操作說明文件

✍️ B10101053 資管三 邱冠銘

環境需求

在執行 Server 端程式之前，請確保系統已安裝以下環境：

- **C++ 編譯器**：建議使用 g++，版本需支持 C++17。
- **Make 工具**：用於簡化編譯過程。
- **套件**：確保系統已安裝必要的套件（如 `sys/socket.h`）。

編譯與執行

1. **使用 Makefile 編譯**：在根目錄執行以下命令以編譯程式，這將使用 `g++` 編譯所有的 `.cpp` 檔案並生成可執行檔 `client.out` 以及 `server.out`。

   ```
   make
   ```

2. **啟動伺服器**：在執行 Client 之前，請確保伺服器端程式已經啟動並在指定的 IP 和 port 上運行。

3. **執行 Server 程式**：使用以下命令來執行 Client 程式，並提供伺服器的 IP 和 port 作為參數：

   ```
   ./server.out <server-port> -a
   ```

   例如：

   ```
   ./server.out 8080
   ```

4. **執行 Client 程式**：使用以下命令來執行 Client 程式，並提供伺服器的 IP 和 port 作為參數：

```
./client.out <server-ip> <server-port>
```

例如：

```
./client.out 127.0.0.1 8080
```

Client 功能簡介

1. **註冊 (`REGISTER` or `a`)**：使用者可以選擇註冊新帳戶。程式會提示輸入使用者名稱並傳送到伺服器，伺服器回傳成功或失敗的訊息。若註冊失敗，程式會顯示錯誤訊息，讓使用者重新輸入。
2. **登入 (`LOGIN` or `b`)**：使用者選擇登入後，程式會提示輸入使用者名稱和監聽的 port。若輸入的 port 無效（例如包含非數字字元），介面會顯示「Invalid port」。成功登入後，會建立一個監聽執行緒來處理來自其他使用者的訊息。
3. **查詢可用使用者列表 (`LIST` or `c`)**：使用者可以查詢當前線上的其他使用者。程式會發送查詢請求並接收伺服器回傳的線上使用者列表。
4. **交易 (`PAY` or `d`)**：選擇交易功能後，程式會提示使用者輸入付款人、收款人以及金額，並進行基本檢查。若輸入的金額包含無效字元（如非數字字元），會顯示「Invalid amount」錯誤訊息。若收款人不存在或不在線上，程式會顯示錯誤訊息並取消交易。
5. **退出 (`EXIT` or `e`)**：選擇退出後，程式會顯示退出訊息並正常關閉客戶端連線。
6. **CLI：**透過 ascii art 做介面顯示，讓使用者更加快速使用介面。並使用互動介面簡化互動流程。
7. **計時器：在**輸入指令前，印出當前時間。並在結束時，印出使用時間。

Server 功能簡介

1. **註冊功能 (`REGISTER#username`)**
   - 透過 `processRequest` 處理請求。
   - 若用戶名不存在於 `userAccounts` 中，新增該用戶並初始化餘額為 10,000。
   - 回應註冊成功訊息 (`100 OK`) 或失敗訊息 (`210 FAIL`)。

2. **登入功能 (`username#port`)**
    - 解析用戶名和 port。
    - 驗證用戶是否存在於 `userAccounts` 且未重複登入。
    - 若成功登入，將用戶加入 `onlineUsers`。
    - 回傳用戶餘額、公鑰及在線用戶清單；失敗則回傳 `220 AUTH_FAIL`。
3. **列出在線用戶 (`List`)**
    - 登入成功後，客戶端可透過 `List` 查詢當前在線用戶。
    - 回傳用戶餘額、公鑰和在線用戶清單。
4. **轉帳功能 (`payer#amount#payee`)**
    - 解析付款人、收款人和金額。
    - 驗證雙方是否存在於 `userAccounts`，且餘額是否足夠。
    - 成功轉帳後更新雙方餘額，並通知付款人操作成功或失敗。
5. **登出功能 (`Exit`)**
    - 客戶端請求登出，從 `onlineUsers` 中移除用戶。
    - 回傳 `Bye` 訊息並關閉該客戶端的 socket。

安全傳輸實作方法與流程

1. 實作方法
    a. **生成 RSA 密鑰 (`generateRSAKey(int bits)`)**
        - 使用 `EVP_PKEY_CTX_new_id` 創建上下文，指定密鑰類型為 RSA。
        - 調用 `EVP_PKEY_keygen_init` 初始化上下文，準備生成密鑰。
        - 設定密鑰長度（如 2048 或 4096 位元）。
        - 調用 `EVP_PKEY_keygen` 生成密鑰並返回。
    b. **提取密鑰字串 (`getPublicKey()` 和 `getPrivateKey()`)**
        - 利用 OpenSSL 的 `BIO`（內存緩衝區）儲存密鑰。
        - 用 `PEM_write_bio_PUBKEY` 和 `PEM_write_bio_PrivateKey` 將公鑰私鑰寫入緩衝區。

- 提取 `BIO` 緩衝區中的字串，返回 PEM 格式的密鑰。

c. **從字串加載密鑰 (`loadPrivateKey()` 和 `loadPublicKey()`)**

- 創建 `BIO` 緩衝區，並將 PEM 格式密鑰內容載入。
- 使用 `PEM_read_bio_PrivateKey` 和 `PEM_read_bio_PUBKEY` 分別讀取私鑰和公鑰。
- 返回對應的 `EVP_PKEY` 物件。

d. **加密 (`encryptMessage()`)**

- 加載公鑰，創建加密上下文。
- 初始化加密並測量加密後的緩衝區大小。
- 使用 `EVP_PKEY_encrypt` 實現加密，將輸入字串加密為二進制資料。
- 返回加密結果。

e. **解密 (`decryptMessage()`)**

- 加載私鑰，創建解密上下文。
- 初始化解密並測量解密後的緩衝區大小。
- 使用 `EVP_PKEY_decrypt` 實現解密，將二進制資料轉為原始字串。
- 返回解密結果。

2. server - client
    - 連線後，server & client 雙方交換公鑰，server 會記錄所有 client 之公鑰以及基本資訊
    - client 使用 server 的公鑰進行訊息的加密，送出訊息給 server
    - server 收到訊息後使用私鑰進行解密
    - server 使用 client 的公鑰進行訊息的加密，送出回覆給 client
    - client 收到訊息後使用私鑰進行解密
3. client - client
    - 連線後，sender & receiver 雙方交換公鑰
    - sender 使用 receiver 的公鑰進行訊息的加密，送出訊息給 receiver
    - receiver 收到訊息後使用私鑰進行解密

- receiver 使用 server 的公鑰進行訊息的加密，送出訊息給 server
- server 收到訊息後使用私鑰解密
- server 使用 sender 的公鑰進行訊息的加密後，送出傳輸結果訊息給 sender
- sender 收到訊息後使用私鑰進行解密