

Socket programming HW1

資管五 徐遠志 b05705046

- 如何 Compile

開啟 terminal 然後 cd 到 client.cpp 的目錄下，執行：

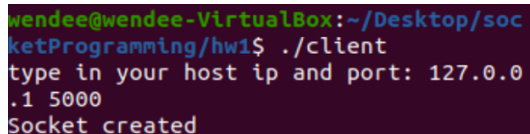
```
g++ -o client client.cpp
```

- 如何執行程式 (包含參數說明)

在 terminal 執行：

```
./client
```

接著程式會問 host address 和 port，就輸入 server 所在的位置和 port，中間用空白隔開：



```
wendee@wendee-VirtualBox:~/Desktop/socketProgramming/hw1$ ./client
type in your host ip and port: 127.0.0.1 5000
Socket created
```

看到 Socket created 就表示成功連上 server 了

- 程式需求、執行需求或環境

虛擬機器安裝 linux 20.04 版本的 ISO 檔

需要 g++ 進行編譯

- 程式邏輯說明與截圖搭配

由於作業需要寫一個 client，既可以和 server 溝通，又要向其他 client 傳輸交易訊息，所以在程式一開始我們就要先執行 fork，fork id == 0 的負責聽有沒有人要和我進行連線。這裡先讓 8888 當成 client 之間溝通的 port (當然之後想增加延展性可以讓用戶手動輸入要用幾號 port)

```

216 | pid_t pid = fork();           You, a day ago • add fork in client
217 | if (pid == -1)
218 | {
219 |     perror("fork fail");
220 | }
221 | else if (pid == 0) // child id, listen and print
222 | {
223 |     string client_add = "127.0.0.1";
224 |     int client_port = 8888;
225 |
226 |     client.createSocket(false, client_add, client_port);
227 |     client.listen_port();
228 | }

```

如果 fork id > 0 的則是送\收訊息的主要區塊，一開始會先和 server 建立 socket，確定連線成功之後，會讓使用者決定現在要送訊息給 h (host，也就是 server) 或是 c (client，也就是要進行 micropayment 的部分)

```

client.createSocket(true, host, server_port); //connect to host
client.isServerConnected = client.connection(true);
client.receive(true);

while (client.isServerConnected)
{
    string receiver;
    cout << "send command to host or client (h/c) ? ";
    cin >> receiver;
}

```

如果使用者想要跟其他的 client 傳送訊息的話，withHost 會是 false，這時會先檢查是否已經有和其他的 client 進行 socket 的建立和連線了，還沒有的話，使用者就先輸入欲連線的 client address 和 port 號碼，接著就建立 socket。

```

bool withHost = receiver == "h";
if (!withHost && !client.getClientConnectStatus())
{
    string client_ip;
    int client_port;
    cout << "type in client receiver's ip and host: ";
    cin >> client_ip >> client_port;
    client.createSocket(withHost, client_ip, client_port);
    client.connection(withHost);
}

```

接著就可以輸入 command，並將 command 傳送給相對應的接收者(server 或是 other client)，receive() 則會將對方回傳的訊息印在 terminal 上。

```
// send message
string command;
cout << "command: ";
cin >> command;
client.send_data(withHost, command);
client.receive(withHost);
```

/* 一些重要 function 介紹 */

- createSocket

socket(domain, type, protocol) 是用來針對指定的 domain 和 type 建立 socket

- domain = AF_INET 是指 Internet Domain
- type = SOCK_STREAM 則表示我們建的是 stream socket (which provides two way communications between a client and server)
- protocol 放 0 則讓 system 自行選擇一個 protocol 來支援我們要求的 socket type

建立完 socket 之後，我們會指定這個 socket 的 ip (sin_addr)、address family (sin_family)、port (sin_port)

```
bool Client::createSocket(bool withHost, string address, int port)
{
    if (withHost)
    {
        // create socket if it is not already created
        if (server_sock == -1)
        {
            server_sock = socket(AF_INET, SOCK_STREAM, 0); // Create socket
            if (server_sock == -1)
            {
                perror("Could not create socket");
            }
            cout << "Socket created\n";
        }

        server.sin_addr.s_addr = inet_addr(address.c_str());
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
    }
}
```

參考：<https://docs.oracle.com/cd/E19620-01/805-4041/6j3r8iu2l/index.html>

- connection

`connect(int socket, struct sockaddr *address, int address_len)` 會負責建立兩個 socket 之間的連線，socket 參數放的是開啟這個 connection request 的 socket descriptor，所以想跟 server 進行連線就在此參數放剛剛在 `socket()` 回傳的 socket descriptor

```
bool Client::connection(bool withHost)
{
    if (withHost)
    {
        // Connect to remote server
        if (connect(server_sock, (struct sockaddr *)&server, sizeof(server)) < 0)
        {
            perror("server connect failed.");
            return false;
        }
    }
}
```

參考:

https://www.ibm.com/support/knowledgecenter/en/SSB27U_6.4.0/com.ibm.zvm.v640.edclv/connect.htm

- Listen_port

針對 client 之間的溝通，我們需要建立一個 socket，並且透過：

- `bind()` 來指派一個 address 給它，讓其他 client 可以指定要 access 到這個 address 所指的 socket
- `listen()` 表示這個 socket 已經可以接受 connection request 了，5 是指最多五個 request 會在 queue 裡等待連線

```
void Client::listen_port()
{
    int bindStatus = bind(client_sock, (struct sockaddr *)&clientReiever,
                          sizeof(clientReiever));
    if (bindStatus < 0)
    {
        cout << "Binding port result: " << bindStatus << endl;
        cerr << "Can't binding socket to local address!" << endl;
        exit(0);
    }

    listen(client_sock, 5);
}
```

- 若有 connection request 的話，`accept()` 會回傳一個新的 socket descriptor (which is connected to the requesting client)
- 接著就透過 `receive()` 來接收新建的 connection 有傳什麼訊息

```

while(true)
{
    new_sock = accept(client_sock, (sockaddr *)&newSocketAddr, &newSocketSize);
    if (new_sock < 0)
    {
        cout << "new_sock fail: " << new_sock << endl;
        cerr << "Can't accepting the request from client!" << endl;
        exit(0);
    }

    cout << "Connection accepted from " << inet_ntoa(newSocketAddr.sin_addr) << endl;
    while (true)
    {
        receive(false);
    }
}

```

- send_data

send(int socket, char *buffer, int length, int flags) 會將 buffer 裡的内容傳送到指定的 socket

```

bool Client::send_data(bool toHost, string data)
{
    cout << "Sending data to ";

    if (toHost)
    {
        cout << "host..." << data << "\n";
        if (send(server_sock, data.c_str(), strlen(data.c_str()), 0) < 0)
        {
            perror("Send failed : ");
            return false;
        }
    }
}

```

參考：

https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2.bpxbd00/snd.htm

- receive

recv(int socket, char *buffer, int length, int flags) 會將指定 socket 裡接收到的資料存進 buffer，整個 recv 函式會回傳接收到的資料長度，因此長度大於 0 的話，我們就把資訊印在 terminal 上

```
void Client::receive(bool fromHost)
{
    char buffer[2000] = {0};
    memset(buffer, '\\0', sizeof(buffer));

    if (fromHost)
    {
        if (recv(server_sock, buffer, sizeof(buffer), 0) > 0)
        {
            cout << "response from server : \\n" << buffer << "\\n";
        }
    }
}
```

參考：

https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2.bpxbd00/rcv.htm#rcv