

作業系統期末專案報告

第 14 組

103703013 資科三 黃育萱

103703015 資科三 蔡雨芝

103702012 心理三 陳冠聞

目錄

- 一. 報告實作主題、內容簡介
- 二. Thread 程式碼與功能介紹
- 三. Multi-Thread 之間的同步與合作
- 四. 根據 Poisson Distribution 產生客人
- 五. GUI 呈現與附加功能
- 六. Github 與版本管理、分工
- 七. 組員心得
- 八. 參考資料

一、報告實作主題、內容簡介

主題：The Sleeping Barber (SB) Problem

語言：C ++ 11

關鍵：Multi-thread, synchronization, without race conditions.

內容：

The barber shop has m barbers with m barber chairs, and n chairs ($m < n$) for waiting customers, if any, to sit in. If there are no customers present, a barber sits down in a barber chair and falls asleep. When a customer arrives, he has to wake up a sleeping barber.

If additional customers arrive while all barbers are cutting customers' hair, they either sit down (if there are empty chairs) or leave the shop (if all chairs are full). The thread synchronization problem is to program the barbers and the customers without getting into race conditions.

二、Thread 程式碼與功能介紹

1. barberThread :

每個 barber 各自擁有獨立的 thread，用來控管 barber 的狀態（現在是 sleeping or cutting hair 的狀態）。

```
void *barberThread(void* arg)
{
    int *pID = (int*)arg;
    ioMutex.lock(); // Acquire access to waiting
    cout << "This is Barber No." << *pID << endl;
    ioMutex.unlock(); // Release waiting

    while(1)
    {
        barMutex.lock();
        cusMutex.lock();
        if(totalServedCustomers < realNum_customer){
            cusMutex.unlock();
            customers.wait(); // Try to acquire a customer.
            //Go to sleep if no customers
            Mutex.lock(); // Acquire access to waiting
            //When a barber is waken -> wants to modify # of available chairs
            barbers.signal(); // The barber is now ready to cut hair

            int nowCut = nextCut;
            nextCut = (nextCut+1) % NUM_CHAIRS;
            availableChairs++;

            Mutex.unlock(); // Release waiting

            /* GUI change barber's mode */
            isBusy[*pID-1] = true;
            cutting[*pID-1] = waitingChairs[nowCut].data->cusID;
            isSit[waitingChairs[nowCut].seqNumber] = false;
            glutPostRedisplay(); //GUI

            cutHair(*pID, waitingChairs[nowCut]); //pick the customer which counter point

            isBusy[*pID-1] = false;
            //this_thread::sleep_for( chrono::milliseconds( 1000 ));
            glutPostRedisplay(); //GUI
        }
        else{
            barMutex.unlock();
            cusMutex.unlock();
            break;
        }
    }
}
```

二、Thread 程式碼與功能介紹 (cont.)

2. customerThread :

每個 customer 各自擁有獨立的 thread，用來模擬客人的行為。當客人進入理髮廳：會判斷是否有空的座位，若有的話就坐下等待、沒有的話就離開。

```
void *customerThread(void* arg)
{
    struct customerData *data = (struct customerData*)arg;
    Mutex.lock(); // Acquire access to waiting
    if( availableChairs == 0 )
    {
        ioMutex.lock(); // Acquire access to waiting
        cout << "There is no available chair. Customer No." << data->cusID << " is leaving!" << endl;
        comeCus[data->cusID-1] = false;

        ioMutex.unlock(); // Release waiting

        cusMutex.lock();
        --realNum_customer;
        cusMutex.unlock();

        Mutex.unlock();
        pthread_exit(0);
    }
    ioMutex.lock(); // Acquire access to waiting
    cout << "Customer No." << data->cusID << " is sitting on chair " << nextSit << "." << endl;
    comeCus[data->cusID-1] = false;
    seat[nextSit] = data->cusID;

    ioMutex.unlock(); // Release waiting

    waitingChairs[nextSit].data = data;
    nextSit = (nextSit+1) % NUM_CHAIRS;
    availableChairs--;
    showWhoSitOnChair();

    //usleep(10000);
    glutPostRedisplay(); //GUI

    customers.signal(); // Wake up a barber (if needed)
    Mutex.unlock(); // Release waiting
    barbers.wait(); // Go to sleep if number of available barbers is 0
    waitForHairCut(data);

    ioMutex.lock(); // Acquire access to waiting
    cout << "(C)Customer No." << data->cusID << " just finished his haircut!"<<endl;
    ioMutex.unlock(); // Release waiting
}
```

三、Multi-Thread 之間的同步與合作

透過 Semaphore & Mutex 避免 Race condition :

Semaphore barbers(NUM_BARBERS);

現在 available 的 barber 數量(有 NUM_BARBERS 個 barber 是有空的)

Semaphore customer(0);

現在的客人數量(一開始有 0 個客人)

mutex Mutex;

barbers 要搶客人時，為了避免 race condition，設立的 Mutex。

mutex cusMutex;

當有發生客滿 (客人離開)，或是客人剪完頭髮的狀況，

為了避免有兩個事件同時修改 customer 的相關數據，而設立的 Mutex。

mutex barMutex;

避免有兩個 barbers 同時在等待最後一個客人，造成無限等待 (因為只剩一個客人需要被服務，但兩個 barbers 同時發現還有客人，便開始等待) 的狀況，而設立的 Mutex。

四、根據 Poisson Distribution 產生客人

用 Poisson Distribution 的方式，分配在該秒鐘內會產生幾個客人。

※註：Demo 時經過老師提醒才知道理解錯原題目的意思

```
int *poissonDistribution(float mean, int range, int num_period)
{
    const int NUM_TIMES = num_period;
    default_random_engine generator;
    poisson_distribution<int> distribution(mean);
    int *frequenceArray = new int[range];

    for(int i=0; i<range; i++)
        frequenceArray[i] = 0;

    for(int i=0; i<NUM_TIMES; )
    {
        int number = distribution(generator);
        if(number < range){
            frequenceArray[number]++;
            i++;
        }
    }

    realNum_customer = 0;
    for(int i=0; i<range; i++)
    {
        cusMutex.lock();

        ioMutex.lock();
        cout << i << " : " << frequenceArray[i] << endl;
        ioMutex.unlock();

        realNum_customer += frequenceArray[i];
        cusMutex.unlock();
    }
    ioMutex.lock();
    cout << "Sum : " << realNum_customer << endl << endl;
    ioMutex.unlock();

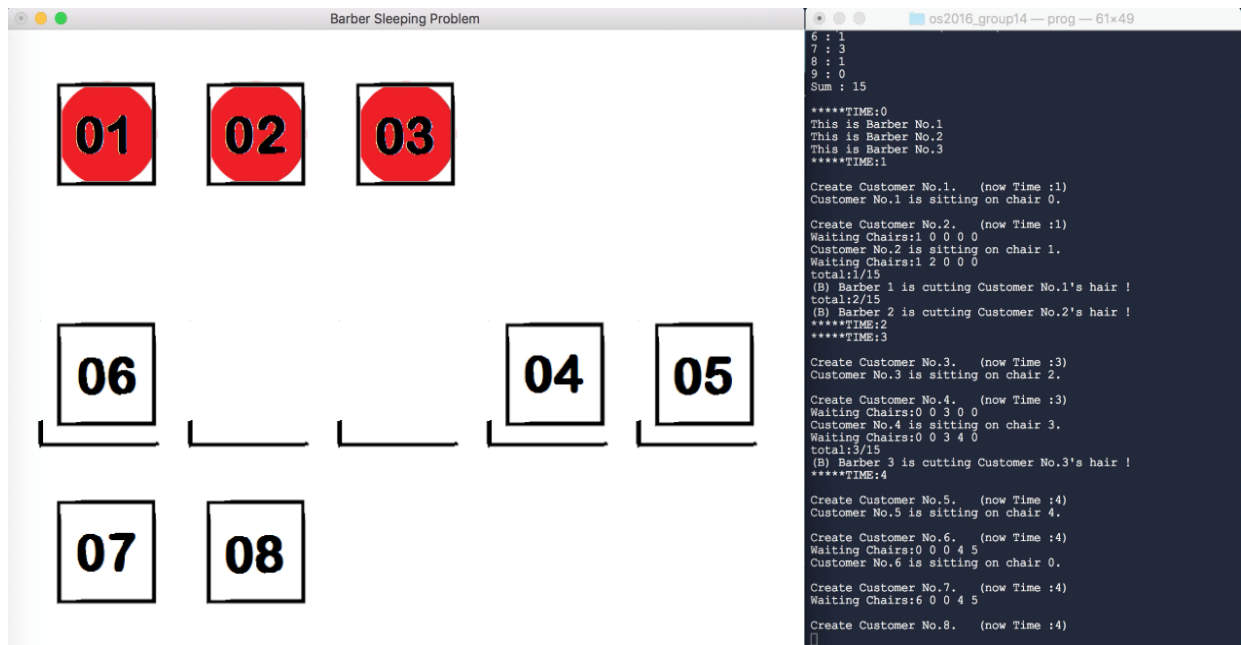
    return frequenceArray;
}
```

五、GUI 呈現與附加功能

初始/結束畫面 (附圖為結束)

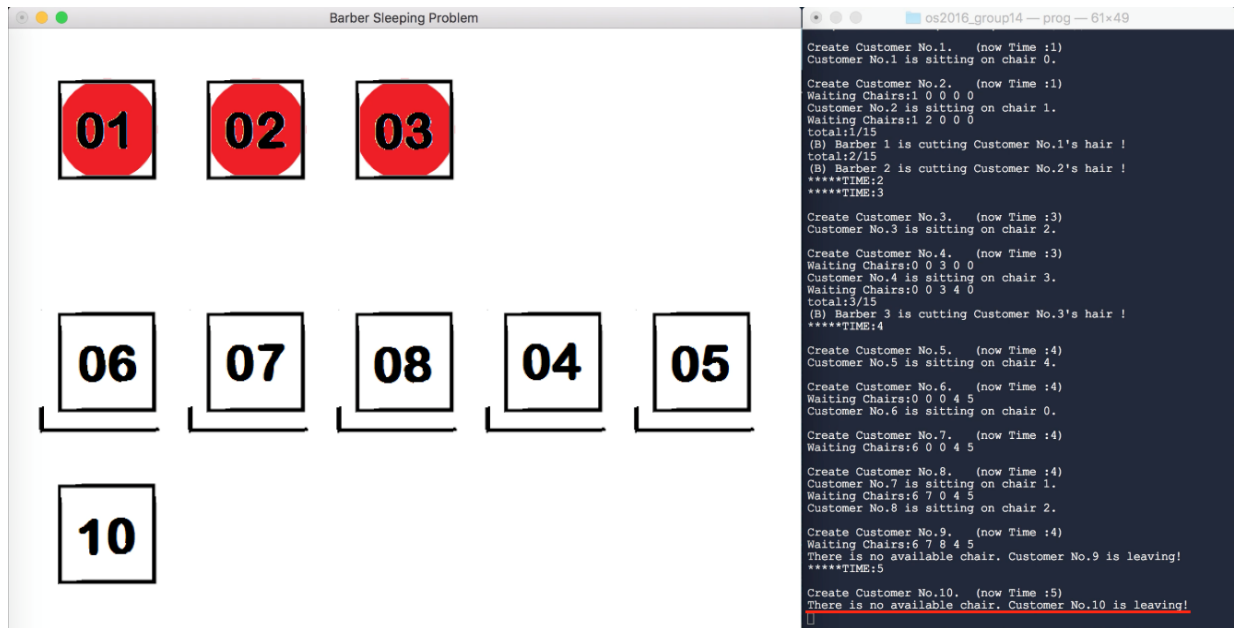


客人進入理髮店等待

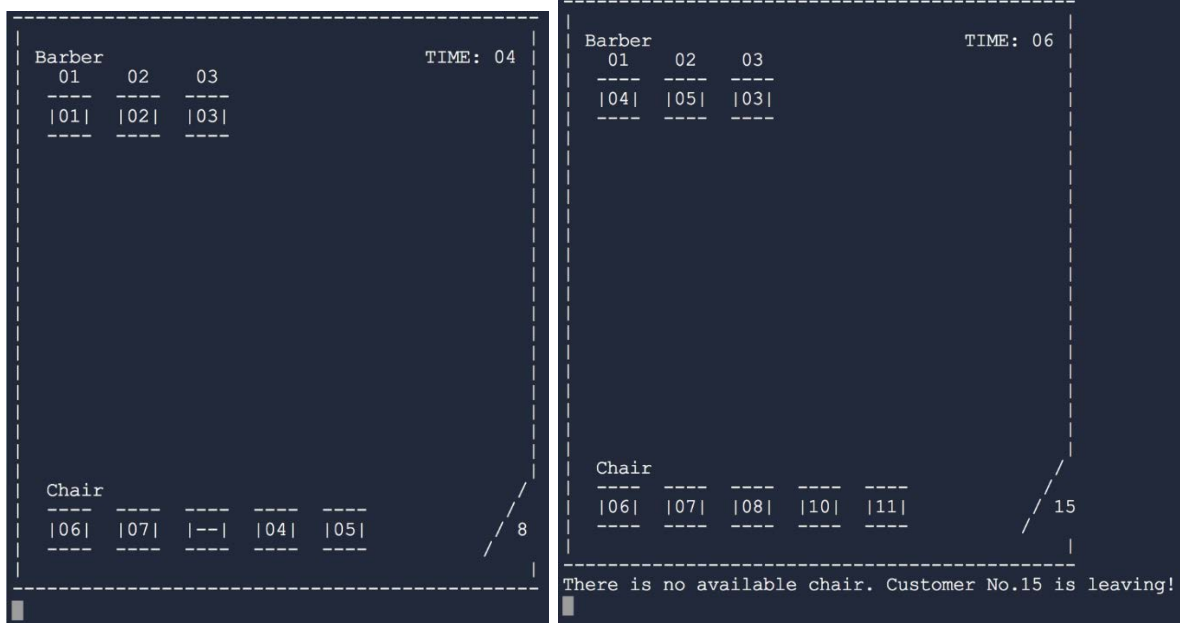


五、GUI 呈現與附加功能(cont.)

若客人進入理髮店時無位置則離開



附加功能：文字 UI 顯示



六、Github 與版本管理、分工

Version 1 : <pthread.h> (適用 Windows/ Ubuntu (Mac OS 有編譯問題))

[SB01~SB10.cpp]

虛擬碼撰寫：陳冠聞

c++ 程式碼撰寫、優化：陳冠聞、蔡雨芝

Version 2 : Windows -> Mac OS (解決 Mac OS 編譯問題)

[SB11.cpp] 程式碼轉換：黃育萱

Version 3 : 程式碼優化、轉換為適用於 Windows 版本

[SB12、SB13win.cpp] 程式碼優化：陳冠聞、蔡雨芝

Version 4 : GUI (using Qt)

[SB13.cpp] GUI 程式碼：黃育萱

Version 5 : <thread> & <mutex> (適用於 Windows/Mac OS/Ubuntu)

[SB15.cpp] 程式碼轉換：蔡雨芝






























































































Version 6 (Final) : GUI (using OpenGL)

[SB16.cpp] GUI 程式碼：黃育萱

報告書：

主要撰寫：蔡雨芝

Github 截圖、GUI 呈現與附加功能：黃育萱

Summary for Dec 14, 2016		
	sem_init -> sem_open Patchset submitted on Dec 10, 2016	 diff 
Comments on Dec 8, 2016		
	Update SB09.cpp Reviewed by sb on Dec 8, 2016	 diff 
	Update SB09.cpp Reviewed by sb on Dec 8, 2016	 diff 
	Comment out barbers' pthread_join	 diff 
	Update SB09.cpp Reviewed by sb on Dec 8, 2016	 diff 
	Create SB_end_condition Reviewed by sb on Dec 8, 2016	 diff 
Comments on Dec 5, 2016		
	Add SB09.cpp Reviewed by sb on Dec 5, 2016	 diff 
	Update SB07/08 - Reviewed by sb on Dec 5, 2016	 diff 
	Add SB06(from Branch chex)	 diff 
	PRINT TIME(when creating customers) in SB07.cpp	 diff 
	Update SB07.cpp Reviewed by sb on Dec 5, 2016	 diff 
	Add SB07.cpp - Reviewed by sb on Dec 5, 2016	 diff 
	Update SB06.cpp - Reviewed by sb on Dec 5, 2016	 diff 
	Update SB06 Reviewed by sb on Dec 5, 2016	 diff 
	Upload out1 - Reviewed by sb on Dec 4, 2016	 diff 
	Updated SB06.cpp - Reviewed by sb on Dec 5, 2016	 diff 
	Update SB06.cpp Reviewed by sb on Dec 5, 2016	 diff 
	add SB06.cpp - Reviewed by sb on Dec 5, 2016	 diff 
	20161205 comment out struct CustomerData - Reviewed by sb on Dec 5, 2016	 diff 
	change getFairCut()	 diff 
Comments on Dec 4, 2016		
	20161204 add SB05.cpp Binary blob submitted on Dec 4, 2016	 diff 
Comments on Dec 3, 2016		
	ver3.1 - Binary blob submitted on Dec 3, 2016	 diff 
	ver3.0 Reviewed by sb on Dec 3, 2016	 diff 
	Update PoissonDistribution.cpp Reviewed by sb on Dec 3, 2016	 diff 
	combine with createCustomer function Reviewed by sb on Dec 3, 2016	 diff 
	Update PoissonDistribution.cpp Reviewed by sb on Dec 3, 2016	 diff 
	Update PoissonDistribution.cpp Reviewed by sb on Dec 3, 2016	 diff 
Comments on Nov 25, 2016		
	ver2.0 Reviewed by sb on Nov 25, 2016	 diff 
Summary on Nov 25, 2016		
	Test for function PoissonDistribution Reviewed by sb on Nov 25, 2016	 diff 
Comments on Nov 16, 2016		
	20161128 ver.8.4 Reviewed by sb on Nov 16, 2016	 diff 
	20161128 ver.8.3 - Reviewed by sb on Nov 16, 2016	 diff 

七、組員心得

103703013 資科三 黃育萱：

我負責的工作主要是將有文字輸出的資料圖形化，在組員剛寫完初步文字介面程式的版本，遇到的第一個問題就是作業系統的差別而不能執行（主因為 Mac OS 系統不能接受無名的 Semaphore 而有初始問題），所以找了很多方法嘗試，大概經過跟組員完成主程式程式碼的時間差不多。在終於能在電腦裡執行之後，當初先利用 QT Creator 原本預想在短時間內就能完成，卻因為不能先顯示 gui 介面再開始執行程式，也試著用 OpenGL 實作，因為都不能成功，先做過了一個以符號表示的介面，後來又試著將 pthread 改成 thread 及 mutex，最後經過跟同學討論中發現實作 gui 需要以函式呼叫才能正常同步執行（經測試後的 QT 也可以在 multi-thread 的環境執行 gui）。我覺得從討論到實作其實常常會遇到各種不同的問題，有很多是在跨平台的執行結果不同，就要猜測及檢視程式碼，經過反覆的測試才能完成這個專案。

103703015 資科三 蔡雨芝：

一開始在和組員討論、撰寫虛擬碼時，還算是順利。但後來加入隨機的客人之後，就出現了不少問題，有些小細節很容易被忽略，因此就要靠著多次嘗試才會發現 bug。在 Demo 的時候也發現原來我們誤解了部分題目的意思（例如 Poisson、Daemon），有些細節也沒有做得很好（Thread 數量的限制（沒辦法無限生成）、GUI 更新畫面速度太快等等）。在跨 OS 的時候也出現了許多編譯器的相關問題，還好後來找到 <dispatch> 函式庫才解決了相關問題。實作時雖然曾出現了許多跨 OS 的棘手問題，但也因此對於各種 OS 有更深入的了解。

103702012 心理三 陳冠聞：

這次的期末 project 剛開始覺得滿簡單的，只要做好 thread 的同步就好。但後來發現在一些題目的要求下，有些實作確實也有些難度，甚至我們誤解了一些題目的意思，有些功能也不能做到完美，如不能產生太多客人等等。本身身為非本系的學生，這學期透過這次期末 project 學到滿多東西的，例如 github 的操作、用途，撰寫不同部分 code 彼此之間的協調，測試結果錯誤如何找 bug 等等。

八、參考資料

1. C++ Reference <http://www.cplusplus.com/reference/>
2. pthread 程式撰寫 <http://blog.xuite.net/nikoung/wretch/154071878>
3. multi-thread 同步 <http://blog.yam.com/swwuyam/article/11459124>
4. C++ multi-thread 程式開發
<https://kheresy.wordpress.com/2012/07/06/multi-thread-programming-in-c-thread-p1/>
5. 使用 Signal 與 Slot (使用按鈕關閉視窗)
<http://openhome.cc/Gossip/Qt4Gossip/SimpleSignalSlot-1.html>
6. Semaphore on OS X
<http://stackoverflow.com/questions/1413785/sem-init-on-os-x>