網路技術與應用：程式作業 Demo1

賴楠天

資管三 B12705010

1 編譯方法

本次作業的 Client 程式以 C 語言撰寫，並使用 GCC 作為主要編譯器。

(1) 使用 **Makefile** 編譯

在作業目錄下執行：

```
make client
```

成功後會生成可執行檔：

```
client
```

若需要重新編譯或清除舊的執行檔，執行：

```
make clean
```

(2) 手動編譯方式

若不使用 Makefile，也可以直接使用 GCC 編譯本次作業程式：

```
gcc -Wall -Wextra -O2 client.c -o client
```

2 程式執行環境說明

本作業之 Client 端程式使用 C 語言撰寫，並在 Linux 環境中完成編譯與測試。以下說明實際執行與測試時的系統環境、編譯器版本與網路設定。

2.1 作業系統環境

本次程式於虛擬機器 VirtualBox 中運行，作業系統為：

- Ubuntu 20.04.6 LTS（64-bit）
- Linux Kernel 5.x 系列

所有 client 端測試皆在此 Linux 環境中執行，並以 SSH 從另一台電腦遠端登入同一資料夾，以同時操作 server 與 client。

2.2 編譯器與工具鏈

程式使用 gcc 進行編譯：

- GCC 版本：gcc 9.4.0
- 編譯指令：`gcc -Wall -Wextra -O2 client.c -o client`

此外，也提供 Makefile 進行簡化編譯：

2.3 執行與測試方式

測試架構如下：

- Server 程式執行於同一虛擬機器中，使用指定 port（如 8888）
- Client 程式亦在同一資料夾中執行，可同時啟動多個 client 進行 P2P 測試
- 不同 client 之間的 P2P 傳輸以 TCP socket 完成

啟動 client 的指令格式如下：

```
./client <server_ip> <server_port>
```

例如：

```
./client 127.0.0.1 8888
```

2.4 網路配置方式

由於 server 與 client 均位於同一台虛擬機器內，使用本機回 loopback：`127.0.0.1`

P2P 連線部分會依照使用者登入時輸入的 port（如 9999、5678 等）在本機額外建立一個 listening socket，並讓其他 client 直接連入。

而且，家中剛好還有另一台 Linux 虛擬機（版本 Ubuntu 22.04.4 LTS ），因此也測試了如同和助教 Demo 那天相同的方式：在 A 機啟動 server、在 B 機啟動 client，兩台皆連上同一個區域網路，透過查詢各自的 IP 進行連線與互相轉帳。這部分實作起來相當有趣，也更直觀理解 P2P 與 TCP 連線在實際網路環境中的運作方式。

2.5 執行需求

程式執行不需額外安裝 OpenSSL 或外部函式庫，此為 HW1 要求之純 TCP socket 程式。

程式所需的系統功能包含：

- POSIX socket API（`socket`、`connect`、`bind`、`listen`、`accept`）
- `select()` 多工監聽
- Linux 檔案描述元（STDIN、socket 等）

所有功能皆為 Linux 預設支援之系統呼叫，無額外依賴。

3 執行 Client 端程式操作流程

本節說明如何啟動 client 程式，以及實際操作註冊、登入、查詢與轉帳的流程。

3.1 啟動 Client

在終端機切換到程式所在目錄後，使用下列指令啟動 client，並連線到指定的 server IP 與 port：

```
./client <server_ip> <server_port>
```

例如本作業中，server 跑在本機的 8888 port，啟動畫面如下：

```
./client 127.0.0.1 8888

===== Connected to 127.0.0.1:8888 =====

#==================== User Info ====================#

```

```
User : (not login)    Balance: 10000
IP   : N/A            Port    : 0

                           Commands
   REGISTER#name           a#amount#b              Exit
   name#port               List

>
```

啟動後畫面上會先顯示目前的使用者資訊（User、Balance、IP、Port），以及可用的指令格式說明，最後一行的 > 為使用者輸入指令的位置。

3.2 註冊帳號：REGISTER#name

第一次使用時，需要先向伺服器註冊使用者名稱，指令格式為：

```
REGISTER#<name>
```

例如輸入：

```
> REGISTER#Sherlock
```

若名稱不符合伺服器規定，或已被其他使用者使用，伺服器會回傳類似下列訊息：

```
#====== Server Reply: ======#
210 FAIL
#============================#
```

在這個狀態下，User Info 中仍會顯示 (not login)，代表尚未完成登入。

3.3 宣告監聽埠並登入：name#port

註冊成功後，接著在 client 端輸入自己的名稱與 P2P 監聽埠，格式為：

```
<name>#<port>
```

例如：

```
> Sherlock#2212
```

程式會在本機建立一個監聽 socket，供其他使用者連入進行 P2P 轉帳，畫面會出現：

```
[INFO] Listening on port 2212 for P2P transfers
```

接著，伺服器會回傳目前餘額與線上使用者清單：

```
#====== Server Reply: ======#
10000
public key
1
Sherlock#127.0.0.1#2212
#============================#
```

client 會解析這段回應並更新畫面上的 User Info，例如：

```
User : Sherlock      Balance: 10000
IP   : 127.0.0.1     Port   : 2212
```

3.4 查詢線上使用者清單：List

登入後，可以輸入 List 指令向伺服器查詢目前的餘額與線上使用者列表：

```
> List
```

伺服器回覆範例如下：

```
#====== Server Reply: ======#
10000
public key
1
Sherlock#127.0.0.1#2212
#============================#
```

client 會將這段回應解析為：

```
Balance: 10000
ServerKey: public key
[INFO] 1 users online:
  - Sherlock@127.0.0.1:2212
```

並同步更新畫面上方的 User Info 區塊。

3.5 P2P 轉帳：a#amount#b

P2P 轉帳的指令格式為：

```
<sender>#<amount>#<receiver>
```

例如由 John 轉 5000 給 Sherlock 的情況下，輸入：

```
> John#5000#Sherlock
```

client 端會先在本地確認輸入是否合法，並顯示確認訊息：

```
[LOCAL] 確認 John → Sherlock (5000)
[INFO] 已送出轉帳請求：John → Sherlock (5000)
```

若伺服器端轉帳成功，會回應：

```
#====== Server Reply: ======#
Transfer OK!
#============================#
```

在接收方 Sherlock 的終端機上，則會看到 P2P 收款提示與更新後的 List 回應，例如：

```
[P2P] John sent you 5000

#====== Server Reply: ======#
15000
public key
2
John#127.0.0.1#2213
Sherlock#127.0.0.1#2212
#===========================#
```

client 端會據此更新餘額與線上使用者清單。

3.6 結束程式：Exit

若要離開系統，輸入：

```
> Exit
```

client 會關閉與伺服器的 TCP 連線以及本地的監聽 socket，並正常結束程式。

4 參考資料、來源

- ChatGPT
- Gemini AI Pro
- Google 搜尋
- 上課 Socket Programming 投影片
- 家人與同學
- 助教 Demo 時的說明 (需由接收方打 List 才會成功轉帳的問題已解決)