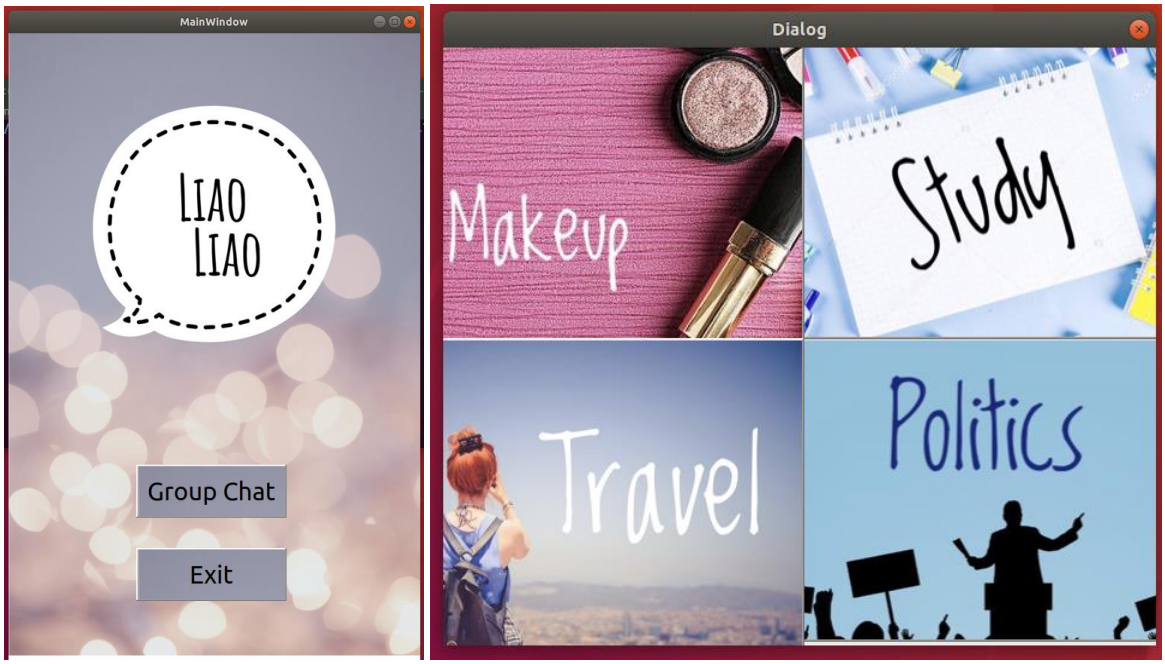
108 上學期程式設計期末作業

組名：期末一起加油

組員：b06303059 謝君模
b06208030 何承諭
b06303077 江雨柔
b08705031 陳沛竹

一、主題

我們這組做的是一個聊天平台——「聊聊」。「聊聊」是以各大常見的社會議題開放出不同的群組聊天室，讓大家可以以匿名的方式加入聊天，在這平台上打破同溫層、聆聽或是和立場不同的人們一起聊天、彼此分享各自的見解，分享彼此的觀點，在這開放且友善的交流平台中忠實地說出自己的看法或是對於當今局勢的分析與評論，也可以在這平台中接收來自各方最新的消息，不論是食衣住行育樂皆能討論，也不怕沒有人回應你！下面用實際的成果來蓋覽我們的「聊聊」。



(圖一)「聊聊」主頁面與聊天群組的選擇

在打開聊聊應用程式時，首先以簡約而可愛的的主畫面 (圖一左) 引導使用者開始進入聊天群組的選擇，在使用者點擊「Group Chat」按鈕時，將彈出聊天群組選擇的視窗 (圖一右) 供使用者選擇欲加入的聊天室，最後在使用者決定欲加入的聊天室後另外彈出新的聊天視窗 (圖二)。在群組聊天視窗中，所有的使用者姓名皆為系統自動分派，屬於匿名的聊天室，而使用者能夠自聊天室得到的資訊包含：(1) 當前聊天室名稱 (2) 當前聊天室人數 (3) 新聊天成員加入/退出聊天室 (4) 發言者與發言內容，透過伺服器與所有客戶端的連線，使用者能夠即時與所有用戶進行意見交流，以達成我們欲實作匿名聊天平台的動機。

(圖二) 聊天視窗，以政治版為例

二、程式架構

本專案的程式架構主要可以拆解成後端(server)與前端(client)。

後端的程式碼主要採用<socket.h>和<pthread.h>這兩個函式庫，前者是用來Linux系統上建立後端伺服器的資訊，我們是使用SOCK_STREAM(TCP)的方式來傳送訊息，相較於另外一種(DGRAM, UDP)，該傳輸方式可以更正確、更不會遺失訊息，比較貼近我們專案的特性；至於後者則是因為後端需要幫各個連線上的前端(使用者)提供「客製化、一對一」的服務，所以我們需要幫每一位使用者建立一個專屬於她的thread，至於每個thread的任務就是及時的把使用者所說的每一句話推播給所有的在線使用者。

前端的部分主要有兩大核心，一是圖形化使用者介面(GUI)的設計，二是當使用者進入聊天室後，前端需要幫使用者開啟一個多工(thread)來隨時等候後段傳送所有的訊息，一直到使用者離開聊天室才結束該多工。前者本專案是使用Qt所提供的各種元件(Widget)的套件來設計我們前端的GUI；後者本專案是採用Qt所提供的QTcpSocket以及QThread的函式庫來實現。

本專案後端的程式碼是架設在Linux的環境中，用tmux session讓後端一直運轉等待新使用者加入；至於使用者則是使用前端的程式碼部分來使用「聊聊」與他人「聊聊」。程式架構圖如下圖所示：

三、系統設計

本章節首先介紹前端的兩大核心：GUI和前後端溝通，最後再介紹後端的程式設計。

1. GUI

「聊聊」的介面風格主元素是清新風格，搭配按鍵式的操作模式打造一個乾淨簡潔的畫面感。因為我們是製作一個聊天平台，不需要太多操作而是要人們快速能找到自己想要聊天的群組。這次使用的程式套件是 Qt。在使用者見面我們主要使用的功能就是按鈕與儲存訊息，主要功能為開啟新視窗與傳送訊息。

i. 按鈕

```
void GroupMenuDialog::on_pushButton_clicked()
{
    Politics politics;
    politics.setModal(true);

    QRect screenGeometry = QApplication::desktop()->screenGeometry();
    int x = (screenGeometry.width()-politics.width()) / 2;
    int y = (screenGeometry.height()-politics.height()) / 2;
    politics.move(x, y);
    politics.show();

    politics.exec();
}
```

按鈕的部分居多是使用這個程式碼。除了可以感應滑鼠的位置還可以確認點

擊。後面再加上我需要的功能即可，不論是將文字串寫入其他頁面（對話框），或是新增新的頁面，或是跳除提示框都可以簡單的在裡面加入指令。

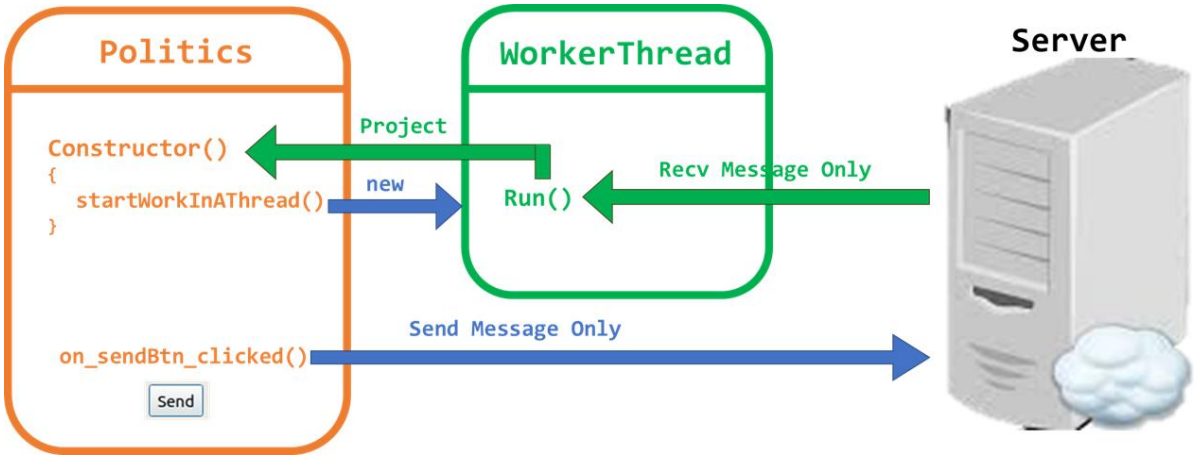
ii. 儲存訊息

```
{
    QStringList list;
    QString text = ui->lineEdit->text();
    list << text;
    list = model->stringList() + list;

    model->setStringList(list);

    ui->chatRecords->setModel(model);
//    string row = text.toStdString();
    ui->lineEdit->clear();
}
```

將訊息以陣列存取，除了紀錄按下send的那個句子，還要記住前面所有的句子方便紀錄到底有幾行，這樣才能有正確的排版。

2. 前後端溝通

前後端溝通的部分，我們捨棄了原先server所採用的<socket.h>和<pthread.h>，分別用Qt所提供的<QTcpSocket>和<QThread>來代替，相較於原本採用的函式庫，後者可以提升使用者的使用體驗，並且建立更完整的Qt前端。前後端溝通主要透過下列兩大物件的合作來實現，分別是聊天室視窗(ex: Politics)以及QThread的多工程序 (WorkerThread)。前後端的溝通主要就是傳送訊息與接收訊息，前者其實就是單純的在聊天室視窗物件(Politics)中直接讀取文字，透過socket直接傳送到後端(伺服器)上；後者就會需要多工程序 (WorkerThread)來幫助，因為前端(使用者)需要一直等待後端(伺服器)所推播的各種訊息一直到前端(使用者)關閉聊天室視窗為止。前後端的溝通機制如下圖所示：



下面重點介紹上述的兩個物件的架構，以及核心函式的公用。

1. 各聊天室的視窗 (ex: Politics)

```
class Politics : public QDialog
{
    Q_OBJECT

public:
    explicit Politics(QWidget *parent = nullptr);
    ~Politics();

private slots:
    void on_sendBtn_clicked();

private:
    QStringListModel* model = new QStringListModel(this);
    Ui::Politics *ui;  QTcpSocket* socket;  QString name;
    WorkerThread* workerThread;
    void disJoinServer();
    void startWorkInAThread();
    void handleResults();
};
```

上圖是聊天室的物件架構，主要牽涉前後端溝通的部分有：

* 建構子：

在各個聊天室的建立之初，除了和其他介面一樣會設定好GUI排版之外，最重要的是聊天室會呼叫自己的成員函式(startWorkInAThread)，new出一個我們另外定義好的物件(WorkerThread)來準備一直等待後端(伺服器)的訊息，詳細的運作情形請見下面。

* startWorkInAThread()：

此函式會被呼叫各聊天室的建構子所呼叫，其目的就是先開出一個WorkerThread物件，幫它設定好所需要的socket連線和會更動到的UI排版指標參數後，進行連線，接下來就全權交給WorkerThread來處理所有接收後端(伺服器)的訊息。

* 解構子：

在使用者關閉聊天室的那瞬間，這個聊天室的物件(Politics)首先會呼叫自己的成員函式(disJoinServer)，並且把所有的UI都清空、釋放記憶體空間。

* disJoinServer()：

此函式會被各個聊天室的解構子所呼叫。用來解除前端使用者和後端(伺服器)的連線，讓後端知道這位前端(使用者)已經退出聊天室，接著關閉多工(消滅WorkerThread)，釋放所有的動態記憶體配置空間。

* on_sendBtn_clicked：

此函式只是單純的把使用者所填入的訊息直接「傳輸」到後端(伺服器)上，接著便把該輸入欄的訊息清空，以便使用者輸入下一則訊息。

2. QThread的多工程序 (WorkerThread)

```
class WorkerThread : public QThread
{
    Q_OBJECT
    void run() override;

private:
    QTcpSocket* socket;
    QStringListModel* model;
    QListView* chatRecord;
    QLabel* userInfo;

public:
    void setSocket(QTcpSocket* s);
    void setStringListModel(QStringListModel* slm);
    void setListView(QListView* qsv);
    void setUserInfo(QLabel* label);

signals:
    void resultReady(const QString &s);
};
```

上圖是WorkerThread的物件架構，它是負責處理各個用戶的在聊天室中所有接收來自後端(伺服器)的訊息，並且把所有的訊息分解成四大類，再根據訊息的類別決定要如何插入對話框(聊天紀錄)中。主要的函式有：

* 四個setter functions：

這四個setter functions就是把該多工程序中會需要用到的參數以物件指標的傳入，主要分成兩類參數。

1. socket連線所需：

setSocket(QTcpSocket* s)

2. 會更動到的UI介面：

setStringListModel(QStringListModel* slm)

setListView(QListView* qsv)

setUserInfo(QLabel* label)

* run()：

這個函式就是WorkerThread這個物件執行的核心任務，就是一直等待後端(伺服器)推播訊息給前端(使用者)，一旦接收到訊息，它就會判斷這個訊息是屬於下列情形的哪一種，並且做出相對應的排版配置，所有的情形如下圖所示：

情境	句型	被更改者	排版處理
新用戶加入	<usrCntIn><usrCnt><Name>	QLabel* userInfo	-
		QListView* chatRecords	置中
舊用戶退出	<usrCntOut><usrCnt><Name>	QLabel* userInfo	-
		QListView* chatRecords	置中
傳送者訊息	message<sender>	QListView* chatRecords	置右
被傳送者訊息	message<sendee>	QListView* chatRecords	置左

3. 後端 (Server)

後端的部分我們是透過<socket.h>和<pthread.h>來架設伺服器，並且持續運行以等待隨時可能加入的新用戶。下左圖是後端(伺服器)的pseudo code，下右圖是後端多工程序(pthread)的核心任務的pseudo code。至於兩者之間的參數傳遞則是我們另外自定義一個結構(Argument)來傳入。

```
int main()
{
    set server information
    set pthread array (max: 10)

    while true // keep server alive
        find new client
        set pthread Argument

        if current user ammount is less than limit:
            create pthread for this client
        else:
            tell this client "Too many people!"
}
```

```
void* newUser(void* argsVoid)

< Part 0: pre-work >
    cast & parse Argument

< Part 1: new user comes in >
    broadcast to all current users who comes in

< Part 2: When the client is still alive >
    while true
        receive client message
        if me:
            send(message + <sender>)
        else:
            send(message + <sendee>)

< Part 3: If this client quit chat room >
    close client socket
    update alive people
    broadcast to all current users who leaves
```

四、分工方式

(1) 前端使用者介面 (GUI) 設計： 江雨柔、陳沛竹
(2) 前後端溝通：何承諭、謝君模
(3) 後端架設：何承諭、謝君模

五、未來展望

經過了本次期末作業，我們實作出了匿名聊天視窗程式的雛形，然而在有限的時間下，我們僅能夠做出第一個聊天室，且功能不如當初企劃書所述精緻，因此我們對於本次作業在未來的期望包含：

(1) 能夠達到跨系統溝通的理想目標
(2) 提高系統的穩定性 (目前在使用者網路不穩的情況下可能導致溝通失敗，伺服器炸裂)
(3) 增加聊天室 (初步想法為增開Port，在伺服器端 tmux 開放更多 session 運行)
(4) 聊天室的排版與美編 (添加聊天對話框與自動下滑至新訊息)
(5) 個人聊天與隨機聊天
(6) 表情包的引入 (初步想法為使用抽換特定關鍵字的方式實踐，如Line表情貼 (laugh))
(7) 話題趨勢走向 (初步想法為自最新100則訊息中以文字探勘的方式取出最新熱門議題)
(8) 隨機話題包 (按鈕式生成有趣的聊天主題以供使用者參考服用)

六、心得

b06303077 經濟三 江雨柔

在這次的專案當中，我們一開始其實想了非常多東西，有很快就開始著手了，而在很早期就遇到了困難，因為我們的系統不支援 <windows.h>，所以我們必須快速的找到可以相容的程式套件。一開始先找到了opencv，但其實他在實際使用上面來說非常的困難，也達不到我們期望的效果，再來他是屬於處理照片與影像聞名的，雖然在製作背景比較方便，但在導入訊息與輸出的時候遇到無解的狀況，只好再繼續找套件，最後找到了非常好用的Qt，真的是一個非常推薦給大家的軟體。在macox、window都能使用，更厲害的事他可以用python也可以用c++來寫，真的非常強大。其實在製作期末專案的時候我們花最多時間除了再處理程式相容性的問題，最長花的時間就是在找符合的套件。因為在製作聊天軟體的範例也不多，不像遊戲有相關的資料可以找尋，很多東西都是試一試發現無效，才趕緊去試一下一個東西。在製作按鈕的過重中，常常使用到pointer的觀念，隨然在讀取座標方面我們不用費心，因為他是一個已經寫好的class了，然而在添加功能時要更注意他究竟是指標，還是是一個陣列，搞清楚之後也發現pointer真的非常的方便，但就是要注意他的形式。不然很容易bulid fail。除此之外我在程式上遇到比較少的bug，因為他ui與cpp黨是整合成一個class，所以我在使用的時候只要include檔案即可使用，朝極無敵方便的。非常感謝其他兩位幫忙架設好伺服器與socket，也想到使用linux環境來編譯我們的client。一開始遇到mac杜能使用的時候其實蠻挫折的，運作起來都是機率的問題，這種不穩定性真的讓人很有挫折感，而且諸多情況都是失敗的……。除了在這個專案之中，在星期四的分享中，有一組講到了polymorphism的觀念，我真的覺得超酷的。很多組在寫遊戲所以他們時常用到state這個詞，可能是因為使用相同的編譯環境所做出來的，像我們的就單純很多，沒有那麼多state，硬要說的話大概只有兩個 mainmenu state跟 chat state，我們真正難在socket的傳輸與多工的處理。自我評比的部分，這次我總算有參與比較多，沒有愧對我的組員了，這次的期末專案真的很棒，也對自己的表現很滿意。同時也謝謝這群這麼棒的組員。

b06208030 地理三 何承諭

在這次期末專案中，我們嘗試了很多沒有接觸過的新事物，從試著理解 Server 與 Client 的互動模式、Socket 套件的使用方法、GUI 的設計方式、Socket 與前端的銜接、至跨作業系統連線的困境等等，每一項任務對於我們來說都是困難的挑戰，不斷的撞牆也迫使我們突破與進步。

在整體專案中，我認為最困難的部分並不是研究 Server 與 Client 溝通的程式 (使用 <socket.h> 套件包) 該怎麼寫，因為伺服器與客戶端的連線教學在網路上已經非常完備，基本上我們只需要實際去理解 Socket 的相關概念與函式庫加以嘗試便能夠實做出來，相較而言我們認為更為棘手的問題是研究如何將我們使用 Socket 套件實做出來的 Client 與 GUI 做整合。

Qt 在 Socket 使用上提供了 <QTcpSocket> 套件包，其中有一系列方便的函數可以在客戶端使用 (如使用 connectToHost() 連線至 Server 、waitForReadyRead() 用來 Block 程式的進程以等待接收訊息、write() 將訊息寫至 Server 端、readAll() 接收 Server 訊息等等)，在實作的過程當中我們非常害怕 Qt 的 Socket 會連線不上我們自己使用 <socket.h> 架設的伺服器 (由於我們是先將 Server 與 Client 端寫好後再來考慮前後端的整合)，所幸經過一番努力後過後 QTcpSocket 是有辦法成功的連線上我們的伺服器的。

在成功能夠使用 QTcpSocket 連線伺服器過後，下一件更麻煩的事情是怎麼 Client 端保持著接收 Server 訊息的狀態。邏輯上來說，這件事情應該使用一個 while 迴圈包著讓它持續地運行，但同一時間它也要持續地正常運作，讓使用者能夠打字傳送訊息出去，因此在這個情況下我們選擇使用多工 (Thread) 的方式來同時進行收發訊息的兩件事情。在 Qt 中它提供 <QThread> 函式庫供開發者使用，但在經過一番研究過後，想要解讀官方網站提供的程式碼並非如同我們想像中那麼容易，經過了兩個全天投注在多工的研究上，我們才成功地讓程式跑起來，最後再加上置左置右排版以及使用者加入退出的功能，我們的聊天室才算是大功告成。

在時間限制與不夠充足的知識背景之下，在某些功能當中我們並沒有辦法實做出來，導致我們最後的應用程式僅能於 Linux 作業系統使用，仍有一些無法解決的問題以及可進步的空間，但經過了這次期末專案，我們學習到了許多新東西，也能夠實際活用課堂上學習到的 OOP 概念，這些感受與當初受挫折的期中專案完全不同(慘烈的期中專案)，能夠在期末以實作出一個聊天應用程式的雛型作結，讓我對這個學期所學習到的事物感到滿意，謝謝組員們的努力付出，大家都辛苦了，也謝謝老師助教在這個學期中的指導！