Pork Zongzi Maker

S1083314柯宣羽

1.完成的功能

- 每10ms-100ms 檢查工廠有無在工作，沒有的話就進入檢討模式或維護模式
- 每 50-100ms 會送上一塊豬肉原塊到備料格，直到M塊豬肉都產生完
- N個備料格如果都滿了，送到冰箱冰約300-500ms，冰完還沒位子的話就繼續冰，有位子就送到備料格
- 如果沒滿且切割工廠沒在工作的話，以 FCFS 送到切割工廠待 100-300ms，切好了送到備料格，備料格都被切好的豬肉佔滿的話就等到有位子
- 備料格如果有切好的豬肉且粽子工廠沒在工作的話，以 FCFS 送到粽子工廠待 500-1000ms，包好了就送走

2.編譯方式

3.執行方式

- `g++ s1083314_OShw3.cpp -lpthread -o s1083314_OShw3.out`
- `./s1083314_OShw3.out <豬肉數量> <備料格數量>`

(ubuntu有安裝pthread和C++)

4.程式運作元件

1. Thread * 4
   - porkGenerate_thd
   - cutter_thd
   - packer_thd
   - freezer_thd
2. mutex * 4
   - print_mutex
   - slot_mutex
   - fridge_mutex
   - working_mutex
3. 儲存的資料結構 * 3 及變數 * 2
   - queue<Pork> origin_slot
   - queue<Pork> cutted_slot
   - vector<Pork> fridge
   - bool cutter_working
   - bool packer_working

程式運作邏輯
(1)生產豬肉原塊

- porkGenerate_thd 不斷產生新豬肉，生產後視情況加到 origin_slot 或是 fridge裡，並分別利用 slot_mutex 和 fridge_mutex 確保沒有其他人同時讀或寫 origin_slot、cutted_slot 或 fridge

程式運作邏輯
(2)切豬肉

- cutter_thd 不斷找豬肉來切，利用 slot_mutex 確保沒有其他人同時讀或寫 origin_slot
- 如果沒有豬肉的話，就利用 working_mutex 存取 packer_working，判斷要維護還是檢討
- 有豬肉的話，切好後加到 cutted_slot裡，並利用 slot_mutex 確保沒有其他人同時讀或寫 cutted_slot 或 origin_slot

程式運作邏輯
(3)包粽子

- packer_thd 不斷找豬肉來包粽子，利用 slot_mutex 確保沒有其他人同時存取 cutted_slot
- 如果沒有豬肉的話，就利用 working_mutex 存取 cutter_working，判斷要維護還是檢討
- 有豬肉的話就再包好後送出去

程式運作邏輯
(4)冷凍庫

- freezer_thd 不斷檢查有沒有豬肉已經冷凍好，且有備料格可以存放了，並分別利用 slot_mutex 和 fridge_mutex 確保沒有其他人同時讀或寫 origin_slot、cutted_slot 或 fridge
- 如果沒問題的話，就將豬肉存到 origin_slot中

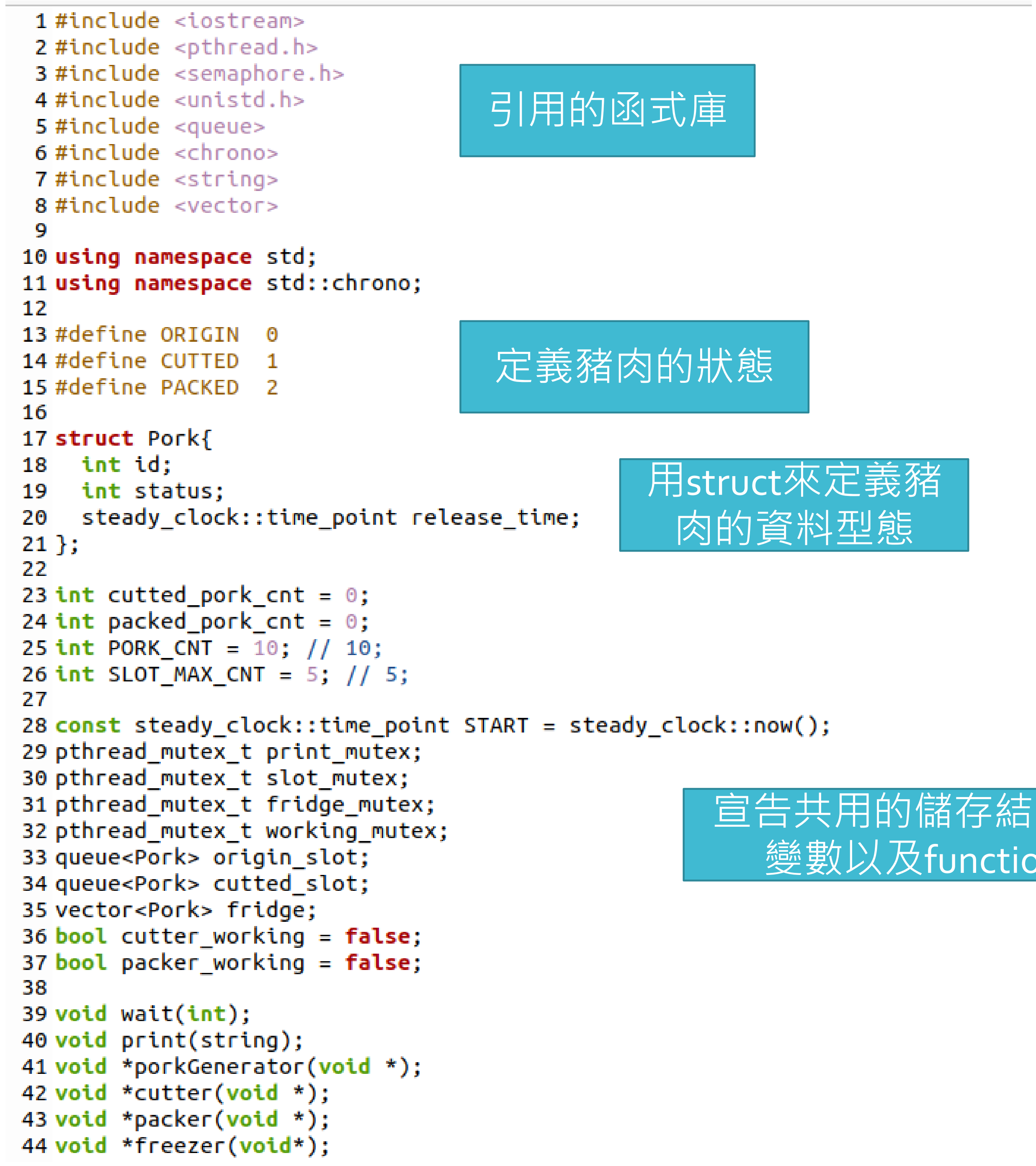
程式運作邏輯
(5)印出信息

- 利用 print_mutex 確保同時只有一個 thread 在輸出

程式運作邏輯
(6) 關於slot_mutex 和 working_mutex 以及deadlock防範

- 這兩個 mutex 都對應到兩個資源：分別是origin_slot + cutted_slot和cutter_working + packer_working
- 之所以不用四個 mutex，讓一個資源對到一個 mutex，是為了防止 deadlock。一個 mutex 中的兩個資源有可能會一起被使用，或是彼此使用意義類似，在目前 consumer 數量稀少的情況下，這種使用方法會是比較好的選擇。

程式碼運作說明



```cpp
#include <iostream>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <queue>
#include <chrono>
#include <string>
#include <vector>

using namespace std;
using namespace std::chrono;

#define ORIGIN  0
#define CUTTED  1
#define PACKED  2

struct Pork{
  int id;
  int status;
  steady_clock::time_point release_time;
};

int cutted_pork_cnt = 0;
int packed_pork_cnt = 0;
int PORK_CNT = 10; // 10;
int SLOT_MAX_CNT = 5; // 5;

const steady_clock::time_point START = steady_clock::now();
pthread_mutex_t print_mutex;
pthread_mutex_t slot_mutex;
pthread_mutex_t fridge_mutex;
pthread_mutex_t working_mutex;
queue<Pork> origin_slot;
queue<Pork> cutted_slot;
vector<Pork> fridge;
bool cutter_working = false;
bool packer_working = false;

void wait(int);
void print(string);
void *porkGenerator(void *);
void *cutter(void *);
void *packer(void *);
void *freezer(void*);
```

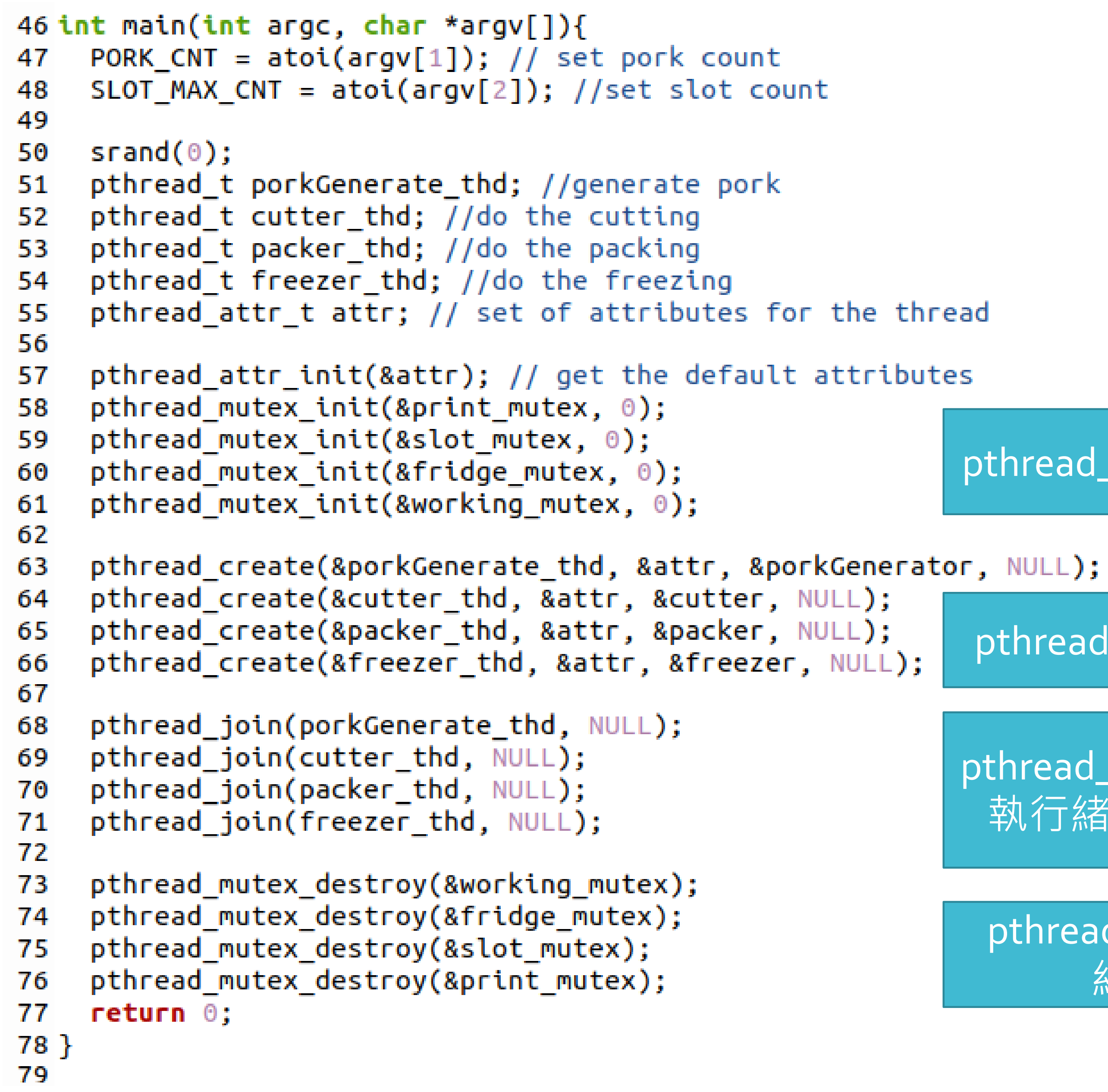
引用的函式庫

定義豬肉的狀態

用struct來定義豬肉的資料型態

宣告共用的儲存結構和變數以及function

程式碼運作說明 main



```c
int main(int argc, char *argv[]){
  PORK_CNT = atoi(argv[1]); // set pork count
  SLOT_MAX_CNT = atoi(argv[2]); //set slot count

  srand(0);
  pthread_t porkGenerate_thd; //generate pork
  pthread_t cutter_thd; //do the cutting
  pthread_t packer_thd; //do the packing
  pthread_t freezer_thd; //do the freezing
  pthread_attr_t attr; // set of attributes for the thread

  pthread_attr_init(&attr); // get the default attributes
  pthread_mutex_init(&print_mutex, 0);
  pthread_mutex_init(&slot_mutex, 0);
  pthread_mutex_init(&fridge_mutex, 0);
  pthread_mutex_init(&working_mutex, 0);

  pthread_create(&porkGenerate_thd, &attr, &porkGenerator, NULL);
  pthread_create(&cutter_thd, &attr, &cutter, NULL);
  pthread_create(&packer_thd, &attr, &packer, NULL);
  pthread_create(&freezer_thd, &attr, &freezer, NULL);

  pthread_join(porkGenerate_thd, NULL);
  pthread_join(cutter_thd, NULL);
  pthread_join(packer_thd, NULL);
  pthread_join(freezer_thd, NULL);

  pthread_mutex_destroy(&working_mutex);
  pthread_mutex_destroy(&fridge_mutex);
  pthread_mutex_destroy(&slot_mutex);
  pthread_mutex_destroy(&print_mutex);
  return 0;
}
```

pthread_mutex_init初始化

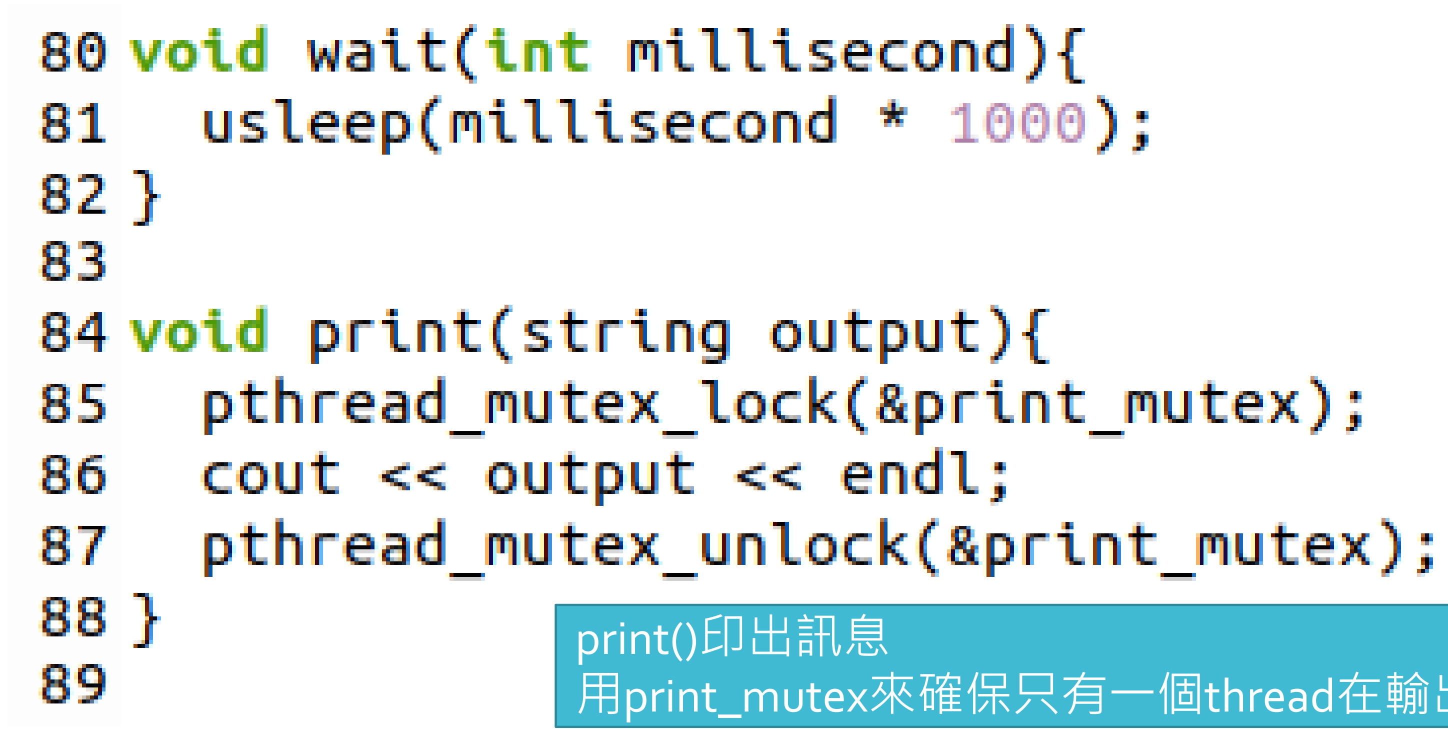
pthread_create建立執行緒

pthread_join使建立的執行緒有機會執行

pthread_mutex_destroy 結束時銷毀

程式碼運作說明 wait和print

執行時會使用的wait()

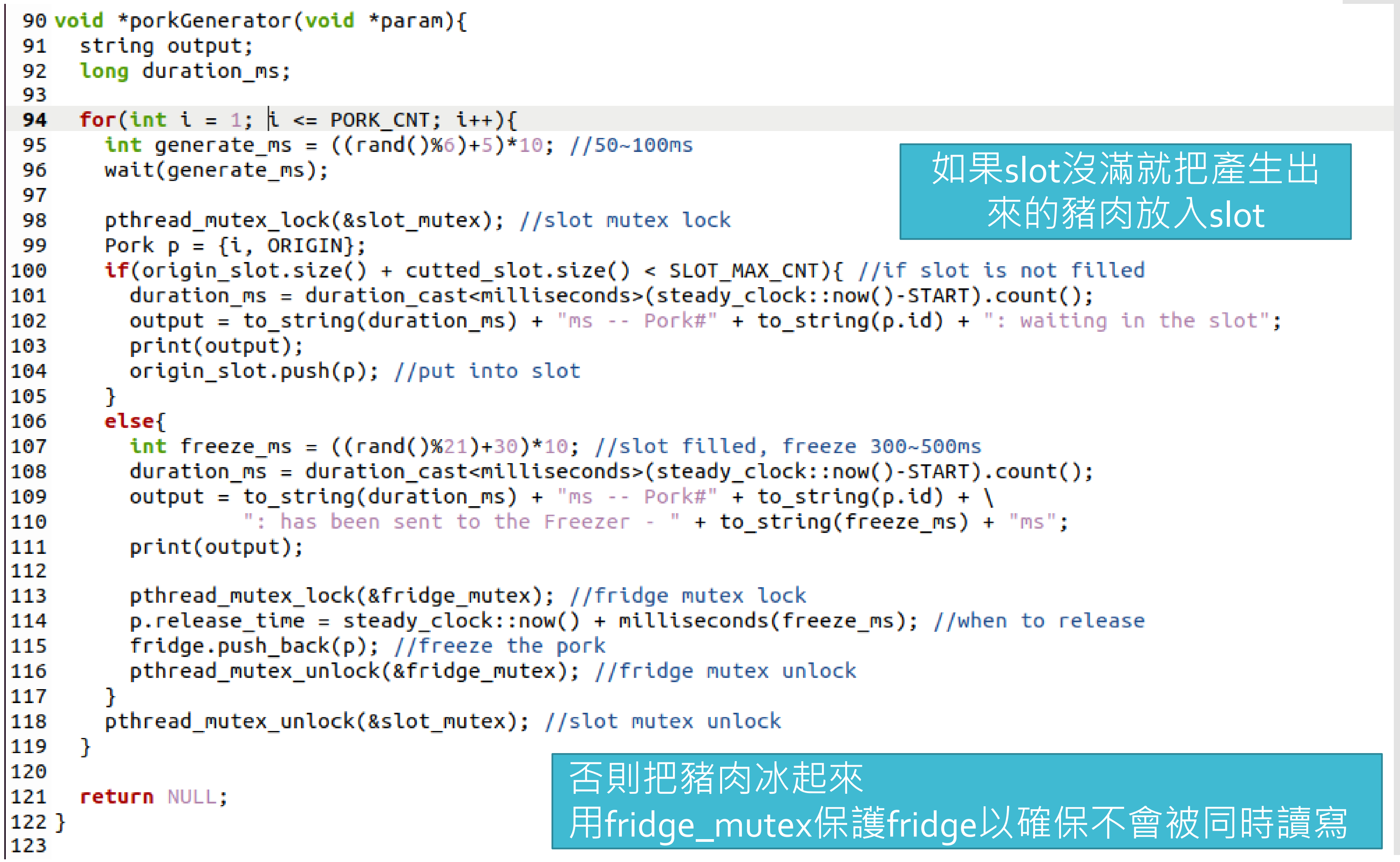


```
void wait(int millisecond){
  usleep(millisecond * 1000);
}

void print(string output){
  pthread_mutex_lock(&print_mutex);
  cout << output << endl;
  pthread_mutex_unlock(&print_mutex);
}

```

print()印出訊息
用print_mutex來確保只有一個thread在輸出

程式碼運作說明 porkGenerator

porkGenerate_thd負責產生豬肉
用slot_mutex保護origin_slot以確保不會被同時讀寫



```
void *porkGenerator(void *param){
  string output;
  long duration_ms;

  for(int i = 1; i <= PORK_CNT; i++){
    int generate_ms = ((rand()%6)+5)*10; //50~100ms
    wait(generate_ms);

    pthread_mutex_lock(&slot_mutex); //slot mutex lock
    Pork p = {i, ORIGIN};
    if(origin_slot.size() + cutted_slot.size() < SLOT_MAX_CNT){ //if slot is not filled
      duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
      output = to_string(duration_ms) + "ms -- Pork#" + to_string(p.id) + ": waiting in the slot";
      print(output);
      origin_slot.push(p); //put into slot
    }
    else{
      int freeze_ms = ((rand()%21)+30)*10; //slot filled, freeze 300~500ms
      duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
      output = to_string(duration_ms) + "ms -- Pork#" + to_string(p.id) + \
               ": has been sent to the Freezer - " + to_string(freeze_ms) + "ms";
      print(output);

      pthread_mutex_lock(&fridge_mutex); //fridge mutex lock
      p.release_time = steady_clock::now() + milliseconds(freeze_ms); //when to release
      fridge.push_back(p); //freeze the pork
      pthread_mutex_unlock(&fridge_mutex); //fridge mutex unlock
    }
    pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
  }

  return NULL;
}
```

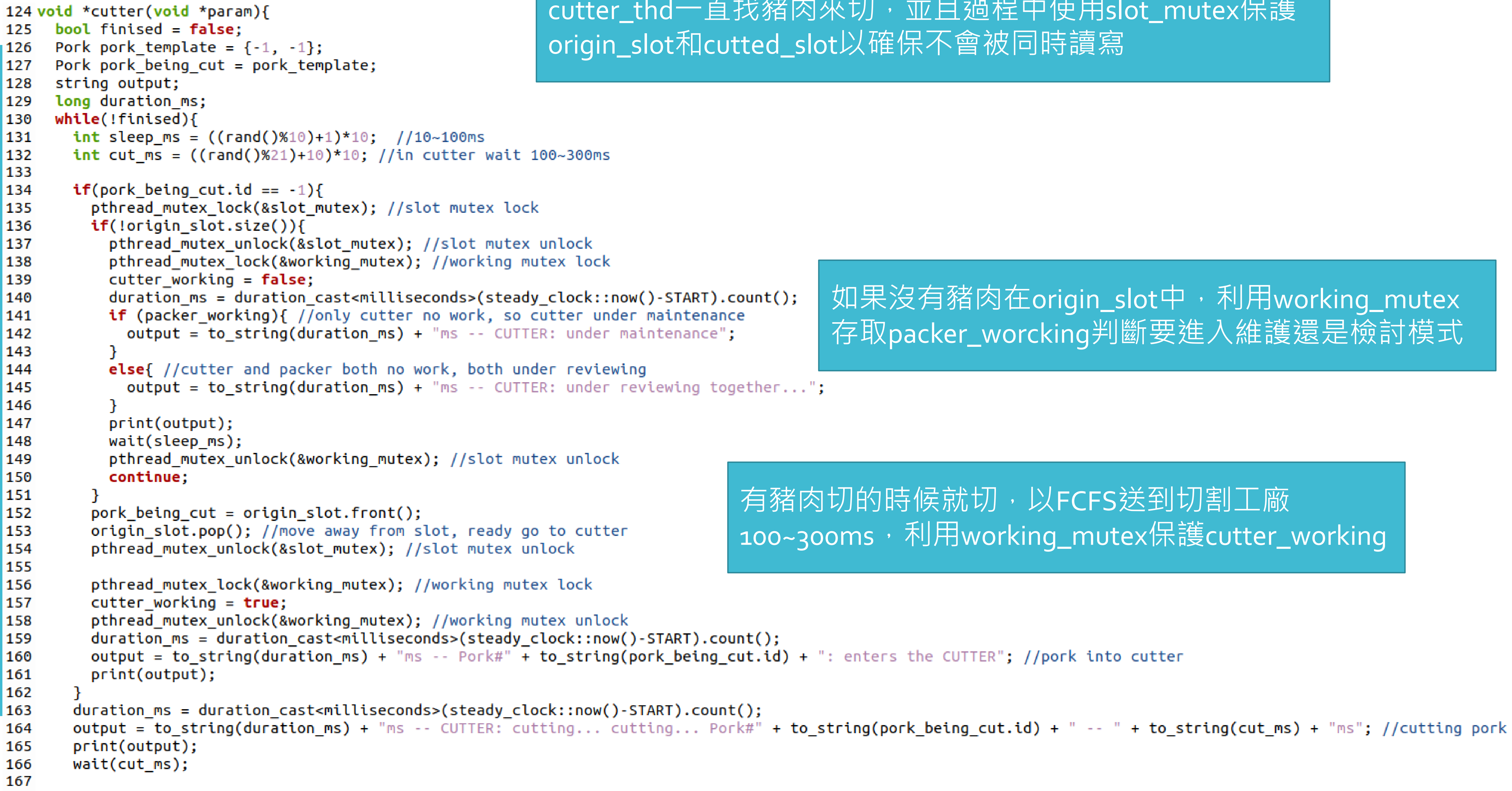
如果slot沒滿就把產生出來的豬肉放入slot

否則把豬肉冰起來
用fridge_mutex保護fridge以確保不會被同時讀寫

程式碼運作說明 cutter

共兩張圖，因為太大張了所以cutter分開放在下兩頁

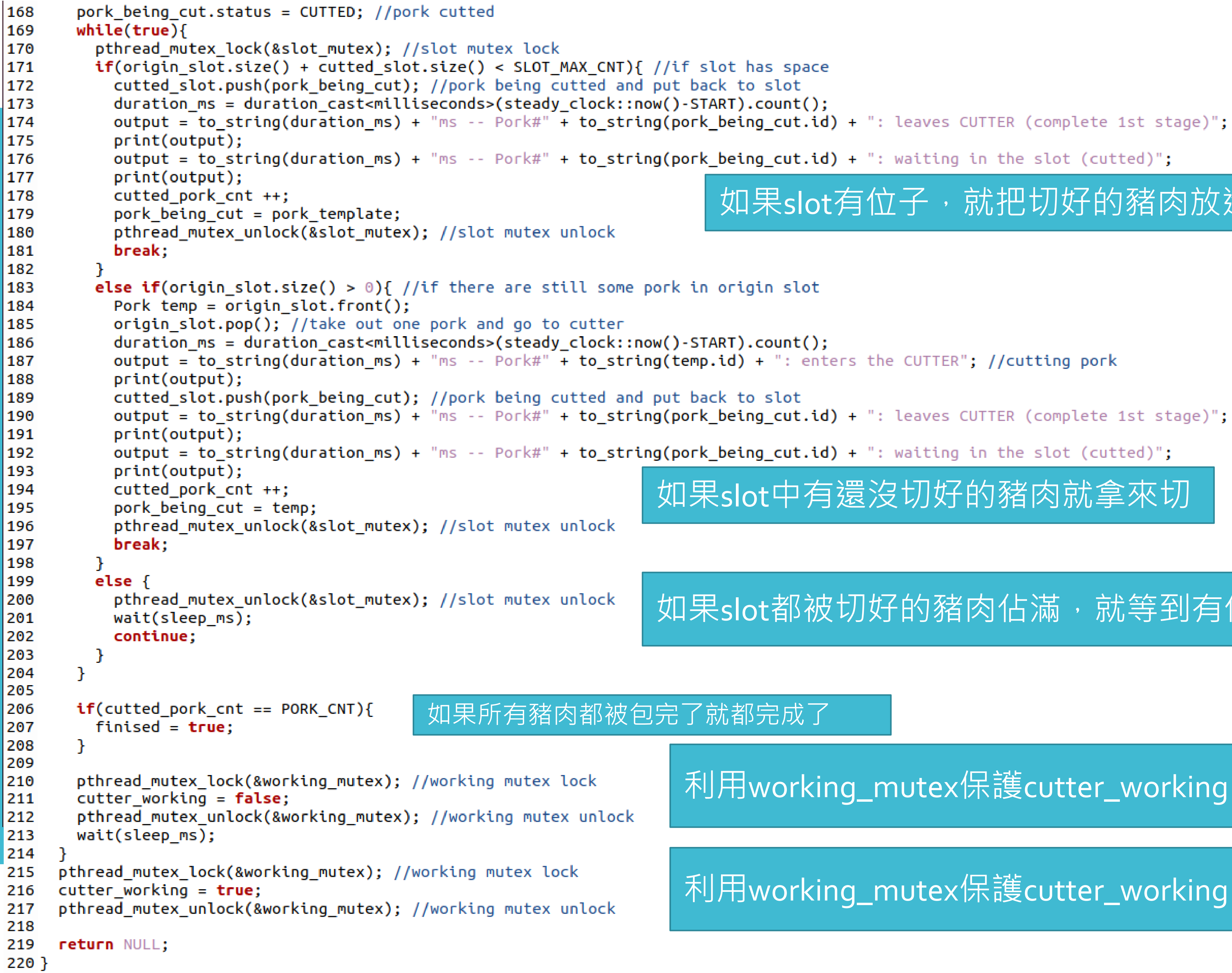
cutter_thd一直找豬肉來切，並且過程中使用slot_mutex保護origin_slot和cutted_slot以確保不會被同時讀寫



```
void *cutter(void *param){
 bool finised = false;
 Pork pork_template = {-1, -1};
 Pork pork_being_cut = pork_template;
 string output;
 long duration_ms;
 while(!finised){
  int sleep_ms = ((rand()%10)+1)*10;  //10~100ms
  int cut_ms = ((rand()%21)+10)*10; //in cutter wait 100~300ms

  if(pork_being_cut.id == -1){
   pthread_mutex_lock(&slot_mutex); //slot mutex lock
   if(!origin_slot.size()){
    pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
    pthread_mutex_lock(&working_mutex); //working mutex lock
    cutter_working = false;
    duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
    if (packer_working){ //only cutter no work, so cutter under maintenance
     output = to_string(duration_ms) + "ms -- CUTTER: under maintenance";
    }
    else{ //cutter and packer both no work, both under reviewing
     output = to_string(duration_ms) + "ms -- CUTTER: under reviewing together...";
    }
    print(output);
    wait(sleep_ms);
    pthread_mutex_unlock(&working_mutex); //slot mutex unlock
    continue;
   }
   pork_being_cut = origin_slot.front();
   origin_slot.pop(); //move away from slot, ready go to cutter
   pthread_mutex_unlock(&slot_mutex); //slot mutex unlock

   pthread_mutex_lock(&working_mutex); //working mutex lock
   cutter_working = true;
   pthread_mutex_unlock(&working_mutex); //working mutex unlock
   duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
   output = to_string(duration_ms) + "ms -- Pork#" + to_string(pork_being_cut.id) + ": enters the CUTTER"; //pork into cutter
   print(output);
  }
  duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
  output = to_string(duration_ms) + "ms -- CUTTER: cutting... cutting... Pork#" + to_string(pork_being_cut.id) + " -- " + to_string(cut_ms) + "ms"; //cutting pork
  print(output);
  wait(cut_ms);

```

如果沒有豬肉在origin_slot中，利用working_mutex存取packer_worcking判斷要進入維護還是檢討模式

有豬肉切的時候就切，以FCFS送到切割工廠100~300ms，利用working_mutex保護cutter_working



```
    pork_being_cut.status = CUTTED; //pork cutted
    while(true){
      pthread_mutex_lock(&slot_mutex); //slot mutex lock
      if(origin_slot.size() + cutted_slot.size() < SLOT_MAX_CNT){ //if slot has space
        cutted_slot.push(pork_being_cut); //pork being cutted and put back to slot
        duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
        output = to_string(duration_ms) + "ms -- Pork#" + to_string(pork_being_cut.id) + ": leaves CUTTER (complete 1st stage)";
        print(output);
        output = to_string(duration_ms) + "ms -- Pork#" + to_string(pork_being_cut.id) + ": waiting in the slot (cutted)";
        print(output);
        cutted_pork_cnt ++;
        pork_being_cut = pork_template;
        pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
        break;
      }
      else if(origin_slot.size() > 0){ //if there are still some pork in origin slot
        Pork temp = origin_slot.front();
        origin_slot.pop(); //take out one pork and go to cutter
        duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
        output = to_string(duration_ms) + "ms -- Pork#" + to_string(temp.id) + ": enters the CUTTER"; //cutting pork
        print(output);
        cutted_slot.push(pork_being_cut); //pork being cutted and put back to slot
        output = to_string(duration_ms) + "ms -- Pork#" + to_string(pork_being_cut.id) + ": leaves CUTTER (complete 1st stage)";
        print(output);
        output = to_string(duration_ms) + "ms -- Pork#" + to_string(pork_being_cut.id) + ": waiting in the slot (cutted)";
        print(output);
        cutted_pork_cnt ++;
        pork_being_cut = temp;
        pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
        break;
      }
      else {
        pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
        wait(sleep_ms);
        continue;
      }
    }

    if(cutted_pork_cnt == PORK_CNT){
      finised = true;
    }

    pthread_mutex_lock(&working_mutex); //working mutex lock
    cutter_working = false;
    pthread_mutex_unlock(&working_mutex); //working mutex unlock
    wait(sleep_ms);
  }
  pthread_mutex_lock(&working_mutex); //working mutex lock
  cutter_working = true;
  pthread_mutex_unlock(&working_mutex); //working mutex unlock

  return NULL;
}
```

如果slot有位子，就把切好的豬肉放進slot

如果slot中有還沒切好的豬肉就拿來切

如果slot都被切好的豬肉佔滿，就等到有位子再

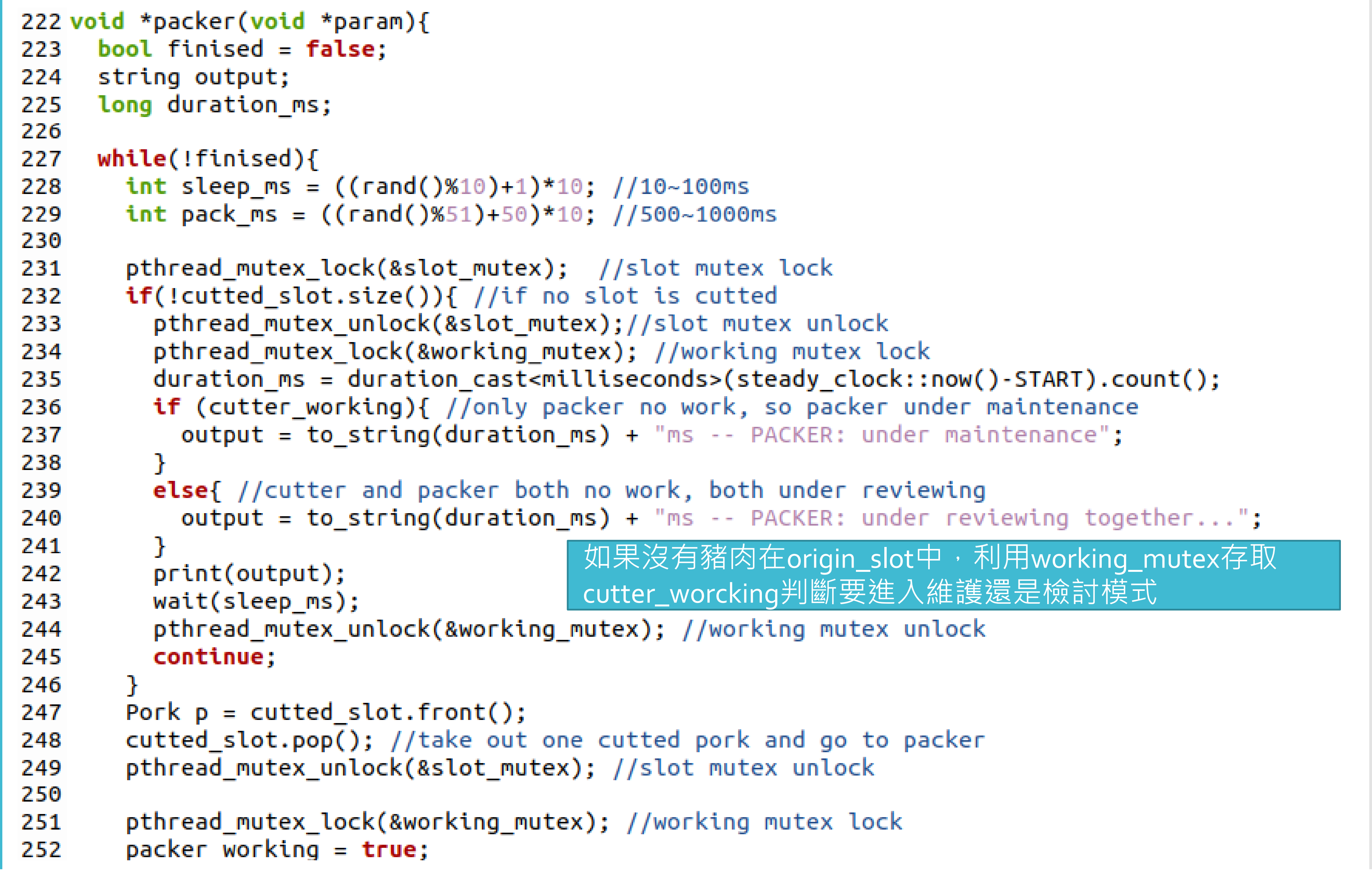
如果所有豬肉都被包完了就都完成了

利用working_mutex保護cutter_working

利用working_mutex保護cutter_working

packer_thd負責包粽子，用slot_mutex保護cutted_slot以確保不會被同時讀寫

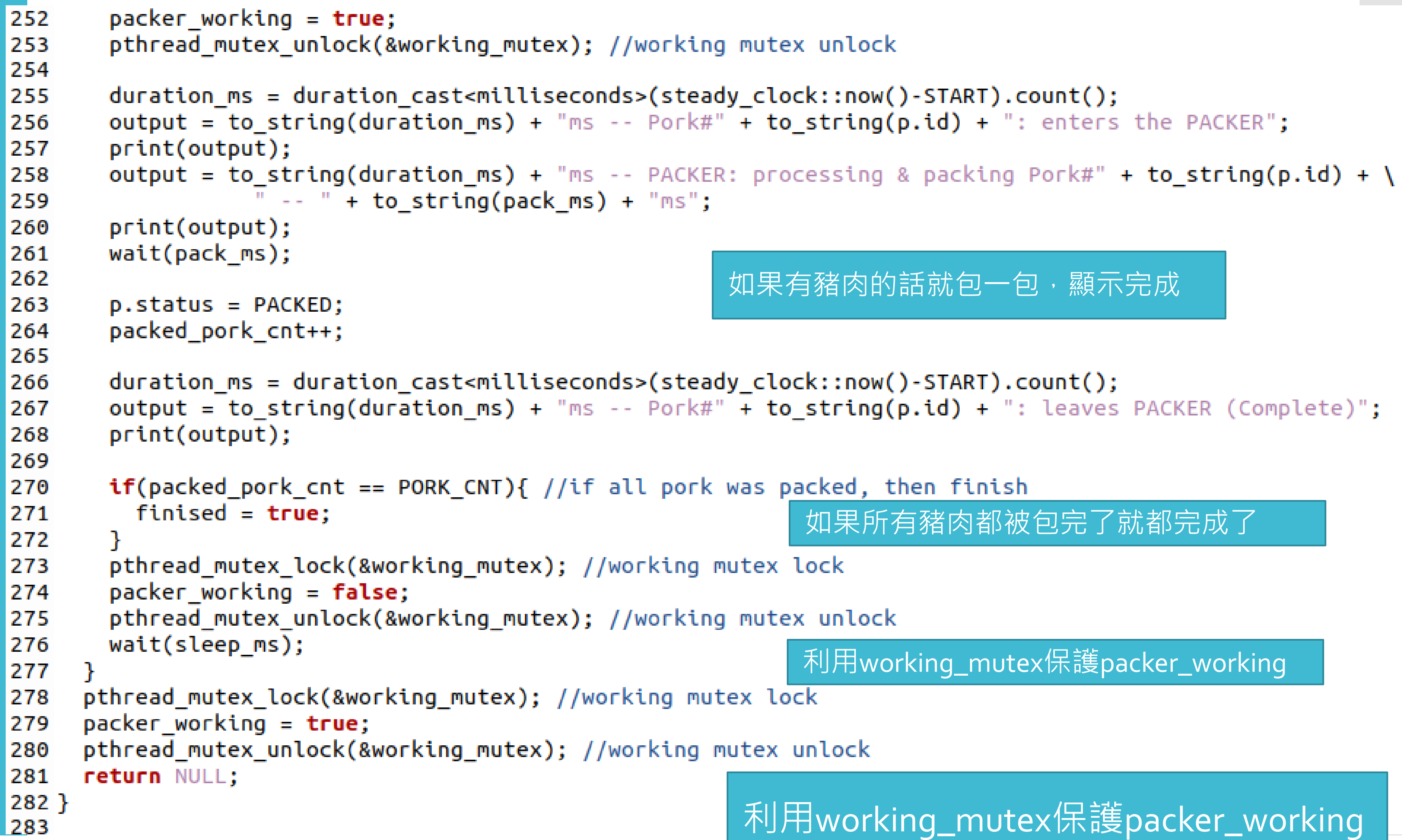
程式碼運作說明 packer



```
void *packer(void *param){
  bool finised = false;
  string output;
  long duration_ms;

  while(!finised){
    int sleep_ms = ((rand()%10)+1)*10; //10~100ms
    int pack_ms = ((rand()%51)+50)*10; //500~1000ms

    pthread_mutex_lock(&slot_mutex);  //slot mutex lock
    if(!cutted_slot.size()){ //if no slot is cutted
      pthread_mutex_unlock(&slot_mutex);//slot mutex unlock
      pthread_mutex_lock(&working_mutex); //working mutex lock
      duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
      if (cutter_working){ //only packer no work, so packer under maintenance
        output = to_string(duration_ms) + "ms -- PACKER: under maintenance";
      }
      else{ //cutter and packer both no work, both under reviewing
        output = to_string(duration_ms) + "ms -- PACKER: under reviewing together...";
      }
      print(output);
      wait(sleep_ms);
      pthread_mutex_unlock(&working_mutex); //working mutex unlock
      continue;
    }
    Pork p = cutted_slot.front();
    cutted_slot.pop(); //take out one cutted pork and go to packer
    pthread_mutex_unlock(&slot_mutex); //slot mutex unlock

    pthread_mutex_lock(&working_mutex); //working mutex lock
    packer_working = true;
```

如果沒有豬肉在origin_slot中，利用working_mutex存取cutter_worcking判斷要進入維護還是檢討模式

程式碼運作說明
cutter



```
    packer_working = true;
    pthread_mutex_unlock(&working_mutex); //working mutex unlock

    duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
    output = to_string(duration_ms) + "ms -- Pork#" + to_string(p.id) + ": enters the PACKER";
    print(output);
    output = to_string(duration_ms) + "ms -- PACKER: processing & packing Pork#" + to_string(p.id) + \
                " -- " + to_string(pack_ms) + "ms";
    print(output);
    wait(pack_ms);

    p.status = PACKED;
    packed_pork_cnt++;

    duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
    output = to_string(duration_ms) + "ms -- Pork#" + to_string(p.id) + ": leaves PACKER (Complete)";
    print(output);

    if(packed_pork_cnt == PORK_CNT){ //if all pork was packed, then finish
      finised = true;
    }
    pthread_mutex_lock(&working_mutex); //working mutex lock
    packer_working = false;
    pthread_mutex_unlock(&working_mutex); //working mutex unlock
    wait(sleep_ms);
  }
  pthread_mutex_lock(&working_mutex); //working mutex lock
  packer_working = true;
  pthread_mutex_unlock(&working_mutex); //working mutex unlock
  return NULL;
}
```

如果有豬肉的話就包一包，顯示完成

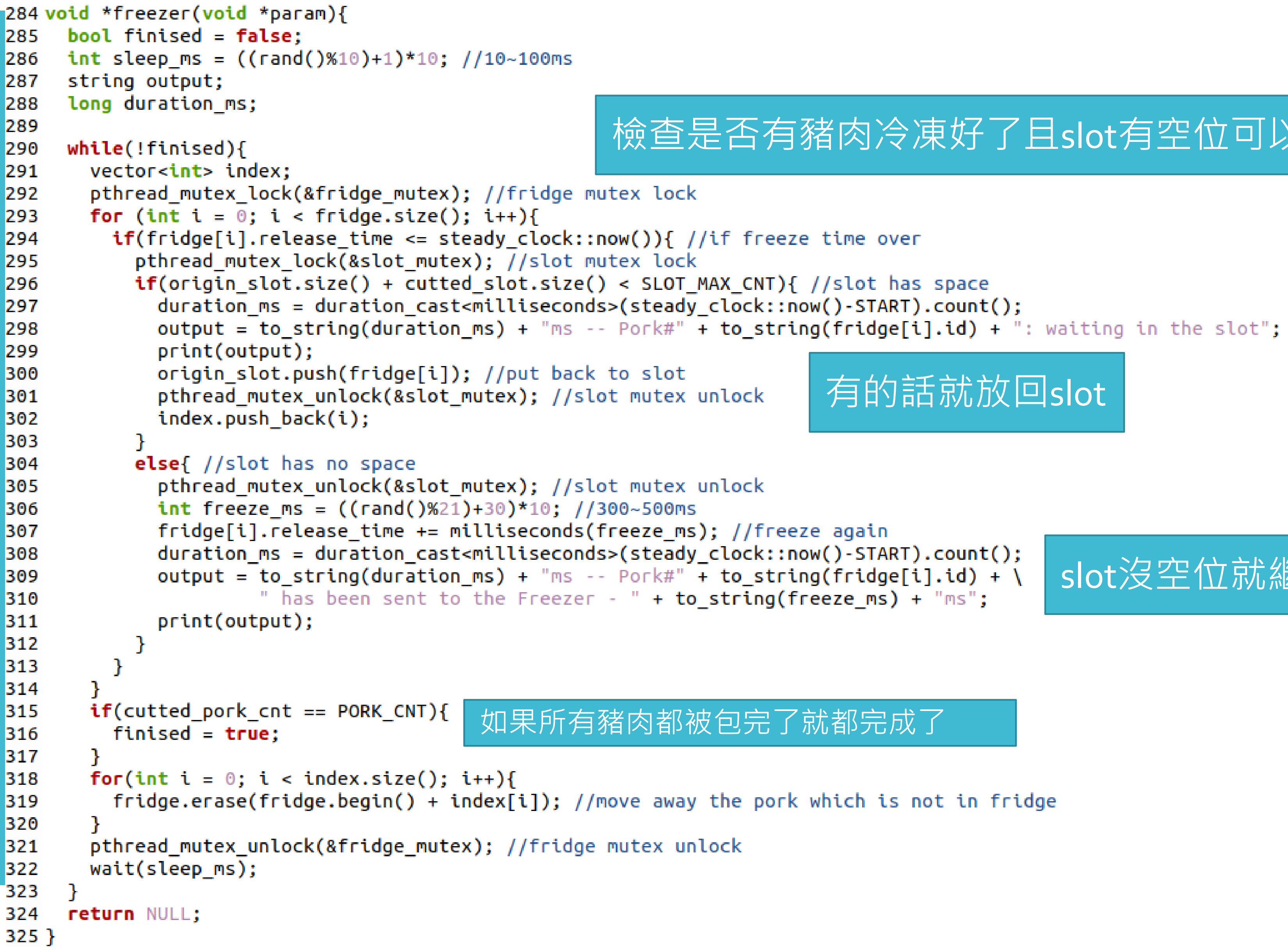
如果所有豬肉都被包完了就都完成了

利用working_mutex保護packer_working

利用working_mutex保護packer_working

程式碼運作說明 freezer

Freezer_thd負責冰豬肉，並用slot_mutex和fridge_mutex保護origin_slot和cutted_slot或fridge，以確保他們不會被同時讀寫



```
void *freezer(void *param){
  bool finised = false;
  int sleep_ms = ((rand()%10)+1)*10; //10~100ms
  string output;
  long duration_ms;

  while(!finised){
    vector<int> index;
    pthread_mutex_lock(&fridge_mutex); //fridge mutex lock
    for (int i = 0; i < fridge.size(); i++){
      if(fridge[i].release_time <= steady_clock::now()){ //if freeze time over
        pthread_mutex_lock(&slot_mutex); //slot mutex lock
        if(origin_slot.size() + cutted_slot.size() < SLOT_MAX_CNT){ //slot has space
          duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
          output = to_string(duration_ms) + "ms -- Pork#" + to_string(fridge[i].id) + ": waiting in the slot";
          print(output);
          origin_slot.push(fridge[i]); //put back to slot
          pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
          index.push_back(i);
        }
        else{ //slot has no space
          pthread_mutex_unlock(&slot_mutex); //slot mutex unlock
          int freeze_ms = ((rand()%21)+30)*10; //300~500ms
          fridge[i].release_time += milliseconds(freeze_ms); //freeze again
          duration_ms = duration_cast<milliseconds>(steady_clock::now()-START).count();
          output = to_string(duration_ms) + "ms -- Pork#" + to_string(fridge[i].id) + \
                   " has been sent to the Freezer - " + to_string(freeze_ms) + "ms";
          print(output);
        }
      }
    }
    if(cutted_pork_cnt == PORK_CNT){
      finised = true;
    }
    for(int i = 0; i < index.size(); i++){
      fridge.erase(fridge.begin() + index[i]); //move away the pork which is not in fridge
    }
    pthread_mutex_unlock(&fridge_mutex); //fridge mutex unlock
    wait(sleep_ms);
  }
  return NULL;
}
```

檢查是否有豬肉冷凍好了且slot有空位可以放了

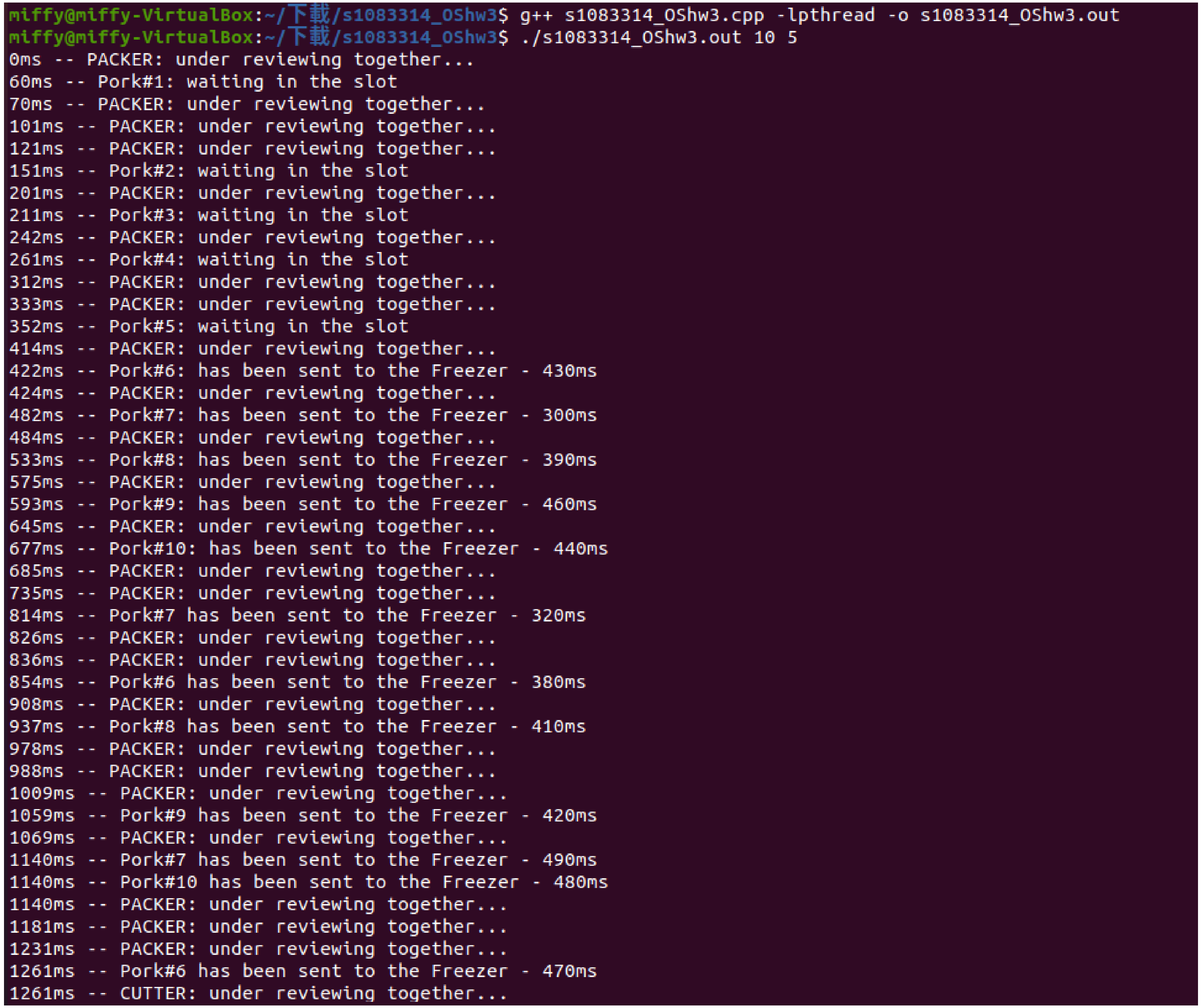
有的話就放回slot

slot沒空位就繼續冰

如果所有豬肉都被包完了就都完成了

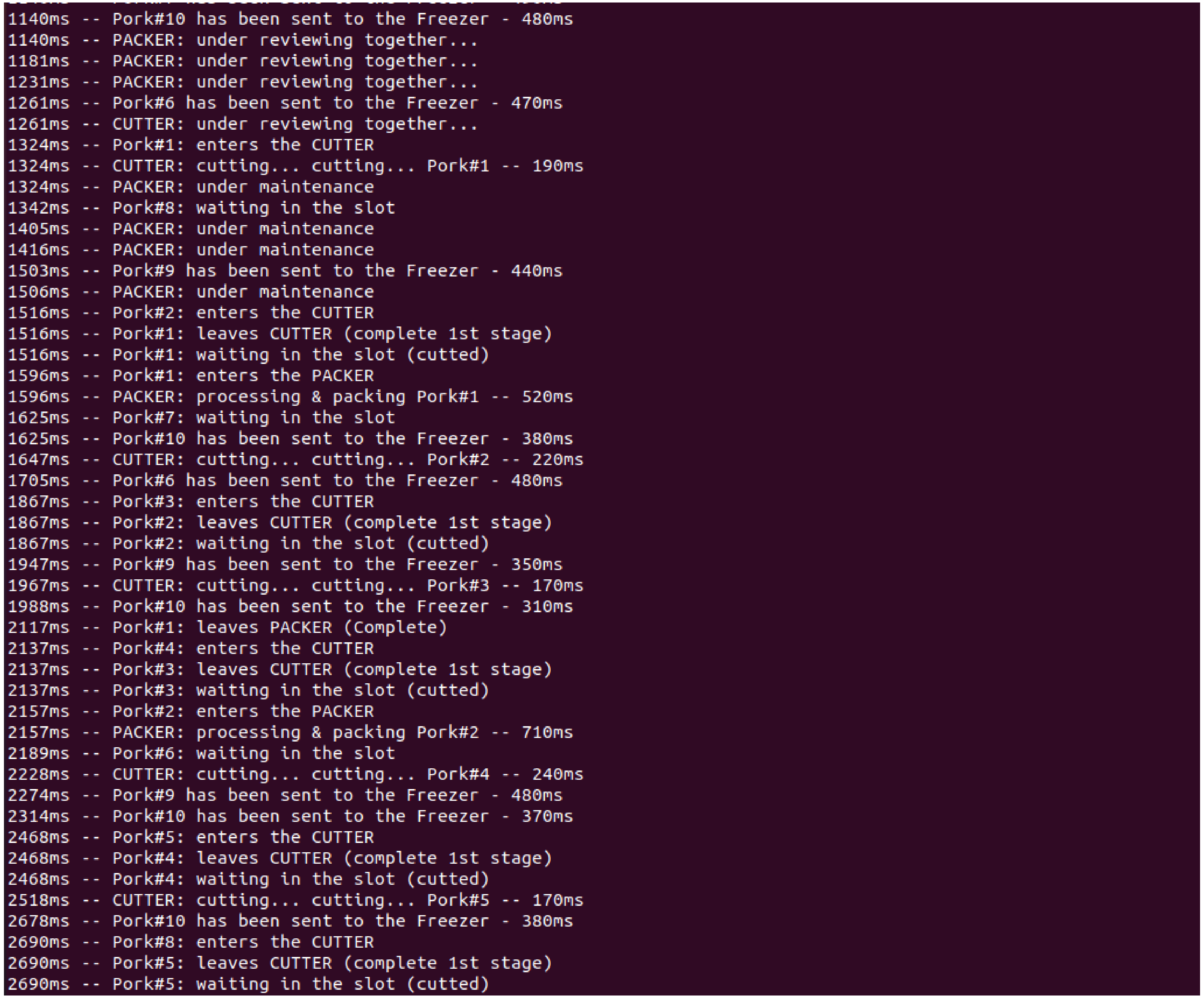
輸出結果展示1

以豬肉數量=10
　備料格數量=5 為例



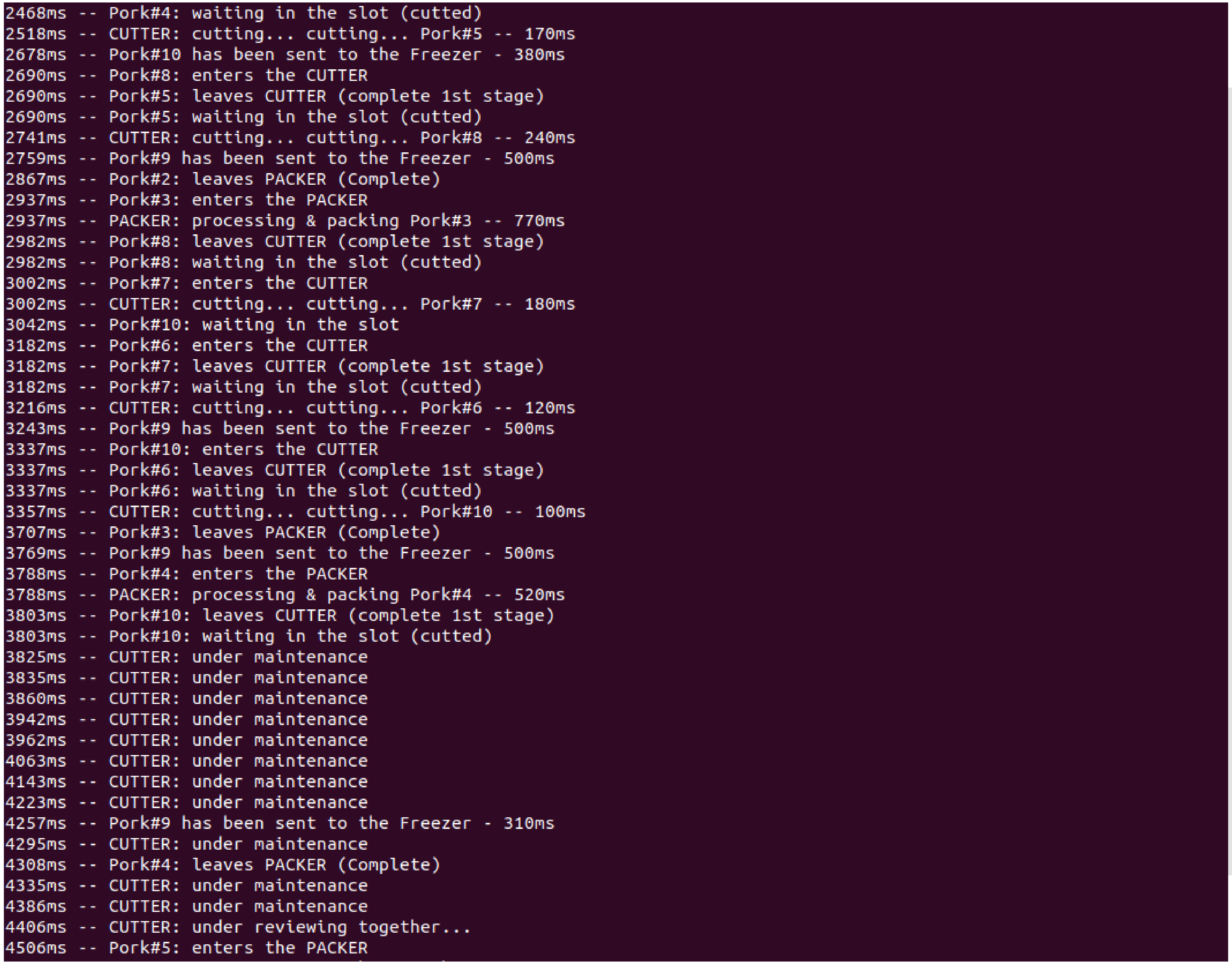
```
miffy@miffy-VirtualBox:~/下載/s1083314_OShw3$ g++ s1083314_OShw3.cpp -lpthread -o s1083314_OShw3.out
miffy@miffy-VirtualBox:~/下載/s1083314_OShw3$ ./s1083314_OShw3.out 10 5
0ms -- PACKER: under reviewing together...
60ms -- Pork#1: waiting in the slot
70ms -- PACKER: under reviewing together...
101ms -- PACKER: under reviewing together...
121ms -- PACKER: under reviewing together...
151ms -- Pork#2: waiting in the slot
201ms -- PACKER: under reviewing together...
211ms -- Pork#3: waiting in the slot
242ms -- PACKER: under reviewing together...
261ms -- Pork#4: waiting in the slot
312ms -- PACKER: under reviewing together...
333ms -- PACKER: under reviewing together...
352ms -- Pork#5: waiting in the slot
414ms -- PACKER: under reviewing together...
422ms -- Pork#6: has been sent to the Freezer - 430ms
424ms -- PACKER: under reviewing together...
482ms -- Pork#7: has been sent to the Freezer - 300ms
484ms -- PACKER: under reviewing together...
533ms -- Pork#8: has been sent to the Freezer - 390ms
575ms -- PACKER: under reviewing together...
593ms -- Pork#9: has been sent to the Freezer - 460ms
645ms -- PACKER: under reviewing together...
677ms -- Pork#10: has been sent to the Freezer - 440ms
685ms -- PACKER: under reviewing together...
735ms -- PACKER: under reviewing together...
814ms -- Pork#7 has been sent to the Freezer - 320ms
826ms -- PACKER: under reviewing together...
836ms -- PACKER: under reviewing together...
854ms -- Pork#6 has been sent to the Freezer - 380ms
908ms -- PACKER: under reviewing together...
937ms -- Pork#8 has been sent to the Freezer - 410ms
978ms -- PACKER: under reviewing together...
988ms -- PACKER: under reviewing together...
1009ms -- PACKER: under reviewing together...
1059ms -- Pork#9 has been sent to the Freezer - 420ms
1069ms -- PACKER: under reviewing together...
1140ms -- Pork#7 has been sent to the Freezer - 490ms
1140ms -- Pork#10 has been sent to the Freezer - 480ms
1140ms -- PACKER: under reviewing together...
1181ms -- PACKER: under reviewing together...
1231ms -- PACKER: under reviewing together...
1261ms -- Pork#6 has been sent to the Freezer - 470ms
1261ms -- CUTTER: under reviewing together...
```

輸出結果展示2



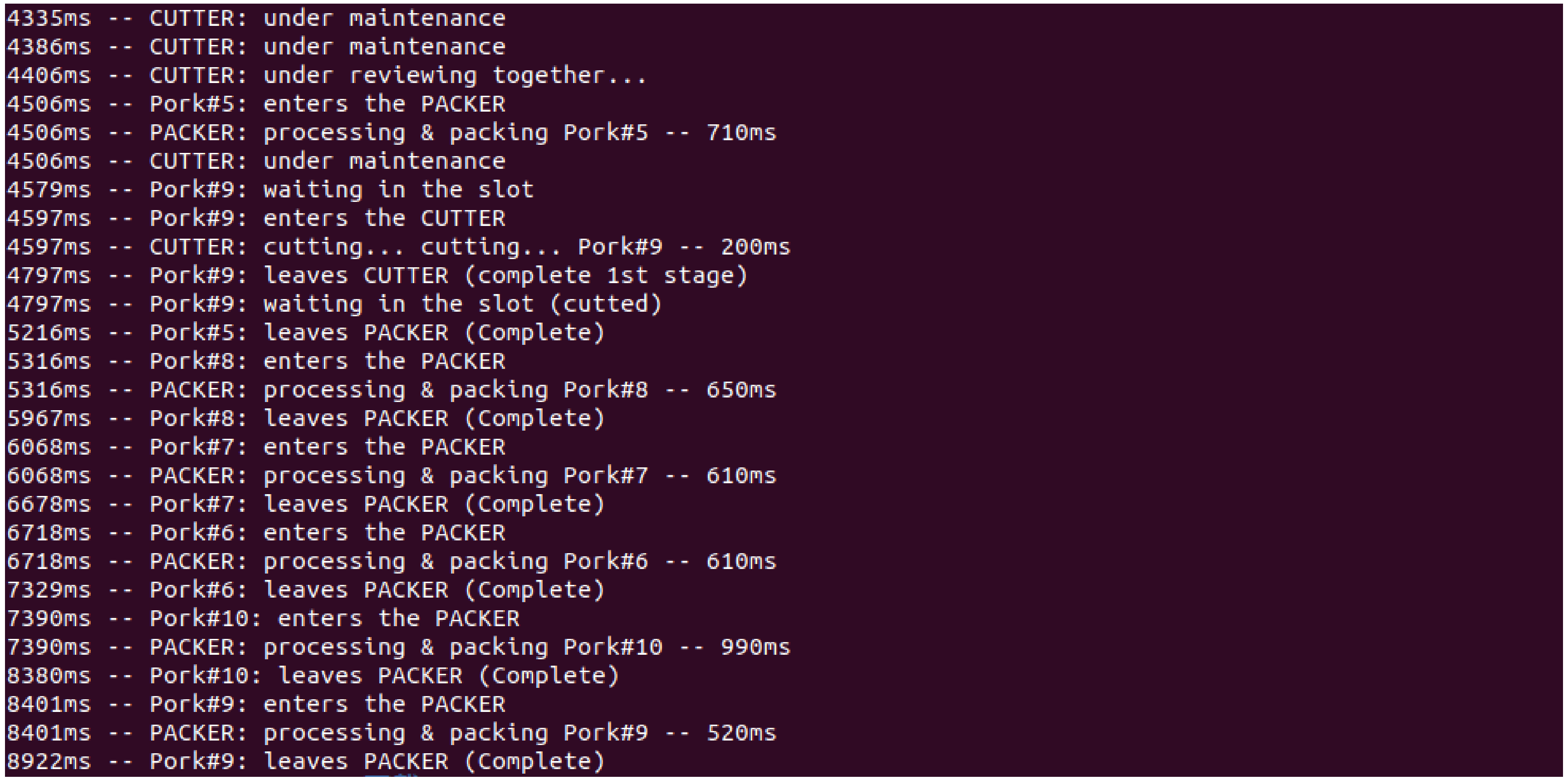
```
1140ms -- Pork#10 has been sent to the Freezer - 480ms
1140ms -- PACKER: under reviewing together...
1181ms -- PACKER: under reviewing together...
1231ms -- PACKER: under reviewing together...
1261ms -- Pork#6 has been sent to the Freezer - 470ms
1261ms -- CUTTER: under reviewing together...
1324ms -- Pork#1: enters the CUTTER
1324ms -- CUTTER: cutting... cutting... Pork#1 -- 190ms
1324ms -- PACKER: under maintenance
1342ms -- Pork#8: waiting in the slot
1405ms -- PACKER: under maintenance
1416ms -- PACKER: under maintenance
1503ms -- Pork#9 has been sent to the Freezer - 440ms
1506ms -- PACKER: under maintenance
1516ms -- Pork#2: enters the CUTTER
1516ms -- Pork#1: leaves CUTTER (complete 1st stage)
1516ms -- Pork#1: waiting in the slot (cutted)
1596ms -- Pork#1: enters the PACKER
1596ms -- PACKER: processing & packing Pork#1 -- 520ms
1625ms -- Pork#7: waiting in the slot
1625ms -- Pork#10 has been sent to the Freezer - 380ms
1647ms -- CUTTER: cutting... cutting... Pork#2 -- 220ms
1705ms -- Pork#6 has been sent to the Freezer - 480ms
1867ms -- Pork#3: enters the CUTTER
1867ms -- Pork#2: leaves CUTTER (complete 1st stage)
1867ms -- Pork#2: waiting in the slot (cutted)
1947ms -- Pork#9 has been sent to the Freezer - 350ms
1967ms -- CUTTER: cutting... cutting... Pork#3 -- 170ms
1988ms -- Pork#10 has been sent to the Freezer - 310ms
2117ms -- Pork#1: leaves PACKER (Complete)
2137ms -- Pork#4: enters the CUTTER
2137ms -- Pork#3: leaves CUTTER (complete 1st stage)
2137ms -- Pork#3: waiting in the slot (cutted)
2157ms -- Pork#2: enters the PACKER
2157ms -- PACKER: processing & packing Pork#2 -- 710ms
2189ms -- Pork#6: waiting in the slot
2228ms -- CUTTER: cutting... cutting... Pork#4 -- 240ms
2274ms -- Pork#9 has been sent to the Freezer - 480ms
2314ms -- Pork#10 has been sent to the Freezer - 370ms
2468ms -- Pork#5: enters the CUTTER
2468ms -- Pork#4: leaves CUTTER (complete 1st stage)
2468ms -- Pork#4: waiting in the slot (cutted)
2518ms -- CUTTER: cutting... cutting... Pork#5 -- 170ms
2678ms -- Pork#10 has been sent to the Freezer - 380ms
2690ms -- Pork#8: enters the CUTTER
2690ms -- Pork#5: leaves CUTTER (complete 1st stage)
2690ms -- Pork#5: waiting in the slot (cutted)
```

輸出結果展示3



```
2468ms -- Pork#4: waiting in the slot (cutted)
2518ms -- CUTTER: cutting... cutting... Pork#5 -- 170ms
2678ms -- Pork#10 has been sent to the Freezer - 380ms
2690ms -- Pork#8: enters the CUTTER
2690ms -- Pork#5: leaves CUTTER (complete 1st stage)
2690ms -- Pork#5: waiting in the slot (cutted)
2741ms -- CUTTER: cutting... cutting... Pork#8 -- 240ms
2759ms -- Pork#9 has been sent to the Freezer - 500ms
2867ms -- Pork#2: leaves PACKER (Complete)
2937ms -- Pork#3: enters the PACKER
2937ms -- PACKER: processing & packing Pork#3 -- 770ms
2982ms -- Pork#8: leaves CUTTER (complete 1st stage)
2982ms -- Pork#8: waiting in the slot (cutted)
3002ms -- Pork#7: enters the CUTTER
3002ms -- CUTTER: cutting... cutting... Pork#7 -- 180ms
3042ms -- Pork#10: waiting in the slot
3182ms -- Pork#6: enters the CUTTER
3182ms -- Pork#7: leaves CUTTER (complete 1st stage)
3182ms -- Pork#7: waiting in the slot (cutted)
3216ms -- CUTTER: cutting... cutting... Pork#6 -- 120ms
3243ms -- Pork#9 has been sent to the Freezer - 500ms
3337ms -- Pork#10: enters the CUTTER
3337ms -- Pork#6: leaves CUTTER (complete 1st stage)
3337ms -- Pork#6: waiting in the slot (cutted)
3357ms -- CUTTER: cutting... cutting... Pork#10 -- 100ms
3707ms -- Pork#3: leaves PACKER (Complete)
3769ms -- Pork#9 has been sent to the Freezer - 500ms
3788ms -- Pork#4: enters the PACKER
3788ms -- PACKER: processing & packing Pork#4 -- 520ms
3803ms -- Pork#10: leaves CUTTER (complete 1st stage)
3803ms -- Pork#10: waiting in the slot (cutted)
3825ms -- CUTTER: under maintenance
3835ms -- CUTTER: under maintenance
3860ms -- CUTTER: under maintenance
3942ms -- CUTTER: under maintenance
3962ms -- CUTTER: under maintenance
4063ms -- CUTTER: under maintenance
4143ms -- CUTTER: under maintenance
4223ms -- CUTTER: under maintenance
4257ms -- Pork#9 has been sent to the Freezer - 310ms
4295ms -- CUTTER: under maintenance
4308ms -- Pork#4: leaves PACKER (Complete)
4335ms -- CUTTER: under maintenance
4386ms -- CUTTER: under maintenance
4406ms -- CUTTER: under reviewing together...
4506ms -- Pork#5: enters the PACKER
```

輸出結果展示4



```
4335ms -- CUTTER: under maintenance
4386ms -- CUTTER: under maintenance
4406ms -- CUTTER: under reviewing together...
4506ms -- Pork#5: enters the PACKER
4506ms -- PACKER: processing & packing Pork#5 -- 710ms
4506ms -- CUTTER: under maintenance
4579ms -- Pork#9: waiting in the slot
4597ms -- Pork#9: enters the CUTTER
4597ms -- CUTTER: cutting... cutting... Pork#9 -- 200ms
4797ms -- Pork#9: leaves CUTTER (complete 1st stage)
4797ms -- Pork#9: waiting in the slot (cutted)
5216ms -- Pork#5: leaves PACKER (Complete)
5316ms -- Pork#8: enters the PACKER
5316ms -- PACKER: processing & packing Pork#8 -- 650ms
5967ms -- Pork#8: leaves PACKER (Complete)
6068ms -- Pork#7: enters the PACKER
6068ms -- PACKER: processing & packing Pork#7 -- 610ms
6678ms -- Pork#7: leaves PACKER (Complete)
6718ms -- Pork#6: enters the PACKER
6718ms -- PACKER: processing & packing Pork#6 -- 610ms
7329ms -- Pork#6: leaves PACKER (Complete)
7390ms -- Pork#10: enters the PACKER
7390ms -- PACKER: processing & packing Pork#10 -- 990ms
8380ms -- Pork#10: leaves PACKER (Complete)
8401ms -- Pork#9: enters the PACKER
8401ms -- PACKER: processing & packing Pork#9 -- 520ms
8922ms -- Pork#9: leaves PACKER (Complete)
```

實作總結

- 訊息皆可以正常印出
- 所有豬肉都能夠正確處理完成，沒有豬肉中途消失或重複處理
- 所有豬肉都有歷經正確的步驟
- 等待或操作的過程都會在有限時間內完成
- 豬肉的生產、切豬肉、 冷凍豬肉、包成粽子操作時間皆正確