {
29    return p1.num > p2.num;
30 }
```

輸出完後將 pdc 進行排序(compare 為排序依據，由於 struct 的排序需要自己寫排序依據，參考上方綠色框中程式碼)，排序完後依序將 pdc[0~2](每個 producer 製作的空拍機數量)輸出，至此所有程式結束

編譯方法:

```
Box:~/Desktop$ g++ /home/jacky/Desktop/HW_3.cpp -o/home/jacky/Desktop/HW_3.out -Wall
```

打開 ubuntu 的 terminal 輸入 g++ /檔案位置/檔名.cpp -o/檔案位置/檔名.out -Wall 建立.out 檔

```
~/Desktop$ ./HW_3.out 0 10
```

輸入 ./檔名.out a b 執行.out 檔及輸入參數

觀看(部分)結果:

```
Producer 2: get battery
Dispatcher: propeller
Producer 2: get propeller
Producer 2: OK 13 drone(s)
Producer 2: get aircraft
Dispatcher: battery
Producer 1 (aircraft): get battery
Dispatcher: aircraft
Dispatcher: battery
Producer 2: get battery
Dispatcher: battery
Producer 3: get battery
Producer 3: OK 16 drone(s)
Producer 3: get aircraft
Dispatcher: propeller
Producer 2: get propeller
Producer 2: OK 14 drone(s)
Dispatcher: aircraft
Producer 2: get aircraft
Dispatcher: aircraft
Dispatcher: propeller
Producer 1 (aircraft): get propeller
Producer 1 (aircraft): OK 18 drone(s)
Dispatcher: propeller
Producer 3: get propeller
Dispatcher: battery
Producer 1 (aircraft): get battery
Dispatcher: battery
Producer 2: get battery
Dispatcher: propeller
Producer 2: get propeller
Producer 2: OK 15 drone(s)
Producer 2: get aircraft
Dispatcher: propeller
Producer 1 (aircraft): get propeller
Producer 1 (aircraft): OK 19 drone(s)

Dispatcher has prepared 51 [Propeller] module accessories
Dispatcher has prepared 50 [Battery] module accessories
Dispatcher has prepared 33 [Aircraft] module accessories

producer 1 (aircraft): 19 aerial cameras
producer 3: 16 aerial cameras
producer 2: 15 aerial cameras
```