Pork Zongzi maker!
歡樂包粽機

1083310 蘇建晟
Bonus 1、2、3都有實作

操作說明

- ./s1083310_OShw3.out 1. 2. 3. 4. 5. 6.

輸入的參數總計有6個，分別為 ：
1. 豬肉數量(Meat)
2. 備料台的格子數量(Slot)
3. 切割工廠數量(Cutter)
4. 包粽機數量(Packer)
5. 1：開啟BONUS 2；0：關閉BONUS 2 (冷凍庫的總時間長度為 1500ms)
6. 1：開啟BONUS 3；0：關閉BONUS 3 (豬肉的閒置時間為 3000ms)

操作說明

```
miro@miro-VirtualBox:~/s1083310_OShw3$ ./s1083310_OShw3.out 15 5 1 1 0 0
```

這種input就是代表著

1. 15塊豬肉(Meat)
2. 5個備料台的格子(Slot)
3. 1個切割工廠(Cutter)
4. 1個包粽機(Packer)
5. 關閉BONUS 2
6. 關閉BONUS 3

操作說明

```
miro@miro-VirtualBox:~/s1083310_OShw3$ ./s1083310_OShw3.out 20 6 3 2 1 1
```

這種input就是代表著

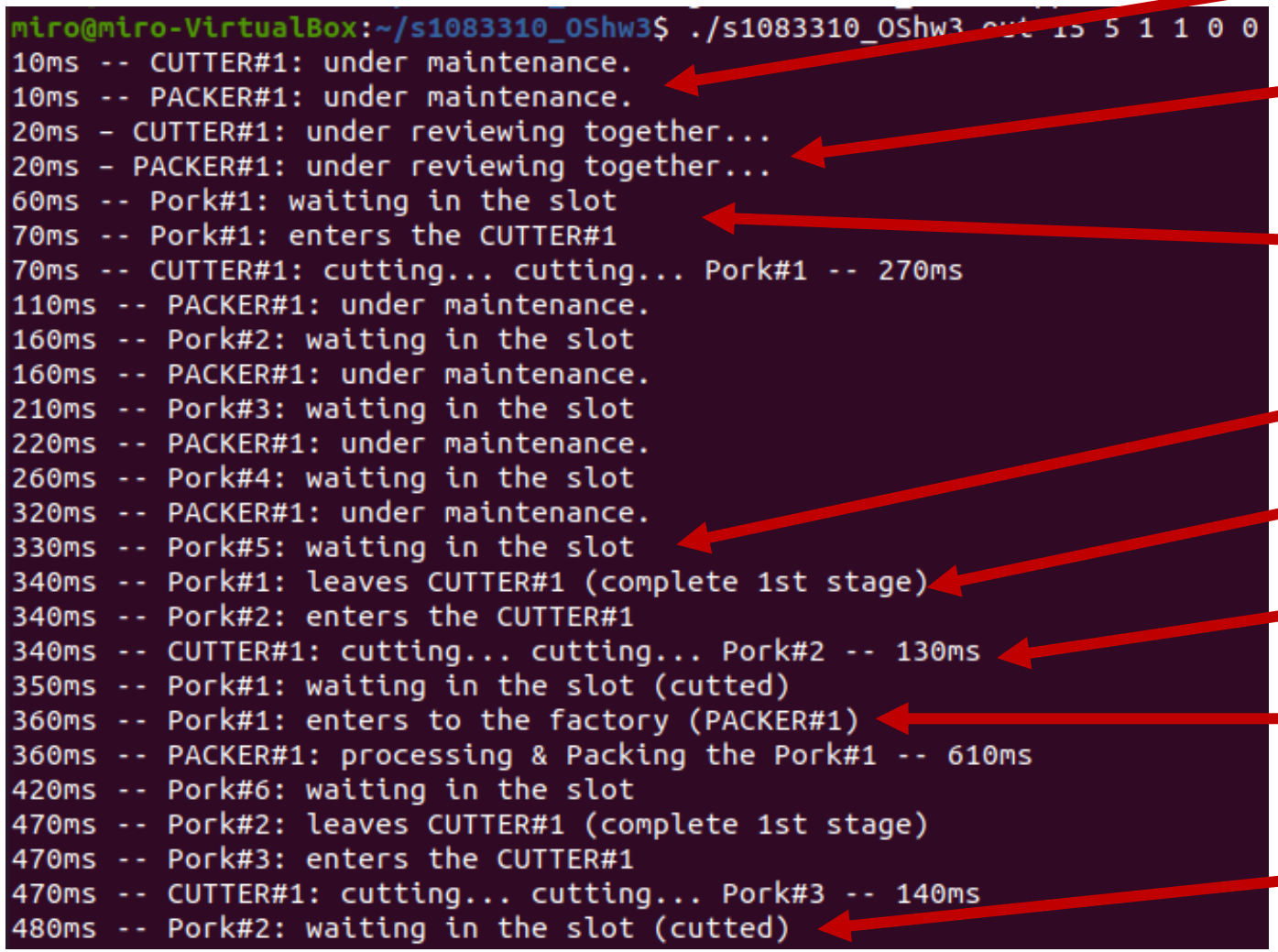
1. 20塊豬肉(Meat)
2. 6個備料台的格子(Slot)
3. 3個切割工廠(Cutter)
4. 2個包粽機(Packer)
5. 開啟BONUS 2
6. 開啟BONUS 3

部分流程說明

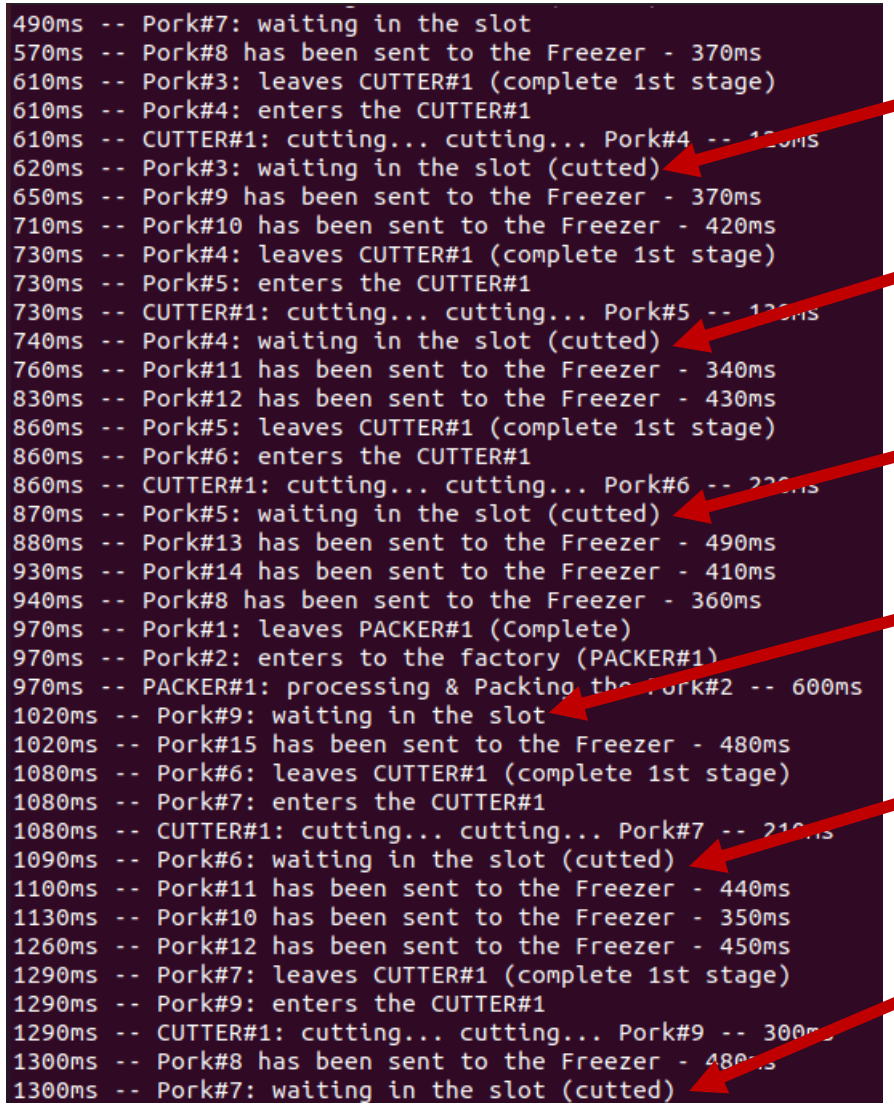
- **多工廠休眠模式**：當其一工廠(Cutter或Packer)進入維護模式時，會尋找另一種工廠(Packer或Cutter)是否也在維護模式，如果存在單個或複數個以上就進入檢討模式，不存在就單獨進入維護模式。
- **喚醒工廠**：在工廠休眠狀態時，有新的肉進入slot，工廠無法及時反應，需要在下個moment(10ms)才能執行。
- **Slot與Cutter**：假設slot總數為5，但是在Cutter閒置時， slot與在cutter內的肉總計數量可以達到6個(一個在工廠內等待slot空出空間)，當有必要時，會將cutter內的肉與slot的肉進行switch。

- 題目是這麼敘述的：**切割工廠備有Ｎ個備料格（slot, 預設為5個）**
  所以每增加一個切割工廠就會增加N個備料格(總計會有N * 切割工廠這麼多個)
- 後續我將第3、4頁的指令作為範例來解釋我的程式。

執行結果
(第3頁範例)



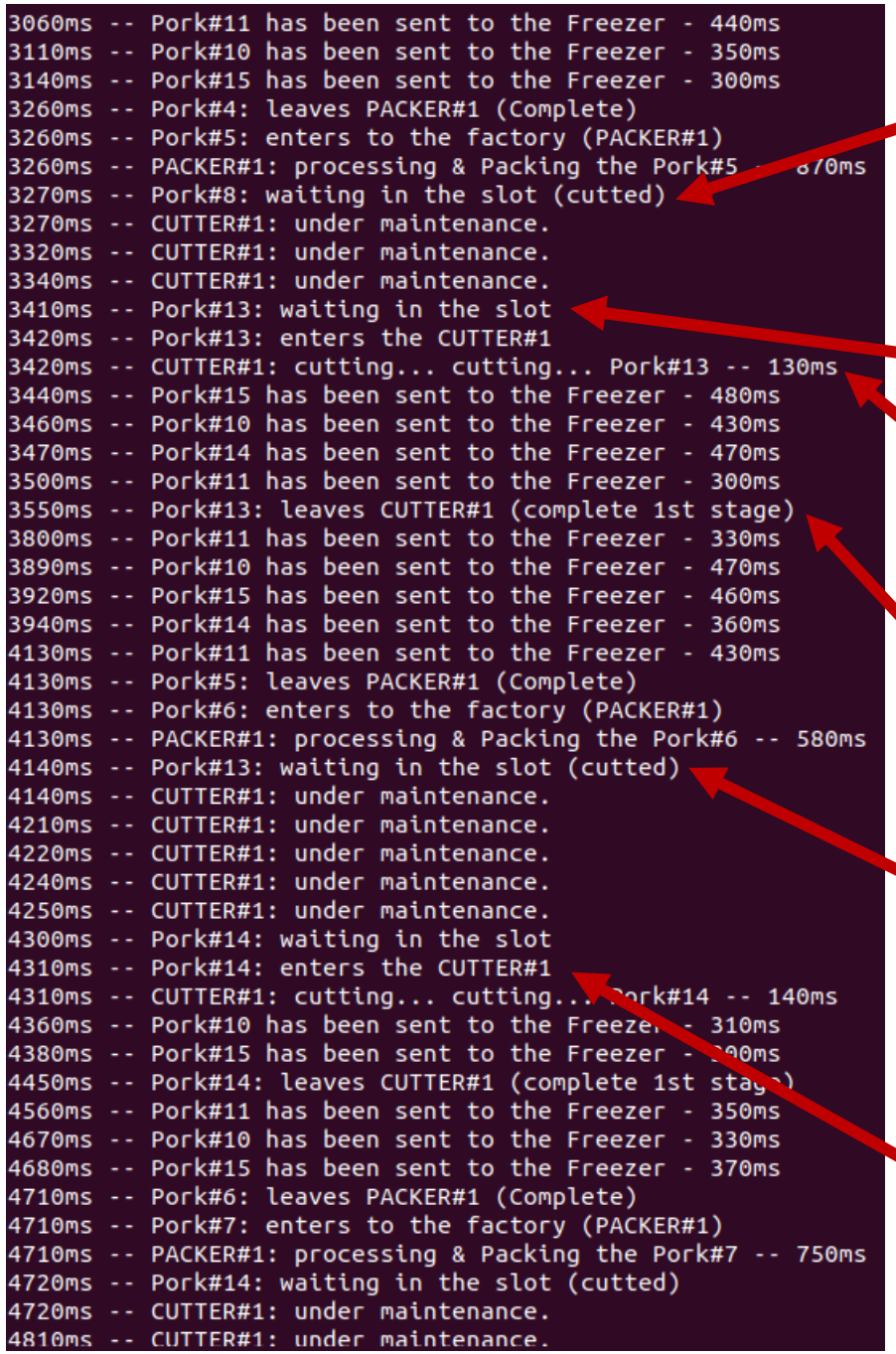
- 一開始Cutter與Packer都在維護模式。
- 發現另一個工廠也在維護模式，所以進入檢討模式。
- **Pork#1進入slot，10ms後，Cutter才開始運行。**
- 2~5號在備料台等，1號正在切。
- 1號完成(時間標記為70 + 270ms = 340ms)，標記切割完成。
- 1號離開Cutter，2號接著處理。
- 1號肉放回slot並經過10ms後，才能進入粽子工廠(喚醒工廠)並空出座位。
- 3號肉正在切，2、4~6在備料台等(2號等待進入Packer)



- 4號肉正在切，2、3、5~7在備料台等
(2、3號等待進入Packer)
- 5號肉正在切，2~4、6、7在備料台等
(2、3、4號等待進入Packer)
- 6號肉正在切，2~5、7在備料台等
(2、3、4、5號等待進入Packer)
- 1號肉完成包裝，2號肉進入粽子工廠，
6號肉正在切， 3~5、7、9在備料台等
(3、4、5號等待進入Packer)
- 7號肉正在切， 3~6、9在備料台等
(3、4、5、6號等待進入Packer)
- 9號肉正在切， 3~7在備料台等
(3、4、5、6、7號等待進入Packer)

```
1340ms -- Pork#14 has been sent to the Freezer - 450ms
1370ms -- Pork#13 has been sent to the Freezer - 400ms
1480ms -- Pork#10 has been sent to the Freezer - 400ms
1500ms -- Pork#15 has been sent to the Freezer - 430ms
1540ms -- Pork#11 has been sent to the Freezer - 360ms
1570ms -- Pork#2: leaves PACKER#1 (Complete)
1570ms -- Pork#3: enters to the factory (PACKER#1)
1570ms -- PACKER#1: processing & Packing the Pork#3 -- 800ms
1590ms -- Pork#9: leaves CUTTER#1 (complete 1st stage)
1600ms -- Pork#9: waiting in the slot (cutted)
1600ms -- CUTTER#1: under maintenance.
1630ms -- CUTTER#1: under maintenance.
1670ms -- CUTTER#1: under maintenance.
1710ms -- Pork#12: waiting in the slot
1720ms -- Pork#12: enters the CUTTER#1
1720ms -- CUTTER#1: cutting... cutting... Pork#12 -- 160ms
1770ms -- Pork#13 has been sent to the Freezer - 400ms
1780ms -- Pork#8 has been sent to the Freezer - 410ms
1790ms -- Pork#14 has been sent to the Freezer - 440ms
1880ms -- Pork#10 has been sent to the Freezer - 400ms
1880ms -- Pork#12: leaves CUTTER#1 (complete 1st stage)
1900ms -- Pork#11 has been sent to the Freezer - 460ms
1930ms -- Pork#15 has been sent to the Freezer - 350ms
2170ms -- Pork#13 has been sent to the Freezer - 430ms
2190ms -- Pork#8 has been sent to the Freezer - 300ms
2230ms -- Pork#14 has been sent to the Freezer - 460ms
2280ms -- Pork#10 has been sent to the Freezer - 330ms
2280ms -- Pork#15 has been sent to the Freezer - 490ms
2360ms -- Pork#11 has been sent to the Freezer - 360ms
2370ms -- Pork#3: leaves PACKER#1 (Complete)
2370ms -- Pork#4: enters to the factory (PACKER#1)
2370ms -- PACKER#1: processing & Packing the Pork#4 -- 800ms
2380ms -- Pork#12: waiting in the slot (cutted)
2380ms -- CUTTER#1: under maintenance.
2390ms -- CUTTER#1: under maintenance.
2400ms -- CUTTER#1: under maintenance.
2450ms -- CUTTER#1: under maintenance.
2490ms -- Pork#8: waiting in the slot
2500ms -- Pork#8: enters the CUTTER#1
2500ms -- CUTTER#1: cutting... cutting... Pork#8 -- 160ms
2600ms -- Pork#13 has been sent to the Freezer - 400ms
2610ms -- Pork#10 has been sent to the Freezer - 500ms
2660ms -- Pork#8: leaves CUTTER#1 (complete 1st stage)
2690ms -- Pork#14 has been sent to the Freezer - 310ms
2720ms -- Pork#11 has been sent to the Freezer - 340ms
2770ms -- Pork#15 has been sent to the Freezer - 370ms
3000ms -- Pork#13 has been sent to the Freezer - 410ms
3000ms -- Pork#14 has been sent to the Freezer - 470ms
```

- 2號肉完成包裝，3號肉進入粽子工廠，
4~7、9在備料台等
(4~7、9號等待進入Packer)
- 因為沒肉切，所以Cutter進入維護模式
- 12號肉正在切，4~7、9在備料台等
(4、5、6、7、9號等待進入Packer)
- 4~7、9在備料台等，**12完成，但是因為slot已滿所以12在cutter內等待空間**
(4~7、9、12號等待進入Packer)
- 3號肉完成包裝，4號肉進入粽子工廠，
此時12才回到slot
5~7、9、12在備料台等
(5~7、9、12號等待進入Packer)
- **執行一次switch(12暫放到cutter，8進入slot，然後2個執行switch)**
- 5~7、9、12在備料台等， **8號肉完成**
(5~7、9、12號等待進入Packer)



- 4號肉完成包裝，5號肉進入粽子工廠，
此時8才回到slot
6、7、9、12、8在備料台等
(6、7、9、12、8號等待進入Packer)

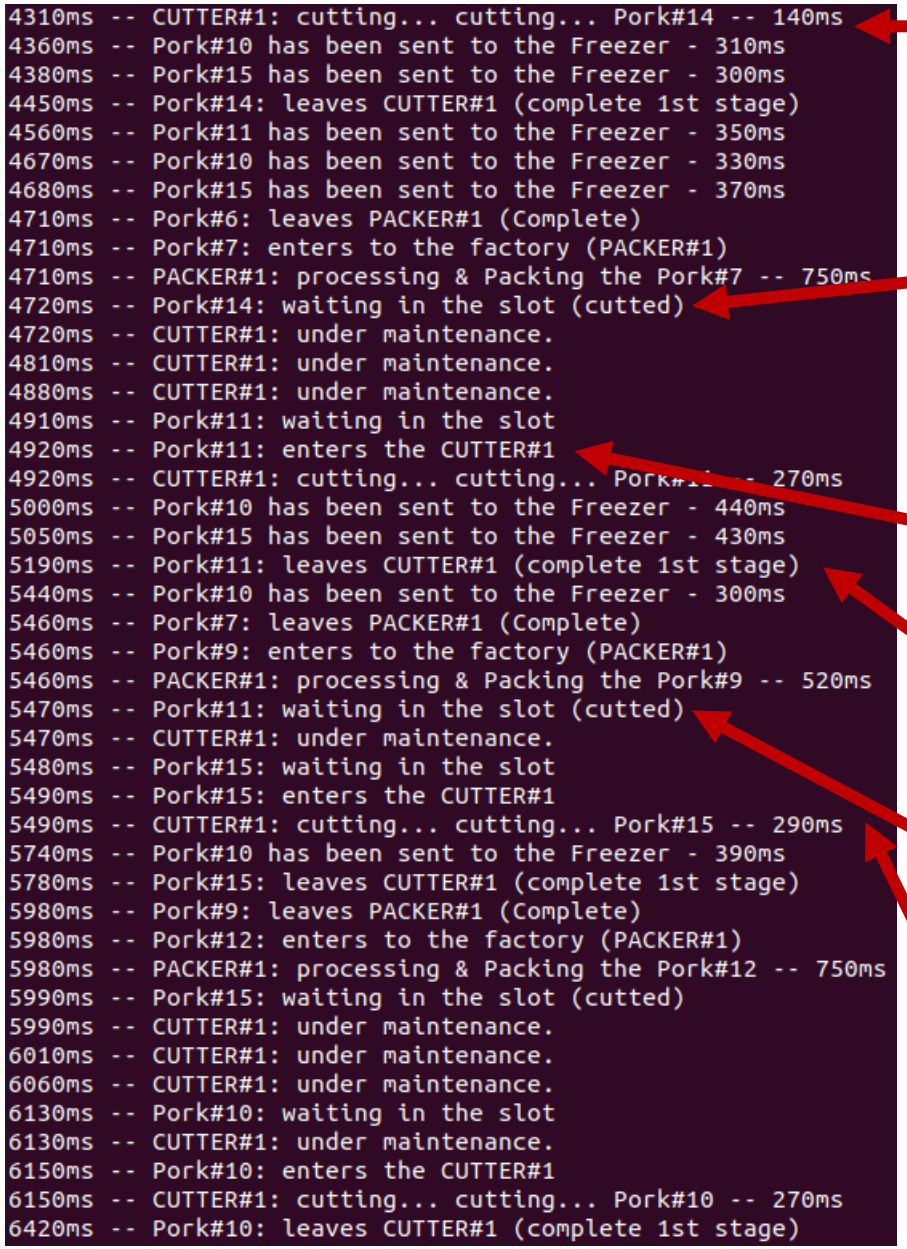
- **執行一次switch(8暫放到cutter，13進入slot，然後2個執行switch)**

- 13號肉正在切
6、7、9、12、8在備料台等
(6、7、9、12、8號等待進入Packer)

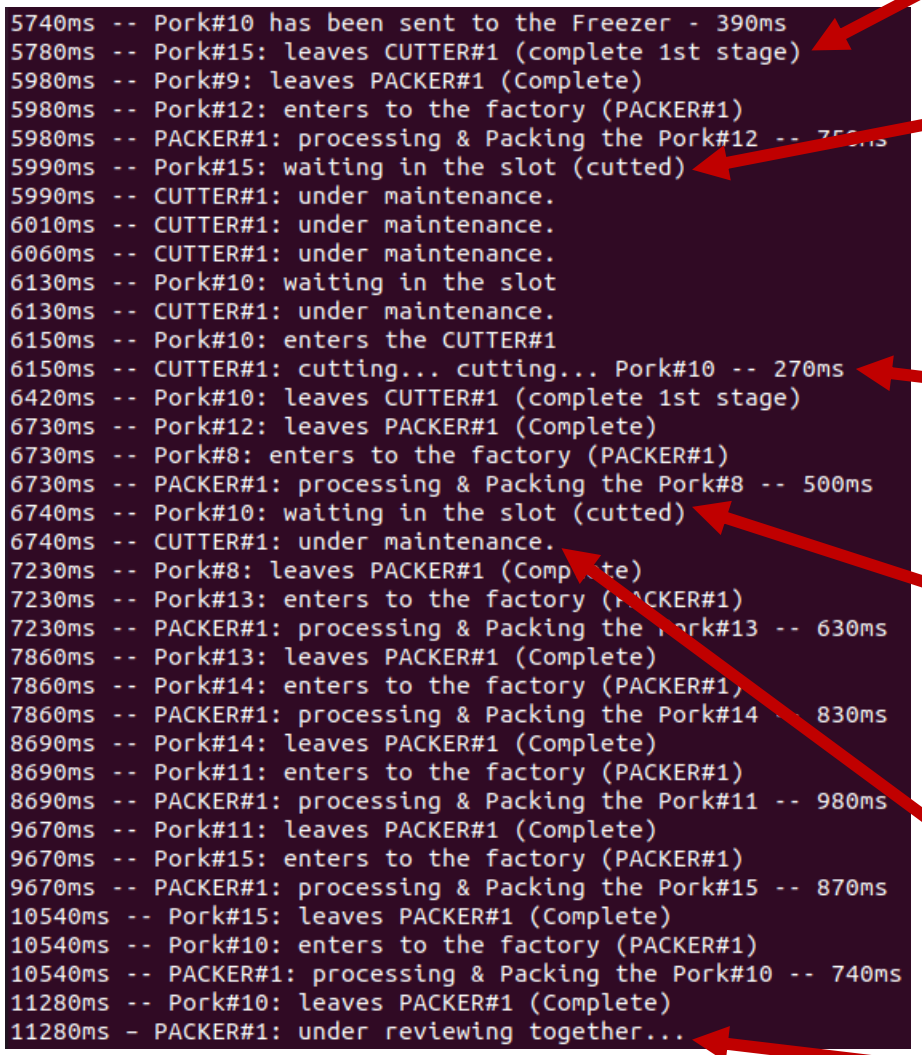
- **13完成，但是因為slot已滿所以13在cutter內等待空間**

- 5號肉完成包裝，6號肉進入粽子工廠，
此時13才回到slot
7、9、12、8、13在備料台等
(7、9、12、8、13號等待進入Packer)

- **執行一次switch(13暫放到cutter，14進入slot，然後2個執行switch)**



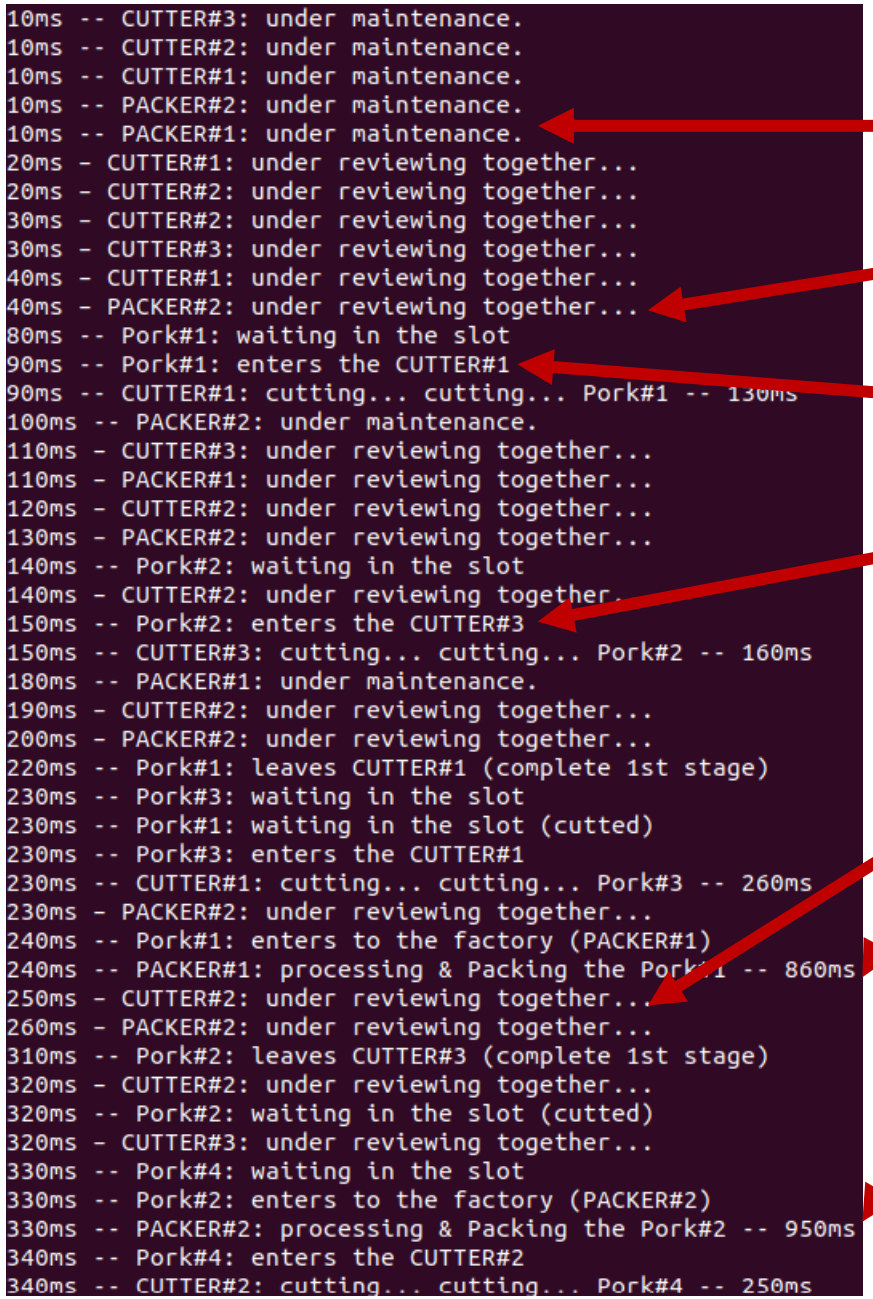
- 14號肉正在切
  7、9、12、8、13在備料台等
  (7、9、12、8、13號等待進入Packer)
- 6號肉完成包裝，7號肉進入粽子工廠，
  此時14才回到slot
  9、12、8、13、14在備料台等
  (9、12、8、13、14號等待進入Packer)
- **執行一次switch(14暫放到cutter，11進入slot，然後2個執行switch)**
- **11完成，但是因為slot已滿所以11在cutter內等待空間**
- 7號肉完成包裝，9號肉進入粽子工廠，
  此時11才回到slot
  12、8、13、14、11在備料台等
  (12、8、13、14、11號等待進入Packer)
- 15號肉正在切 **(利用switch)**
  12、8、13、14、11在備料台等
  (12、8、13、14、11號等待進入Packer)



- **15完成，但是因為slot已滿所以15在cutter內等待空間**
- 9號肉完成包裝，12號肉進入粽子工廠，
此時15才回到slot
8、13、14、11、15在備料台等
(8、13、14、11、15號等待進入Packer)
- 10號肉正在切 **(利用switch)**
8、13、14、11、15在備料台等
(8、13、14、11、15號等待進入Packer)
- 12號肉完成包裝，8號肉進入粽子工廠，
此時15才回到slot
13、14、11、15、10在備料台等
(13、14、11、15、10號等待進入Packer)
- 全部的肉都切完了，所以cutter離線
(不再更新)
- 後續依序進入粽子工廠，Packer運作結束

執行結果
(第4頁範例)

這個範例只針對BONUS的部分進行講解

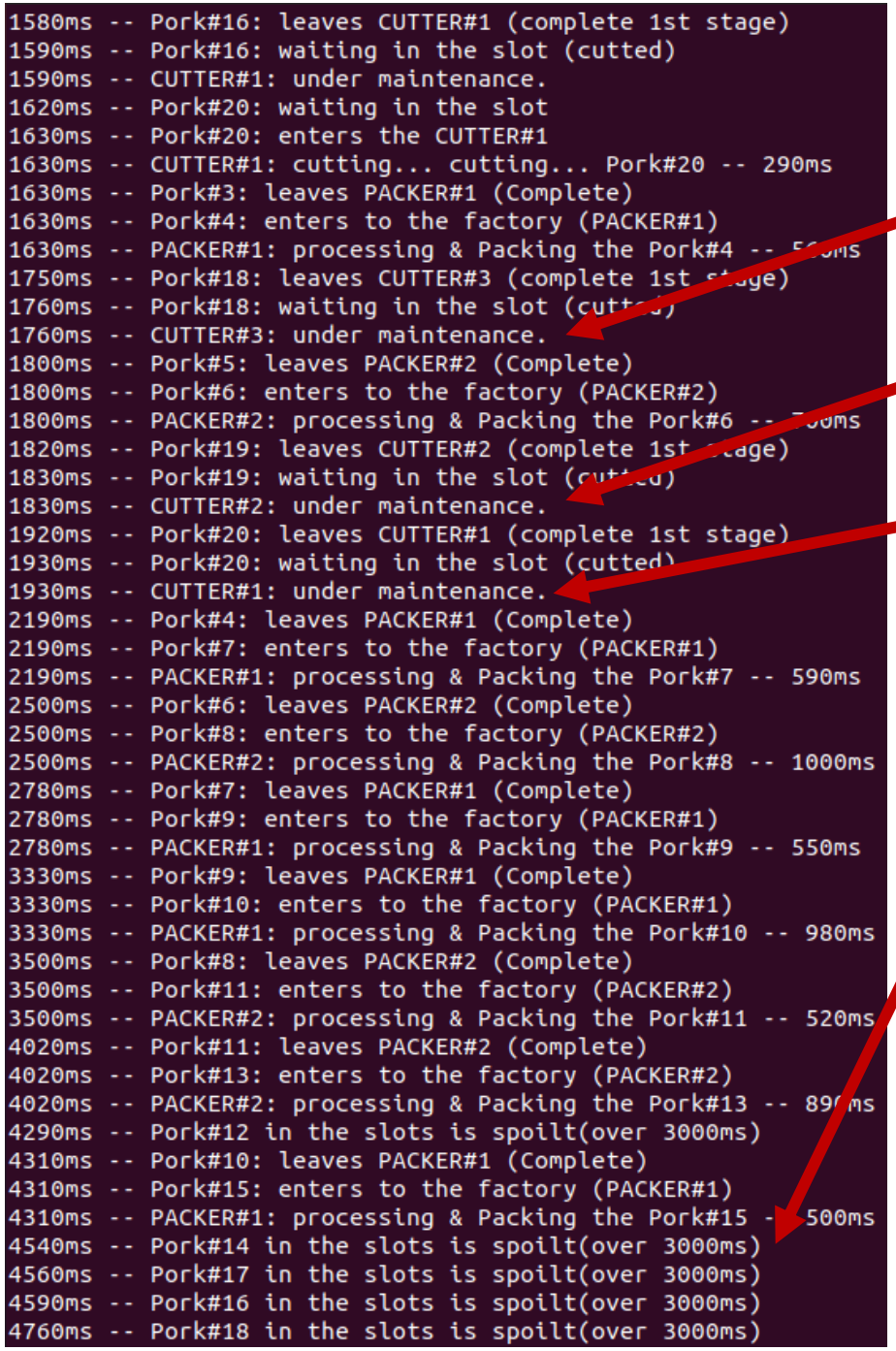


- 一開始Cutter與Packer都在維護模式。
- 發現另一種工廠也有在維護模式，所以進入檢討模式。
- **Pork#1進入slot，10ms後Cutter#1才開始運行**
- **Pork#2進入slot，10ms後Cutter#3才開始運行**
- **Pork#1回到slot，10ms後Packer#1才開始運行**
- **Cutter#2一直處於檢討模式**
  (肉都被其他cutter搶走)
- **Pork#2回到slot，10ms後Packer#2才開始運行**

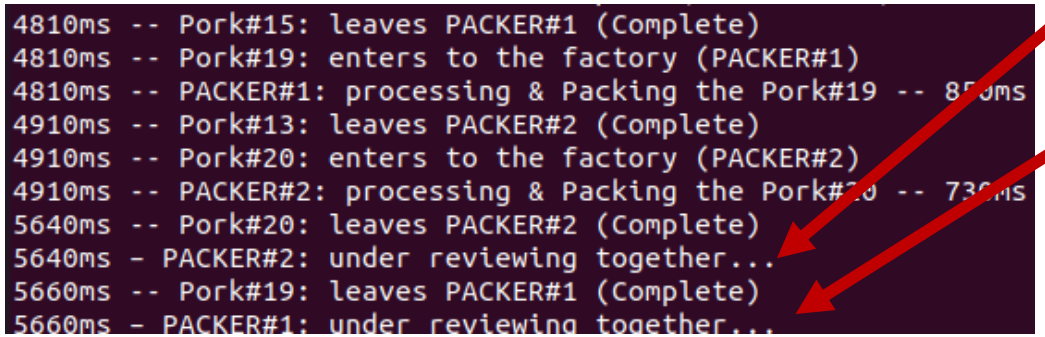
```
400ms -- Pork#5: waiting in the slot
400ms -- Pork#5: enters the CUTTER#3
400ms -- CUTTER#3: cutting... cutting... Pork#5 -- 170ms
490ms -- Pork#3: leaves CUTTER#1 (complete 1st stage)
500ms -- Pork#6: waiting in the slot
500ms -- Pork#3: waiting in the slot (cutted)
500ms -- Pork#6: enters the CUTTER#1
500ms -- CUTTER#1: cutting... cutting... Pork#6 -- 100ms
570ms -- Pork#5: leaves CUTTER#3 (complete 1st stage)
580ms -- Pork#5: waiting in the slot (cutted)
580ms -- CUTTER#3: under maintenance.
590ms -- Pork#7: waiting in the slot
590ms -- Pork#4: leaves CUTTER#2 (complete 1st stage)
590ms -- Pork#7: enters the CUTTER#2
590ms -- CUTTER#2: cutting... cutting... Pork#7 -- 170ms
600ms -- Pork#6: leaves CUTTER#1 (complete 1st stage)
600ms -- Pork#4: waiting in the slot (cutted)
610ms -- Pork#6: waiting in the slot (cutted)
610ms -- CUTTER#1: under maintenance.
650ms -- CUTTER#1: under maintenance.
660ms -- CUTTER#3: under maintenance.
670ms -- Pork#8: waiting in the slot
680ms -- Pork#8: enters the CUTTER#1
680ms -- CUTTER#1: cutting... cutting... Pork#8 -- 140ms
730ms -- CUTTER#3: under maintenance.
760ms -- Pork#9: waiting in the slot
760ms -- Pork#7: leaves CUTTER#2 (complete 1st stage)
760ms -- Pork#9: enters the CUTTER#2
760ms -- CUTTER#2: cutting... cutting... Pork#9 -- 210ms
770ms -- Pork#7: waiting in the slot (cutted)
820ms -- Pork#8: leaves CUTTER#1 (complete 1st stage)
820ms -- CUTTER#3: under maintenance.
830ms -- Pork#8: waiting in the slot (cutted)
830ms -- CUTTER#1: under maintenance.
850ms -- Pork#10: waiting in the slot
860ms -- Pork#10: enters the CUTTER#1
860ms -- CUTTER#1: cutting... cutting... Pork#10 -- 190ms
910ms -- CUTTER#3: under maintenance.
950ms -- Pork#11: waiting in the slot
960ms -- Pork#11: enters the CUTTER#3
960ms -- CUTTER#3: cutting... cutting... Pork#11 -- 280ms
970ms -- Pork#9: leaves CUTTER#2 (complete 1st stage)
980ms -- Pork#9: waiting in the slot (cutted)
980ms -- CUTTER#2: under maintenance.
```

Pork#7進入slot
10ms後Cutter#2
才開始運行

```
1010ms -- Pork#12: waiting in the slot
1020ms -- Pork#12: enters the CUTTER#2
1020ms -- CUTTER#2: cutting... cutting... Pork#12 -- 260ms
1050ms -- Pork#10: leaves CUTTER#1 (complete 1st stage)
1060ms -- Pork#10: waiting in the slot (cutted)
1060ms -- CUTTER#1: under maintenance.
1080ms -- CUTTER#1: under maintenance.
1090ms -- Pork#13: waiting in the slot
1100ms -- Pork#13: enters the CUTTER#1
1100ms -- CUTTER#1: cutting... cutting... Pork#13 -- 160ms
1100ms -- Pork#1: leaves PACKER#1 (Complete)
1100ms -- Pork#3: enters to the factory (PACKER#1)
1100ms -- PACKER#1: processing & Packing the Pork#3 -- 530m
1190ms -- Pork#14: waiting in the slot
1240ms -- Pork#11: leaves CUTTER#3 (complete 1st stage)
1240ms -- Pork#14: enters the CUTTER#3
1240ms -- CUTTER#3: cutting... cutting... Pork#14 -- 290ms
1250ms -- Pork#11: waiting in the slot (cutted)
1260ms -- Pork#13: leaves CUTTER#1 (complete 1st stage)
1270ms -- Pork#13: waiting in the slot (cutted)
1270ms -- CUTTER#1: under maintenance.
1280ms -- Pork#15: waiting in the slot
1280ms -- Pork#12: leaves CUTTER#2 (complete 1st stage)
1280ms -- Pork#15: enters the CUTTER#2
1280ms -- CUTTER#2: cutting... cutting... Pork#15 -- 130ms
1280ms -- Pork#2: leaves PACKER#2 (Complete)
1280ms -- Pork#5: enters to the factory (PACKER#2)
1280ms -- PACKER#2: processing & Packing the Pork#5 -- 520m
1290ms -- CUTTER#1: under maintenance.
1290ms -- Pork#12: waiting in the slot (cutted)
1350ms -- Pork#16: waiting in the slot
1360ms -- Pork#16: enters the CUTTER#1
1360ms -- CUTTER#1: cutting... cutting... Pork#16 -- 220ms
1400ms -- Pork#17: waiting in the slot
1410ms -- Pork#15: leaves CUTTER#2 (complete 1st stage)
1410ms -- Pork#17: enters the CUTTER#2
1410ms -- CUTTER#2: cutting... cutting... Pork#17 -- 140ms
1420ms -- Pork#15: waiting in the slot (cutted)
1450ms -- Pork#18: waiting in the slot
1530ms -- Pork#19: waiting in the slot
1530ms -- Pork#14: leaves CUTTER#3 (complete 1st stage)
1530ms -- Pork#18: enters the CUTTER#3
1530ms -- CUTTER#3: cutting... cutting... Pork#18 -- 220ms
1540ms -- Pork#14: waiting in the slot (cutted)
1550ms -- Pork#17: leaves CUTTER#2 (complete 1st stage)
1550ms -- Pork#19: enters the CUTTER#2
1550ms -- CUTTER#2: cutting... cutting... Pork#19 -- 270ms
1560ms -- Pork#17: waiting in the slot (cutted)
```



- 全部的肉都切完了，所以cutter#3離線(不再更新)
- 全部的肉都切完了，所以cutter#2離線(不再更新)
- 全部的肉都切完了，所以cutter#1離線(不再更新)
- **BONUS 3 豬肉壞掉事件，Pork#14在1540ms進入slot，經過3000ms後，都沒有被使用，所以觸發損壞事件**



- 全部的肉都切完了，所以Packer#2離線(不再更新)
- 全部的肉都切完了，所以Packer#1離線(不再更新)

執行結果
(BONUS 2)

前一個例子沒有示範出BONUS 2，所以我在這邊舉個範例

```
miro@miro-VirtualBox:~/s1083310_OShw3$ ./s1083310_OShw3.out 20 6 1 2 1 1
```

```
2050ms -- Pork#6: enters to the factory (PACKER#1)
2050ms -- PACKER#1: processing & Packing the Pork#6 -- 520ms
2080ms -- Pork#11: waiting in the slot
2130ms -- Pork#19 has been sent to the Freezer - 320ms
2200ms -- Pork#12 has been sent to the Freezer - 420ms
2220ms -- Pork#18 has been sent to the Freezer - 340ms
2230ms -- Pork#17 has been sent to the Freezer - 420ms
2260ms -- Pork#10: leaves CUTTER#1 (complete 1st stage)
2260ms -- Pork#16: enters the CUTTER#1
2260ms -- CUTTER#1: cutting... cutting... Pork#16 -- 270ms
2270ms -- Pork#10: waiting in the slot (cutted)
2350ms -- Pork#15 has been sent to the Freezer - 470ms
2410ms -- Pork#5: leaves PACKER#2 (Complete)
2410ms -- Pork#7: enters to the factory (PACKER#2)
2410ms -- PACKER#2: processing & Packing the Pork#7 -- 720ms
2420ms -- Pork#14: waiting in the slot
2450ms -- Pork#19 has been sent to the Freezer - 450ms
2510ms -- Pork#20 has been sent to the Freezer - 380ms
2530ms -- Pork#16: leaves CUTTER#1 (complete 1st stage)
2530ms -- Pork#11: enters the CUTTER#1
2530ms -- CUTTER#1: cutting... cutting... Pork#11 -- 150ms
2540ms -- Pork#16: waiting in the slot (cutted)
2560ms -- Pork#18 has been sent to the Freezer - 390ms
2570ms -- Pork#6: leaves PACKER#1 (Complete)
2570ms -- Pork#8: enters to the factory (PACKER#1)
2570ms -- PACKER#1: processing & Packing the Pork#8 -- 1000ms
2620ms -- Pork#12: waiting in the slot
2650ms -- Pork#17 has been sent to the Freezer - 420ms
2680ms -- Pork#11: leaves CUTTER#1 (complete 1st stage)
2680ms -- Pork#14: enters the CUTTER#1
2680ms -- CUTTER#1: cutting... cutting... Pork#14 -- 210ms
2690ms -- Pork#11: waiting in the slot (cutted)
2820ms -- Pork#15: exits freezer and waits beside the slot
2890ms -- Pork#20 has been sent to the Freezer - 350ms
2890ms -- Pork#14: leaves CUTTER#1 (complete 1st stage)
2890ms -- Pork#12: enters the CUTTER#1
2890ms -- CUTTER#1: cutting... cutting... Pork#12 -- 100ms
2900ms -- Pork#19 has been sent to the Freezer - 350ms
2900ms -- Pork#14: waiting in the slot (cutted)
2950ms -- Pork#18: exits freezer and waits beside the slot
2990ms -- Pork#12: leaves CUTTER#1 (complete 1st stage)
3070ms -- Pork#17: exits freezer and waits beside the slot
3130ms -- Pork#7: leaves PACKER#2 (Complete)
3130ms -- Pork#9: enters to the factory (PACKER#2)
3130ms -- PACKER#2: processing & Packing the Pork#9 -- 890ms
3140ms -- Pork#12: waiting in the slot (cutted)
3140ms -- CUTTER#1: under maintenance.
3140ms -- Pork#15: waiting in the slot(BONUSII)
```

- 發現Pork#15在freezer待超過1500ms
(後續將優先排入slot)

- slot有空間，Pork#15優先排入
(雖說優先，但依舊僅次於在cutter內的肉，
這麼設定是以免肉在cutter內卡死)

```
1100ms -- Pork#1: leaves PACKER#1 (Complete)
1100ms -- Pork#4: enters to the factory (PACKER#1)
1100ms -- PACKER#1: processing & Packing the Pork#4 -- 950ms
1160ms -- Pork#6: leaves CUTTER#1 (complete 1st stage)
1160ms -- Pork#7: enters the CUTTER#1
1160ms -- CUTTER#1: cutting... cutting... Pork#7 -- 250ms
1170ms -- Pork#6: waiting in the slot (cutted)
1180ms -- Pork#10: waiting in the slot
1190ms -- Pork#14 has been sent to the Freezer - 370ms
1280ms -- Pork#15 has been sent to the Freezer - 300ms
1300ms -- Pork#11 has been sent to the Freezer - 420ms
1350ms -- Pork#12 has been sent to the Freezer - 370ms
1350ms -- Pork#16 has been sent to the Freezer - 440ms
1400ms -- Pork#17 has been sent to the Freezer - 470ms
1410ms -- Pork#7: leaves CUTTER#1 (complete 1st stage)
1410ms -- Pork#8: enters the CUTTER#1
1410ms -- CUTTER#1: cutting... cutting... Pork#8 -- 180ms
1420ms -- Pork#7: waiting in the slot (cutted)
1450ms -- Pork#18 has been sent to the Freezer - 340ms
1530ms -- Pork#19 has been sent to the Freezer - 300ms
1560ms -- Pork#14 has been sent to the Freezer - 410ms
1580ms -- Pork#15 has been sent to the Freezer - 440ms
1590ms -- Pork#8: leaves CUTTER#1 (complete 1st stage)
1590ms -- Pork#9: enters the CUTTER#1
1590ms -- CUTTER#1: cutting... cutting... Pork#9 -- 130ms
1600ms -- Pork#8: waiting in the slot (cutted)
1620ms -- Pork#20 has been sent to the Freezer - 390ms
1720ms -- Pork#11 has been sent to the Freezer - 360ms
1720ms -- Pork#12 has been sent to the Freezer - 480ms
1720ms -- Pork#9: leaves CUTTER#1 (complete 1st stage)
1720ms -- Pork#13: enters the CUTTER#1
1720ms -- CUTTER#1: cutting... cutting... Pork#13 -- 250ms
1720ms -- Pork#3: leaves PACKER#2 (Complete)
1720ms -- Pork#5: enters to the factory (PACKER#2)
1720ms -- PACKER#2: processing & Packing the Pork#5 -- 690ms
1730ms -- Pork#9: waiting in the slot (cutted)
1790ms -- Pork#16: waiting in the slot
1790ms -- Pork#18 has been sent to the Freezer - 430ms
1830ms -- Pork#19 has been sent to the Freezer - 300ms
1870ms -- Pork#17 has been sent to the Freezer - 360ms
1970ms -- Pork#14 has been sent to the Freezer - 450ms
1970ms -- Pork#13: leaves CUTTER#1 (complete 1st stage)
1970ms -- Pork#10: enters the CUTTER#1
1970ms -- CUTTER#1: cutting... cutting... Pork#10 -- 290ms
1980ms -- Pork#13: waiting in the slot (cutted)
2010ms -- Pork#20 has been sent to the Freezer - 500ms
2020ms -- Pork#15 has been sent to the Freezer - 330ms
2050ms -- Pork#4: leaves PACKER#1 (Complete)
```

時間證明 (BONUS 2)

- **Pork#15首次進入freezer (1280ms)**
- **Pork#15再次進入freezer (1580ms)**
- **Pork#15再次進入freezer (2020ms)**

```
2050ms -- Pork#6: enters to the factory (PACKER#1)
2050ms -- PACKER#1: processing & Packing the Pork#6 -- 520ms
2080ms -- Pork#11: waiting in the slot
2130ms -- Pork#19 has been sent to the Freezer - 320ms
2200ms -- Pork#12 has been sent to the Freezer - 420ms
2220ms -- Pork#18 has been sent to the Freezer - 340ms
2230ms -- Pork#17 has been sent to the Freezer - 420ms
2260ms -- Pork#10: leaves CUTTER#1 (complete 1st stage)
2260ms -- Pork#16: enters the CUTTER#1
2260ms -- CUTTER#1: cutting... cutting... Pork#16 -- 270ms
2270ms -- Pork#10: waiting in the slot (cutted)
2350ms -- Pork#15 has been sent to the Freezer - 470ms
2410ms -- Pork#5: leaves PACKER#2 (Complete)
2410ms -- Pork#7: enters to the factory (PACKER#2)
2410ms -- PACKER#2: processing & Packing the Pork#7 -- 720ms
2420ms -- Pork#14: waiting in the slot
2450ms -- Pork#19 has been sent to the Freezer - 450ms
2510ms -- Pork#20 has been sent to the Freezer - 380ms
2530ms -- Pork#16: leaves CUTTER#1 (complete 1st stage)
2530ms -- Pork#11: enters the CUTTER#1
2530ms -- CUTTER#1: cutting... cutting... Pork#11 -- 150ms
2540ms -- Pork#16: waiting in the slot (cutted)
2560ms -- Pork#18 has been sent to the Freezer - 390ms
2570ms -- Pork#6: leaves PACKER#1 (Complete)
2570ms -- Pork#8: enters to the factory (PACKER#1)
2570ms -- PACKER#1: processing & Packing the Pork#8 -- 1000ms
2620ms -- Pork#12: waiting in the slot
2650ms -- Pork#17 has been sent to the Freezer - 420ms
2680ms -- Pork#11: leaves CUTTER#1 (complete 1st stage)
2680ms -- Pork#14: enters the CUTTER#1
2680ms -- CUTTER#1: cutting... cutting... Pork#14 -- 210ms
2690ms -- Pork#11: waiting in the slot (cutted)
2820ms -- Pork#15: exits freezer and waits beside the slot
2890ms -- Pork#20 has been sent to the Freezer - 350ms
2890ms -- Pork#14: leaves CUTTER#1 (complete 1st stage)
2890ms -- Pork#12: enters the CUTTER#1
2890ms -- CUTTER#1: cutting... cutting... Pork#12 -- 100ms
2900ms -- Pork#19 has been sent to the Freezer - 350ms
2900ms -- Pork#14: waiting in the slot (cutted)
2950ms -- Pork#18: exits freezer and waits beside the slot
2990ms -- Pork#12: leaves CUTTER#1 (complete 1st stage)
3070ms -- Pork#17: exits freezer and waits beside the slot
3130ms -- Pork#7: leaves PACKER#2 (Complete)
3130ms -- Pork#9: enters to the factory (PACKER#2)
3130ms -- PACKER#2: processing & Packing the Pork#9 -- 890ms
3140ms -- Pork#12: waiting in the slot (cutted)
3140ms -- CUTTER#1: under maintenance.
3140ms -- Pork#15: waiting in the slot(BONUSII)
```

時間證明 (BONUS 2)

- **Pork#15再次進入freezer (2350ms)**
- **Pork#15超過時間上限 (>1280 + 1500ms)**
  所以在備料台旁排隊

程式說明

Meat的結構

```
struct Pork
{
	int Num = 0; //number of meat
	int next_Time;
};
```

semaphore與mutex

semaphore與mutex

```
int M; //Meat數量
Int N; //每個切割工廠所配置的slot數量
Int C; //切割工廠數量
Int P; //粽子工廠數量
Int F; //開啟、關閉BONUS 2
Int T; //開啟、關閉BONUS 3
Int Error; //存取錯誤
Int Total_Cut = 0; // 總計Cut數量
Int Total_Pack = 0; // 總計Pack數量
Int finished_Factory; // 離線的工廠數量
Int total_Factory; // 總計的工廠數量
Int finished_Meat; // 完成的meat數量(跑完全部流程)
int total_Meat; // 全部的meat數量
int Time = 0; //目前時間
bool *Cut_Working; //各個Cutter的執行狀態(處理中or維護、檢討)
bool *Pack_Working; //各個Packer的執行狀態(處理中or維護、檢討)
```

semaphore與mutex

因為我一直把slot打成spot所以乾脆將錯就錯就讓它是spot吧！

```
sem_t Spot; //計算slot的剩餘空間(每次New_Slot、 Cutted_Slot多一個，Spot就會減少)
sem_t New; //保護New_Slot queue
sem_t Cutted; //保護Cutted_Slot queue
sem_t *Is_Cut; //批准Cutter執行(10ms一次)
sem_t *Is_Pack; //批准Packer執行(10ms一次)
sem_t *Try_in; //批准Meat thread執行(10ms一次)

pthread_mutex_t Is_New_queue;  //是否能操作 New_queue
Pthread_mutex_t Is_Cutted_queue; //是否能操作 Cutted_queue
pthread_mutex_t Is_finished_Factory; //是否能操作 finished_Factory
pthread_mutex_t Is_total_Factory; //是否能操作 total_Factory
pthread_mutex_t Is_finished_Meat; //是否能操作 finished_Meat
pthread_mutex_t Is_total_Meat; //是否能操作 total_Meat
pthread_mutex_t Is_waiting_queue; //是否能操作 waiting_queue
pthread_mutex_t Is_cout; //是否能執行 cout

queue <Pork> New_Slot; //Slot內未完成cut的meat
queue <Pork> Cutted_Slot; //Slot已完成cut的meat
queue <Pork> waiting; //BONUS 2 的儲存空間(備料台旁邊排隊)
```

Main function

處理例外情形

```
298 int main(int argc, char* argv[])
299 {
300         int millisecond;
301         if(argc != 7){
302                 cout << "input number is not right (it need 7 parameters)" << endl;
303                 return 0;
304         }
305
306         for(int i = 0; i < strlen(argv[1]); i++)
307                 if(argv[1][i] > '9' || argv[1][i] < '0') {
308                         cout << "Meat number is not right" << endl;
309                         return 0;
310                 }
311         for(int i = 0; i < strlen(argv[2]); i++)
312                 if(argv[2][i] > '9' || argv[2][i] < '0') {
313                         cout << "Slot number is not right" << endl;
314                         return 0;
315                 }
316         for(int i = 0; i < strlen(argv[3]); i++)
317                 if(argv[3][i] > '9' || argv[3][i] < '0') {
318                         cout << "CUTTER number is not right" << endl;
319                         return 0;
320                 }
321         for(int i = 0; i < strlen(argv[4]); i++)
322                 if(argv[4][i] > '9' || argv[4][i] < '0') {
323                         cout << "PACKER number is not right" << endl;
324                         return 0;
325                 }
326         if(strlen(argv[5]) == 1) {
327                 if(argv[5][0] != '1' && argv[5][0] != '0') {
328                         cout << "BONUSII number is not right (only 1 or 0)" << endl;
329                         return 0;
330                 }
331         }
332         else {
333                 cout << "BONUSII number is not right (only 1 or 0)" << endl;
334                 return 0;
335         }
336         if(strlen(argv[6]) == 1) {
337                 if(argv[6][0] != '1' && argv[6][0] != '0') {
338                         cout << "BONUSIII number is not right (only 1 or 0)" << endl;
339                         return 0;
340                 }
341         }
342         else {
343                 cout << "BONUSIII number is not right (only 1 or 0)" << endl;
344                 return 0;
```

```
345         }
346         M = atoi(argv[1]);
347         N = atoi(argv[2]);
348         C = atoi(argv[3]);
349         P = atoi(argv[4]);
350         F = atoi(argv[5]);
351         T = atoi(argv[6]);
352         if(M == 0) {
353                 cout << "There is not any meat(must >= 1)" << endl;
354                 return 0;
355         }
356         if(N == 0) {
357                 cout << "There is not any slot(must >= 1)" << endl;
358                 return 0;
359         }
360         if(C == 0) {
361                 cout << "There is not any CUTTER(must >= 1)" << endl;
362                 return 0;
363         }
364         if(P == 0) {
365                 cout << "There is not any PACKER(must >= 1)" << endl;
366                 return 0;
367         }
```

```
397         srand(N);
398         finished_Factory = C + P;
399         total_Factory  = C + P;
400         finished_Meat = M;
401         total_Meat  = M;
402
403         pthread_t Meat[M];
404         pthread_t Cutter[C];
405         pthread_t Packer[P];
406         pthread_t Clk;
407         Pork In[M];
408         int C_Num[C];
409         int P_Num[P];
410         Cut_Working = new bool[C];
411         for(int i = 0; i < C; i++)
412                 Cut_Working[i] = true;
413         Pack_Working = new bool[P];
414         for(int i = 0; i < P; i++)
415                 Pack_Working[i] = true;
416         sem_init(&Spot,0,(N + 1) * C);          //N + 1 the meat in process
417         sem_init(&Cutted,0,0);
418         sem_init(&New,0,0);
419         Is_Cut = new sem_t [C];
420         for(int i = 0; i < C; i++)
421                 sem_init(&(Is_Cut[i]),0,0);
422
423         Is_Pack = new sem_t [P];
424         for(int i = 0; i < P; i++)
425                 sem_init(&(Is_Pack[i]),0,0);
426
427         Try_in = new sem_t [M];
428         for(int i = 0; i < M; i++)
429                 sem_init(&(Try_in[i]),0,0);
430         pthread_mutex_init(&Is_New_queue, 0);
431         pthread_mutex_init(&Is_Cutted_queue, 0);
432         pthread_mutex_init(&Is_finished_Factory, 0);
433         pthread_mutex_init(&Is_total_Factory, 0);
434         pthread_mutex_init(&Is_finished_Meat, 0);
435         pthread_mutex_init(&Is_total_Meat, 0);
436         pthread_mutex_init(&Is_waiting_queue, 0);
437         pthread_mutex_init(&Is_cout, 0);
```

Pthread的宣告
&&
semaphore與mutex的初始化

```
407         //initialize the time
408         In[0].Num = 1;
409         In[0].next_Time = ((rand() % 6) + 5) * 10;
410         for(int i = 1; i < M; i++ ) {
411                 millisecond = ((rand() % 6) + 5) * 10;
412                 In[i].Num = i + 1;
413                 In[i].next_Time = In[i - 1].next_Time + millisecond;
414
415         }
416         for(int i = 0; i < C; i++ )
417                 C_Num[i] = i + 1;
418         for(int i = 0; i < P; i++ )
419                 P_Num[i] = i + 1;
420         for(int i = 0; i < M; i++ ) {
421                 Error = pthread_create(&(Meat[i]), NULL,Add_to_Slot,&(In[i]));
422                 if(Error != 0) {
423                         cout << "Couldn't Create Meat Pthread" << endl;
424                         return 0;
425                 }
426                 //cout << In[i].Num << " " << In[i].next_Time << endl;
427         }
428         Error = pthread_create(&(Clk), NULL, CLK, NULL);
429         if(Error != 0) {
430                 cout << "Couldn't Create Clk Pthread" << endl;
431                 return 0;
432         }
433         for(int i = 0; i < C; i++ ) {
434                 Error = pthread_create(&(Cutter[i]), NULL, Cut, &(C_Num[i]));
435                 if(Error != 0) {
436                         cout << "Couldn't Create Cutter Pthread" << endl;
437                         return 0;
438                 }
439         }
440         for(int i = 0; i < P; i++ ) {
441                 Error = pthread_create(&(Packer[i]), NULL, Pack, &(P_Num[i]));
442                 if(Error != 0) {
443                         cout << "Couldn't Create Packer Pthread" << endl;
444                         return 0;
445                 }
446         }
447         for(size_t i = 0; i < M; i++ )
448                 pthread_join(Meat[i], NULL);
449         for(size_t i = 0; i < C; i++ )
450                 pthread_join(Cutter[i], NULL);
451         for(size_t i = 0; i < P; i++ )
452                 pthread_join(Packer[i], NULL);
453         pthread_join(Clk, NULL);
```

初始化、建立各個meat

建立 Clker(統一計算秒數)

初始化、建立各個Cutter

初始化、建立各個Packer

等待各個thread完成

```
455                 /*Debug*/
456         int size = New_Slot.size();
457         if(size > 0) {
458             for (int i = 0; i < size; i++) {
459                 cout << New_Slot.front().Num << " ";
460                 New_Slot.pop();
461             }
462             cout << "\n";
463         }
464         size = Cutted_Slot.size();
465         if(size > 0) {
466             for (int i = 0; i < size; i++) {
467                 cout << Cutted_Slot.front().Num << " ";
468                 Cutted_Slot.pop();
469             }
470             cout << "\n";
471         }
472         size = waiting.size();
473         if(size > 0) {
474             for (int i = 0; i < size; i++) {
475                 cout << waiting.front().Num << " ";
476                 waiting.pop();
477             }
478             cout << "\n";
479         }
480         return 0;
481 }
```

Debug用的
(正常運作完的話
並不會輸出任何東西)

Clk function

```
39 void *CLK(void *arg) {
40         while(total_Factory) {
41                 while(finished_Factory < total_Factory);
42                 while(finished_Meat < total_Meat);
43                 usleep(10000); //us * 1000
44                 Time += 10;
45                 finished_Factory = 0;
46                 finished_Meat = 0;
47                 for(int i = 0; i < M; i++)
48                         sem_post(&(Try_in[i]));
49                 for(int i = 0; i < C; i++)
50                         sem_post(&(Is_Cut[i]));
51                 for(int i = 0; i < P; i++)
52                         sem_post(&(Is_Pack[i]));
53         }
54         pthread_exit(NULL);
55 }
```

- 主要是使用一個thread來數秒，當運作中的工廠總數工廠 >0 就會不斷地數秒
- 等待其他thread回報完成，
- 每次數秒為10ms並將上個cycle已完成的thread記錄歸零(等同於重新等待thread回報完成)
- 允許各個thread開始運行其程式碼

Over_Time function

```
58 bool Over_Time(Pork Target) {
59         if((Time - Target.next_Time) > 3000) { // if the meat in slot is over 3000ms, it will spoil
60                 pthread_mutex_lock(&Is_cout); // protect cout
61                 cout << Time << "ms -- Pork#" << Target.Num << " in the slots is spoilt(over 3000ms)" << endl;
62                 pthread_mutex_unlock(&Is_cout); // protect cout
63                 return true;
64         }
65         else
66                 return false;
67 }
```

- 主要是用於檢查這塊肉是不是在freezer待超過3000ms
  是的話回傳true；反之回傳false

Add_to_Slot function

```
276 void *Add_to_Slot(void *arg)
277 {
278         int  millisecond, First_in = 0;
279         bool Finished = false;
280         Pork Meat;
281         Meat.Num = ((Pork *) arg)->Num;
282         Meat.next_Time = ((Pork *) arg)->next_Time;
283         while(!Finished) {
284                 sem_wait(&(Try_in[Meat.Num - 1]));
285                 if(Time >= Meat.next_Time) {
286                         if(sem_trywait(&Spot) == 0) {
287                                 pthread_mutex_lock(&Is_New_queue); // protect New_Slot
288                                 New_Slot.push(Meat);// uncutted meat
289                                 sem_post(&New);
290                                 pthread_mutex_unlock(&Is_New_queue);
291                                 pthread_mutex_lock(&Is_cout); // protect cout
292                                 cout << Time << "ms -- Pork#" << Meat.Num << ": waiting in the slot" << endl;
293                                 pthread_mutex_unlock(&Is_cout); // protect cout
294                                 Finished = true;
295                         }
296                         else if((First_in != 0 && (Time - First_in) > 1490) && F == 1) { // BONUS II
297                                 pthread_mutex_lock(&Is_waiting_queue);  // protect New_Slot
298                                 Meat.next_Time = Time; // update the enter time
299                                 waiting.push(Meat);
300                                 pthread_mutex_unlock(&Is_waiting_queue); // protect New_Slot
301                                 pthread_mutex_lock(&Is_cout); // protect cout
302                                 cout << Time << "ms -- Pork#" << Meat.Num << ": exits freezer and waits beside the slot" << endl;
303                                 pthread_mutex_unlock(&Is_cout); // protect cout
304                                 Finished = true;
305                         }
306                         else {
307                                 if(First_in == 0)
308                                         First_in = Meat.next_Time;
309                                 millisecond = ((rand() % 21) + 30) * 10;
310                                 Meat.next_Time = Time + millisecond;
311                                 pthread_mutex_lock(&Is_cout); // protect cout
312                                 cout << Time << "ms -- Pork#" << Meat.Num << " has been sent to the Freezer - " << millisecond << "ms"<< endl;
313                                 pthread_mutex_unlock(&Is_cout); // protect cout
314                         }
315                 }
316                 pthread_mutex_lock(&Is_finished_Meat); // protect finished_Meat
317                 finished_Meat ++;
318                 pthread_mutex_unlock(&Is_finished_Meat); // protect finished_Meat
319
320         }
321         pthread_mutex_lock(&Is_total_Meat); // protect total_Meat
322         total_Meat --;
323         pthread_mutex_unlock(&Is_total_Meat); // protect total_Meat
324         pthread_exit(NULL);
325 }
```

- 儲存傳入的meat編號與觸發時間
- 每個新的cycle，都能被解鎖
- Spot有空位所以放入，並印出訊息，Finished = true等同於該thread結束
- **BONUS 2：查看該個thread離首次觸發時間是否>=1500，如果有代表就移入waiting_queue，優先進入slot (由F來控制是否套用)**
- 產生新的亂數時間(300~500ms)，進入freezer等待
- 回報這個thread在本次cycle的作業已完成
- 這個meat已成功進入queue total_Meat -- 代表不用在等待該thread

Cut function

```
158 void *Cut(void *arg)
159 {
160         int  millisecond = 0, last_Time = 0, Cut_Num = *((int *) arg);
161         Pork Meat, last_Meat, check, BONUSII;
162         while(Total_Cut < M) {
163                 sem_wait(&Is_Cut[Cut_Num - 1]);
164                 if(last_Meat.Num != 0) { //put it to slot latter
165                         if(Cutted_Slot.size() + New_Slot.size() < N * C) {
166                                 pthread_mutex_lock(&Is_Cutted_queue); // protect Cutted_Slot
167                                 Cutted_Slot.push(last_Meat); //Put in the Cutted Slot
168                                 sem_post(&Cutted);
169                                 pthread_mutex_unlock(&Is_Cutted_queue);
170                                 pthread_mutex_lock(&Is_cout); // protect cout
171                                 cout << Time << "ms -- Pork#" << last_Meat.Num << ": waiting in the slot (cutted)" << endl;
172                                 pthread_mutex_unlock(&Is_cout); // protect cout
173                                 last_Meat.Num = 0;
174                                 Total_Cut++; //finish one
175                         }
176                 }
177                 if(Meat.Num != 0 && Time >= Meat.next_Time) {
178                         pthread_mutex_lock(&Is_cout); // protect cout
179                         cout << Time << "ms -- Pork#" << Meat.Num << ": leaves CUTTER#"<< Cut_Num <<" (complete 1st stage)" << endl;
180                         pthread_mutex_unlock(&Is_cout); // protect cout
181                         last_Meat = Meat;
182                         Meat.Num = 0;
183                 }
184                 if(Meat.Num == 0) {
185                         if(sem_trywait(&New) == 0) {
186                                 if(!Cut_Working[Cut_Num - 1]) {//wake up the cutter
187                                         sem_post(&New); // not count
188                                         Cut_Working[Cut_Num - 1] = true;
189                                 }
190                                 else {
191                                         pthread_mutex_lock(&Is_New_queue); // protect New_Slot
192                                         Meat = New_Slot.front();       //take from slots
193                                         New_Slot.pop();
194                                         pthread_mutex_unlock(&Is_New_queue); // protect New_Slot
195                                         pthread_mutex_lock(&Is_cout); // protect cout
196                                         cout << Time << "ms -- Pork#" << Meat.Num << ": enters the CUTTER#"<< Cut_Num << endl;
197                                         pthread_mutex_unlock(&Is_cout); // protect cout
198                                         millisecond = ((rand() % 21) + 10) * 10;
199                                         Meat.next_Time = Time + millisecond;//update time
200                                         pthread_mutex_lock(&Is_cout); // protect cout
201                                         cout << Time << "ms -- CUTTER#"<< Cut_Num <<": cutting... cutting... Pork#" << Meat.Num << " -- " << millisecond << "ms"<< endl;
202                                         pthread_mutex_unlock(&Is_cout); // protect cout
203                                 }
204                         }
```

- 每個新的cycle，才能被解鎖並執行以下程式
- 將剛切好的肉嘗試放回slot (Cutted_queue)，有空位所以放入，印出訊息也將Total_Cut++代表已完成的肉+1
- 目前的時間>=目前切割中的肉的觸發時間，代表該肉已完成，離開cutter ，並印出訊息
- Meat.Num == 0代表目前沒有處理中的肉，查看Cutter是否在運作狀態，是就喚醒(10ms後才會開始工作)
- Meat.Num == 0代表目前沒有處理中的肉，所以嘗試從slot中拿出未完成的肉(從New_queue拿出)

```
205         }
206         if(Meat.Num == 0 && last_Meat.Num == 0) { //no works
207                 if(Time >= last_Time) {
208                         for(int i = 0; i < P; i++) {
209                                 if(!Pack_Working[i]) {
210                                         pthread_mutex_lock(&Is_cout); // protect cout
211                                         cout << Time <<"ms - CUTTER#"<< Cut_Num <<": under reviewing together..." << endl;
212                                         pthread_mutex_unlock(&Is_cout); // protect cout
213                                         break;
214                                 }
215                                 if(i == P - 1) {
216                                         pthread_mutex_lock(&Is_cout); // protect cout
217                                         cout << Time <<"ms -- CUTTER#"<< Cut_Num <<": under maintenance." << endl;
218                                         pthread_mutex_unlock(&Is_cout); // protect cout
219                                 }
220                         }
221                         Cut_Working[Cut_Num - 1] = false;
222                         last_Time = Time + ((rand() % 10) + 1) * 10;
223                 }
224                 int Rest = Total_Cut;
225                 for(int i = 0; i < P; i++)
226                         if(Cut_Working[i] && i != Cut_Num - 1) // find the rest one
227                                 Rest ++;
228                 if(Rest == M)
229                         break;
230         }
231         if(waiting.size() > 0) {
232                 if(sem_trywait(&Spot) == 0) {
233                         pthread_mutex_lock(&Is_waiting_queue);  // protect waiting_queue
234                         BONUSII = waiting.front();
235                         waiting.pop();
236                         pthread_mutex_unlock(&Is_waiting_queue);  // protect waiting_queue
237
238                         pthread_mutex_lock(&Is_New_queue); // protect New_Slot
239                         New_Slot.push(BONUSII);// uncutted meat
240                         sem_post(&New);
241                         pthread_mutex_unlock(&Is_New_queue);
242                         pthread_mutex_lock(&Is_cout); // protect cout
243                         cout << Time << "ms -- Pork#" << BONUSII.Num << ": waiting in the slot(BONUSII)" << endl;
244                         pthread_mutex_unlock(&Is_cout); // protect cout
245                 }
246         }
```

- Meat_Num == 0
  && last_Meat_Num == 0
  代表目前Cutter沒有事情做，所以進入維護或是檢討模式，同時也決定下次的檢查時間。

- 確定自己是否能夠離線，計算其他的Cutter正在切割的豬肉與所有完成的數量加總，如果已達到豬肉總數就離線；反之繼續維護或是檢討模式。

- **BONUS 2：如果slot有空位會優先將waiting_queue內的豬肉放入slot中(New_queue)並印出訊息**

```
247             if(T) {
248                     pthread_mutex_lock(&Is_New_queue); // protect New_Slot
249                     int size = New_Slot.size();
250                     if(size > 0) {
251                         for (int i = 0; i < size; i++) {
252                             check = New_Slot.front();
253                             New_Slot.pop();
254                             if(!Over_Time(check))
255                                     New_Slot.push(check);
256                             else {
257                                     sem_wait(&New);
258                                     sem_post(&Spot);
259                                     Total_Cut++;
260                             }
261                         }
262                     }
263                     pthread_mutex_unlock(&Is_New_queue);
264             }
265             pthread_mutex_lock(&Is_finished_Factory); // protect finished_Factory
266             finished_Factory ++;
267             pthread_mutex_unlock(&Is_finished_Factory); // protect finished_Factory
268         }
269         pthread_mutex_lock(&Is_total_Factory); // protect total_Factory
270         total_Factory --;
271         pthread_mutex_unlock(&Is_total_Factory); // protect total_Factory
272         Cut_Working[Cut_Num - 1] = false;
273         pthread_exit(NULL);
274 }
```

- **BONUS 3：每個cycle即將結束時，都會察看每個在slot(New_Slot)內的豬肉有沒有over time如果有就丟棄並釋出Slot空間，沒有就放回去。**
- 回報這個thread在本次cycle的作業已完成
- total_Factory --
  代表不用在等待該thread
  這個cutter永久地停止工作

Pack function

```
69 void *Pack(void *arg)
70 {
71         int  millisecond = 0, last_Time = 0, Pack_Num = *((int *) arg);
72         Pork Meat, check;
73         while(Total_Pack < M) {
74                 sem_wait(&Is_Pack[Pack_Num - 1]);
75                 if(Meat.Num != 0 && Time >= Meat.next_Time) {
76                         pthread_mutex_lock(&Is_cout); // protect cout
77                         cout << Time << "ms -- Pork#" << Meat.Num << ": leaves PACKER#"<< Pack_Num <<" (Complete)" << endl;
78                         pthread_mutex_unlock(&Is_cout); // protect cout
79                         Total_Pack++;
80                         Meat.Num = 0;
81                 }
82                 if(Meat.Num == 0) {
83                         if(sem_trywait(&Cutted) == 0) {
84                                 if(!Pack_Working[Pack_Num - 1]) {//wake up the cutter
85                                         sem_post(&Cutted); // not count
86                                         Pack_Working[Pack_Num - 1] = true;
87                                 }
88                                 else {
89                                         pthread_mutex_lock(&Is_Cutted_queue); // protect Cutted_Slot
90                                         Meat = Cutted_Slot.front();    //take from slots
91                                         Cutted_Slot.pop();
92                                         sem_post(&Spot);
93                                         pthread_mutex_unlock(&Is_Cutted_queue);
94                                         pthread_mutex_lock(&Is_cout); // protect cout
95                                         cout << Time<< "ms -- Pork#" << Meat.Num << ": enters to the factory (PACKER#" << Pack_Num << ")" << endl;
96                                         pthread_mutex_unlock(&Is_cout); // protect cout
97                                         millisecond = ((rand() % 51) + 50) * 10;
98                                         Meat.next_Time = Time + millisecond; //update time
99
100                                        cout << Time << "ms -- PACKER#"<< Pack_Num <<": processing & Packing the Pork#" << Meat.Num << " -- " << millisecond << "ms"<< endl;
101                                }
102                        }
103                }
```

- 每個新的cycle，才能被解鎖並執行以下程式
- 讓包好的肉離開Packer，印出訊息也將Total_Pack++代表已完成的肉+1
- Meat.Num == 0代表目前沒有處理中的肉，查看Packer是否在運作狀態，是就喚醒(10ms後才會開始工作)
- Meat.Num == 0代表目前沒有處理中的肉，所以嘗試從slot中拿出未完成的肉(從Cutted_queue拿出)

```
104         if(Meat.Num == 0) { //no works
105                 if(Time >= last_Time) {
106                         for(int i = 0; i < C; i++) {
107                                 if(!Cut_Working[i]) {
108                                         pthread_mutex_lock(&Is_cout); // protect cout
109                                         cout << Time <<"ms - PACKER#"<< Pack_Num <<": under reviewing together..." << endl;
110                                         pthread_mutex_unlock(&Is_cout); // protect cout
111                                         break;
112                                 }
113                                 if(i == C - 1) {
114                                         pthread_mutex_lock(&Is_cout); // protect cout
115                                         cout << Time <<"ms -- PACKER#"<< Pack_Num <<": under maintenance." << endl;
116                                         pthread_mutex_unlock(&Is_cout); // protect cout
117                                 }
118                         }
119                         Pack_Working[Pack_Num - 1] = false;
120                         last_Time = Time + ((rand() % 10) + 1) * 10;
121                 }
122                 int Rest = Total_Pack;
123                 for(int i = 0; i < P; i++)
124                         if(Pack_Working[i] && i != Pack_Num - 1) // find the rest one
125                                 Rest ++;
126                 if(Rest == M)
127                         break;
128         }
129         if(T) {
130                 pthread_mutex_lock(&Is_Cutted_queue); // protect Cutted_Slot
131                 int size = Cutted_Slot.size();
132                 if(size > 0) {
133                     for (int i = 0; i < size; i++) {
134                         check = Cutted_Slot.front();
135                         Cutted_Slot.pop();
136                         if(!Over_Time(check))
137                                 Cutted_Slot.push(check);
138                         else {
139                                 sem_wait(&Cutted);
140                                 sem_post(&Spot);
141                                 Total_Pack++;
142                         }
143                     }
144                 }
145                 pthread_mutex_unlock(&Is_Cutted_queue);
146         }
```

- Meat_Num == 0而且沒有進入前一個if代表目前Packer沒有事情做，所以進入維護或是檢討模式，同時也決定下次的檢查時間。
- 確定自己是否能夠離線，計算其他的Packer正在包的豬肉與所有完成的數量加總，如果已達到豬肉總數就離線；反之繼續維護或是檢討模式。
- **BONUS 3：每個cycle即將結束時，都會察看每個在slot(Cutted_Slot)內的豬肉有沒有over time如果有就丟棄並釋出Slot空間，沒有就放回去。**

```
147             pthread_mutex_lock(&Is_finished_Factory); // protect finished_Factory
148             finished_Factory ++;
149             pthread_mutex_unlock(&Is_finished_Factory); // protect finished_Factory
150         }
151         pthread_mutex_lock(&Is_total_Factory); // protect finished_Factory
152         total_Factory --;
153         pthread_mutex_unlock(&Is_total_Factory); // protect finished_Factory
154         Pack_Working[Pack_Num - 1] = false;
155         pthread_exit(NULL);
156 }
```

- 回報這個thread在本次cycle的作業已完成
- total_Factory --
  代表不用在等待該thread
  這個cutter永久地停止工作

以上為我的HW3作業介紹
感謝助教與教授
願意花時間看完