OS 作業3 說明文件

1103309 王聖允

測試建置環境

- Distro：KDE Neon 5.27 based on Ubuntu 22.04
- Toolchain：g++ 11.3.0
- Shell：bash 5.1.16 with KDE Konsole

編譯指令

先cd到檔案存放路徑，並執行以下指令：

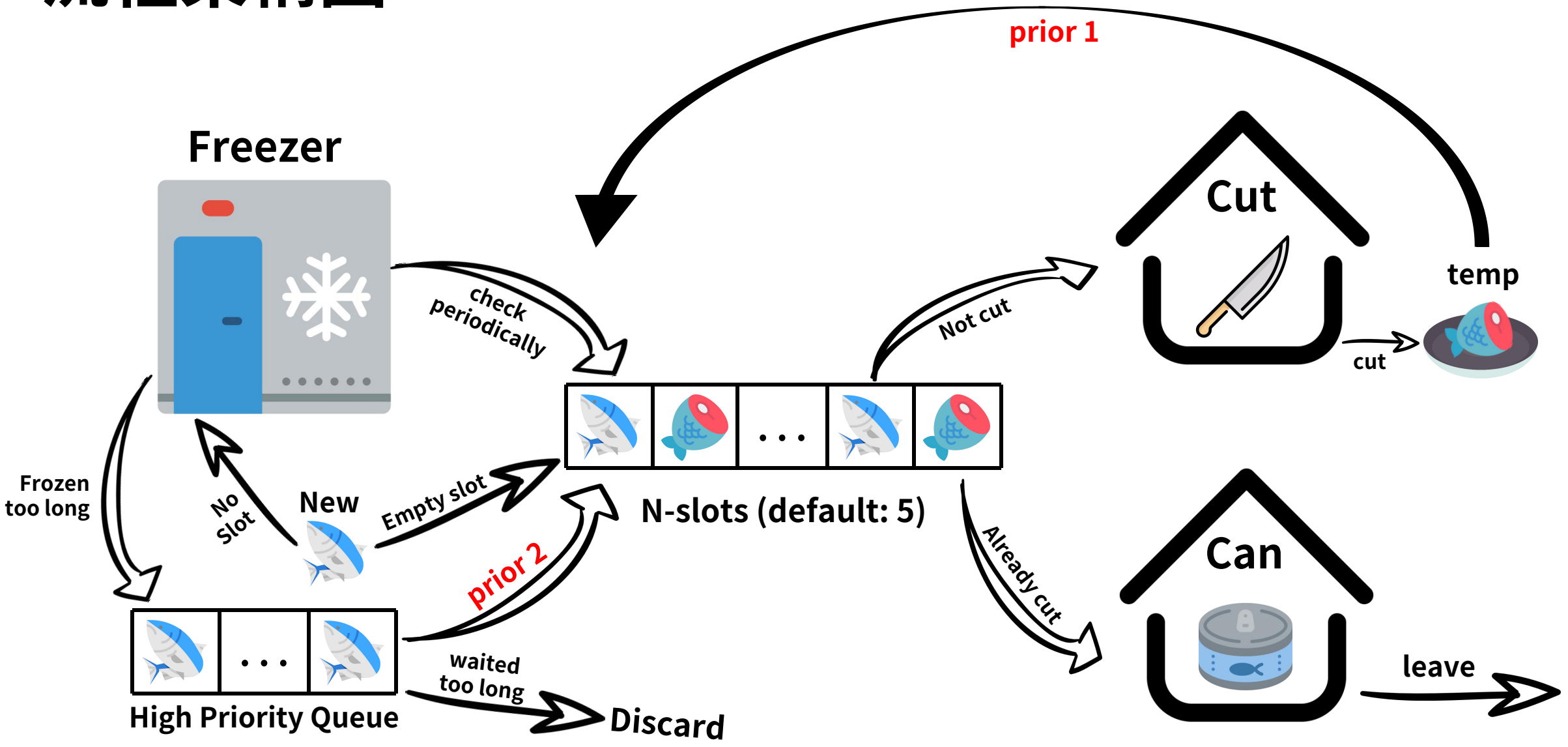
```
g++ ./main.cpp -lpthread -o main.out
```

執行指令

指令格式：`./main.out <arg1> <arg2>`

註：<arg1>：fish count (required)、<arg2>：slot count (optional)

流程架構圖


prior 1
Freezer
check periodically
Cut
temp
cut
Not cut
Frozen too long
No Slot
New
Empty slot
N-slots (default: 5)
prior 2
Already cut
Can
leave
waited too long
High Priority Queue
Discard

實做方法及原則

- 一個工廠一個Thread，並依照排隊順序處理。
- Timer為獨立的一個Thread，負責計時、產生新魚、並檢查工廠運作狀態
- 邏輯上建立2個Queue，分別給「待切」和「待裝罐」，讓程式更容易執行
  但是實際總位置還是N個，用Semaphore的概念實現（使用-1、釋放+1）
- 因為fish離開Freezer的順序及時間點不定，所以在Freezer中，每一條魚所佔的位置都自成一個Thread，個別判斷該fish是否應該離開Freezer
- 多執行緒中共享之資源皆用互斥鎖控管，避免同時取用修改資料造成錯誤
  例如：slot_count、各個queue、cout

實做方法及原則

- 固定10ms檢查一次工廠是否 under maintenance 或 reviewing，
  避免大量輸出狀態訊息
- 如果已經沒有待切的庫存，切割工廠先關閉，停止輸出維護訊息
- 過程中 Timer 協助檢查是否有 slot 超用的情況，防止設計瑕疵
- 裝罐工廠會協助檢查最後的魚隻數量是否與輸入符合，防止過程中 fish 丟失

Bonus

- [Bonus 1 - 多個工廠]：未實行

- [Bonus 2 - 等待上限]：設定如果冷凍超過600ms，直接設定高優先權，
  並且到高優先的 Queue 中排隊等待，有位子直接進入
  但如果又等待300ms還是沒有進入 slot 則丟棄。

- [Bonus 3 - 死結可能性]：經過多次測試，都沒有發生死結
  程式中互斥鎖都有設計在正確時機鎖定和解鎖
  而且遇到交疊鎖定時，都有按照一定的階層進行
  所以我認為應該不會造成 deadlock

基本輸出訊息解釋

工廠狀態訊息（Reviewing / Maintenance）

```
[  -----  ] 0ms: Cutter and Canner are under reviewing together
```

魚進入 slot 等待（橙色）（多種來源，參見流程圖及詳細訊息）

```
[ Waiting ] 6ms: Fish 1 (NEW) waiting in slot to enter cut factory
```

進入切割工廠（藍色），顯示所需時間

```
[ Cut_Ent ] 6ms: Fish 1 enters the cut factory, need 19ms
```

離開切割工廠（深綠）

```
[ Cut_Out ] 25ms: Fish 1 left the cutter.
```

進入罐頭工廠（藍色），顯示所需時間

```
[ Can_Ent ] 25ms: Fish 1 enters the canning factory, need 87ms
```

離開切割工廠（淺綠）

```
[ Can_Out ] 112ms: Fish 1 left the canner (Complete).
```

基本輸出訊息解釋

魚在冷凍庫等待，顯示多久後檢查是否有空位

```
[ Freezer ] 56ms: No empty slot, Fish 8 (NEW) Enters the freezer for 34ms
```

魚在冷凍庫太久了，設定高優先權

```
[  PRIOR  ] 667ms: Fish 9 left the freezer (frozen for too long). Set higher priority
```

高優先權者成功進入 slot

```
[ Waiting ] 769ms: Fish 9 enters the slot (high priority), waiting to be cut
```

高優先權者還是等不到 slot，超過時間限制遭丟棄

```
[ DISCARD ] 1001ms: Fish 12 Waited for too long. Discarded
```

確定沒有魚要切了，關閉切割工廠

```
[  CLOSE  ] 1242ms: Nothing more to be cut, shutting down the cutter...
```

確定所有魚都被正確處理，結束程式

```
[  CLOSE  ] 1700ms: All fish are handled properly, end of the job.
```

測試時可以更改之數值

此為計時器的相對時間單位，可以改變整體運作速度
1500為預設值，與實際ms相近，執行速度稍快，不便肉眼觀察工廠動向
調整為 150000 最適合觀看程式運行之流程並檢查

```
int real_ms = 1500; // control the speed of the program
```

此為隨機數之種子，更改後可以測試不同的秒數組合

```
#define SEED 0  // the seed for srand()
```

範例測資（已測試）

```
./main.out 15 5

./main.out 10 2

./main.out 20 5

./main.out 50 10

./main.out 80 20
```

測試輸出過於冗長，不附上測試截圖
但配合「流程圖」和「輸出訊息解釋」
即可理解運作過程及判別合理性
也可以從這些測資中測試 Bonus 功能

其他解說

- 程式碼中有更多詳細的步驟註解，就不附上冗長的程式截圖。以上講解若有任何不足之處，請見 main.cpp 檔案中的完整註解。