Socket Programming

B05202044 林展慶

Outline

---

1. 如何compile
2. 環境
3. 如何執行程式
4. 程式需求、執行需求
5. 程式邏輯說明
6. 檔案說明
7. 所實做的各功能
8. bonus

如何compile

```
run the following command

make
```

環境

---

ubuntu16.04

如何執行程式

Server:

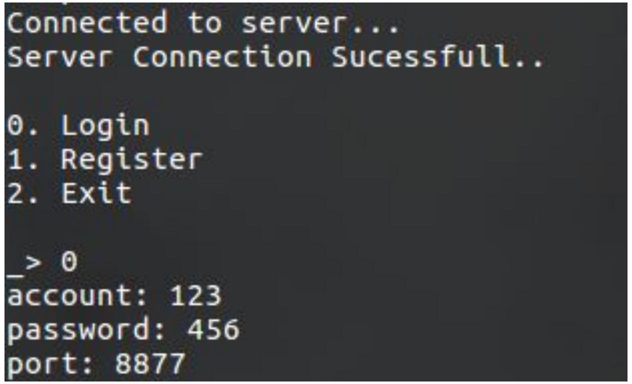
```
./server [port]
```

Client:

```
./client [ip] [port]
```

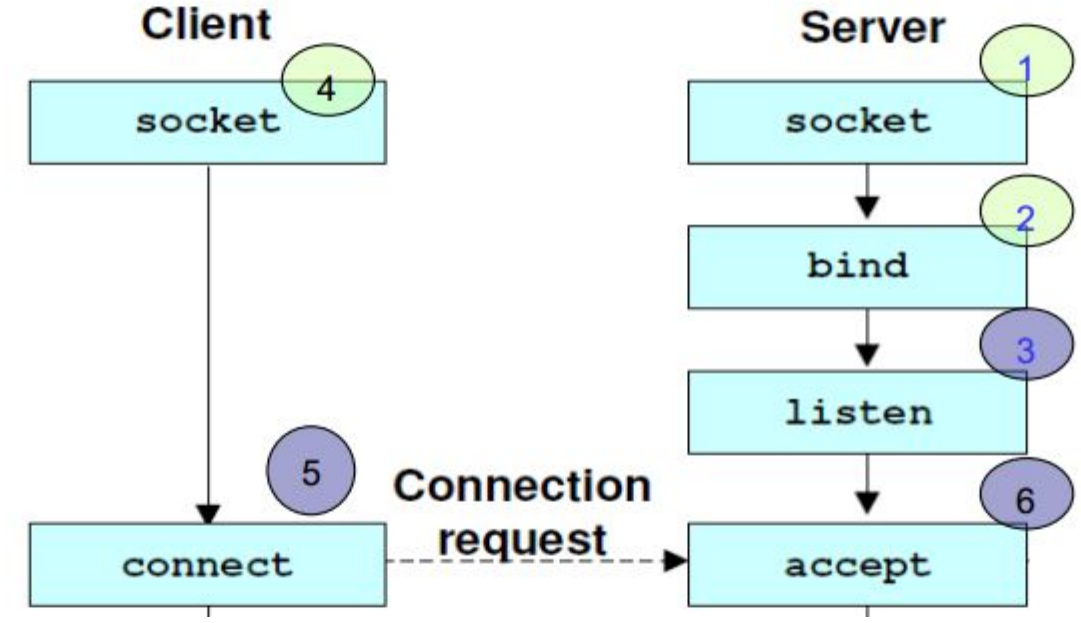
程式邏輯說明

Interactive Shell



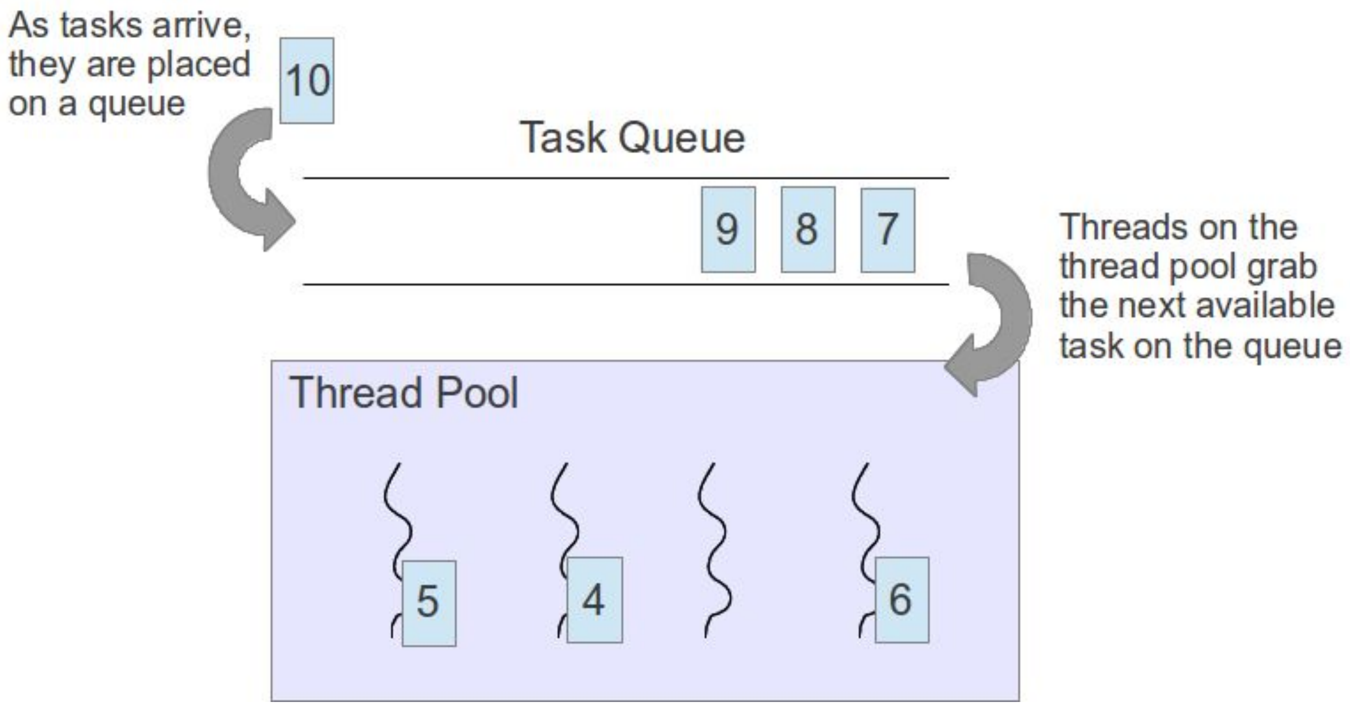
```
Connected to server...
Server Connection Sucessfull..

0. Login
1. Register
2. Exit

_> 0
account: 123
password: 456
port: 8877
```

Connection

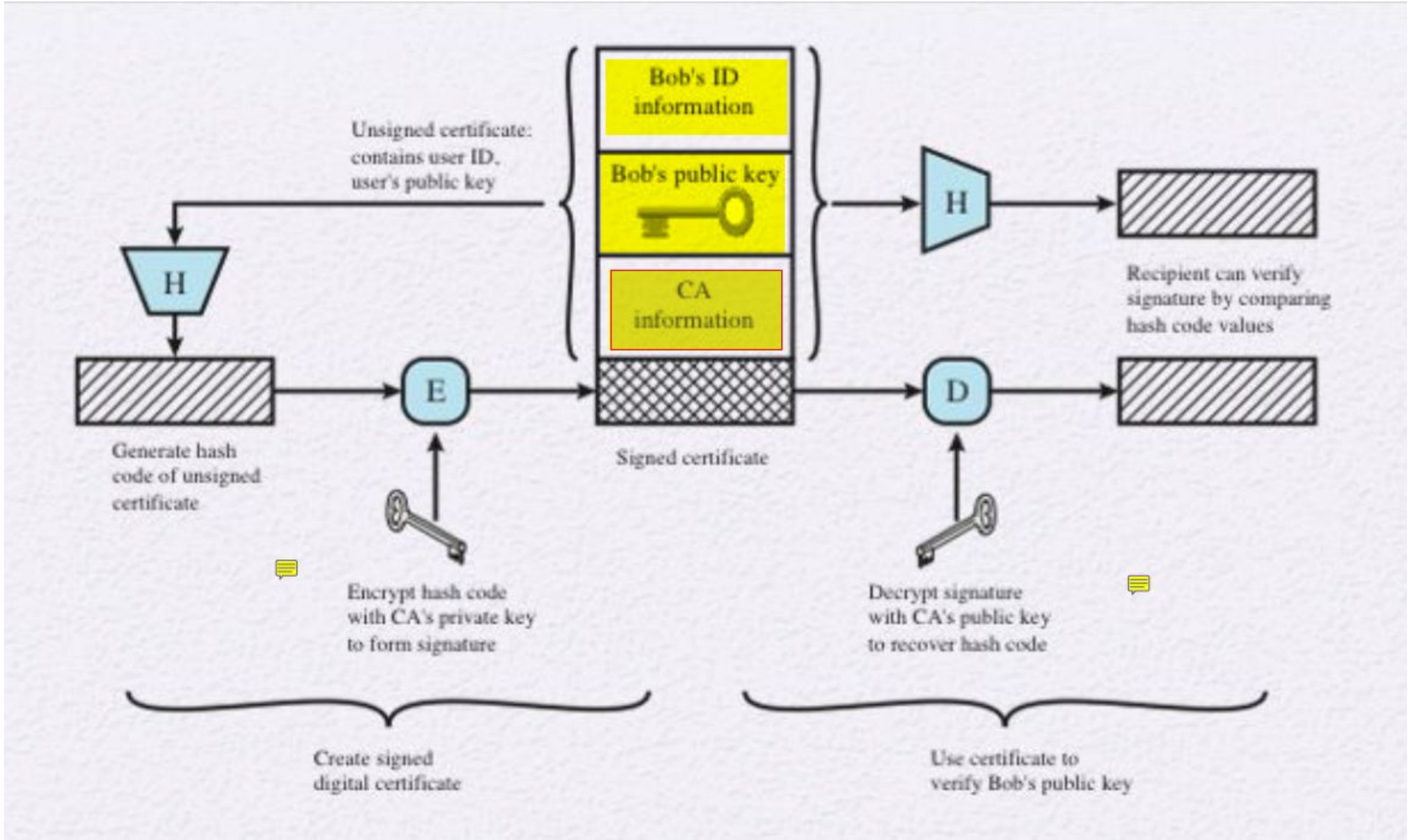


Client
Server
4
socket
1
socket
2
bind
3
listen
5
Connection request
6
connect
accept

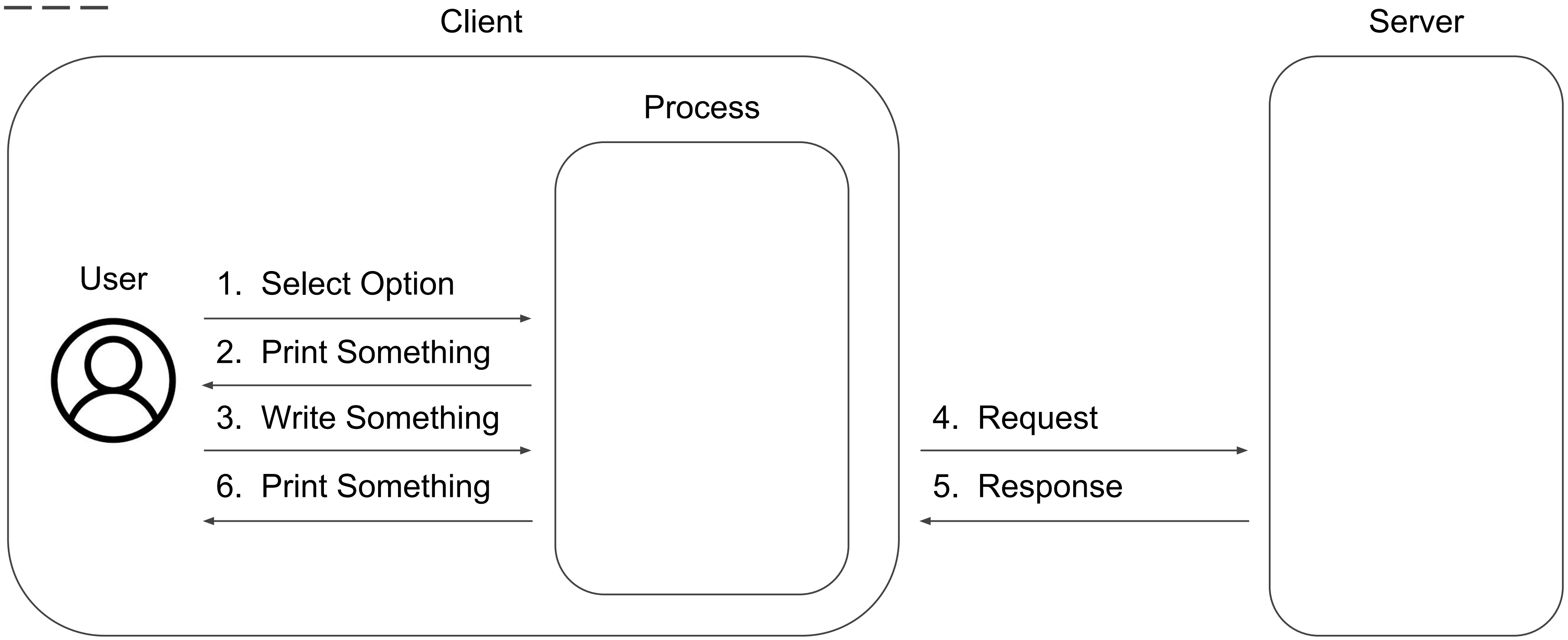
thread pool


As tasks arrive, they are placed on a queue
10
Task Queue
9
8
7
Threads on the thread pool grab the next available task on the queue
Thread Pool
5
4
6

Authentication


Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
H
E
D
Recipient can verify signature by comparing hash code values
Generate hash code of unsigned certificate
Signed certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Request & Response



Request Packet

- Encrypted Header + Body
  - Header
    - Method
    - Path
    - Parameter
    - Cookie
  - Body
    - Content

Response Packet

- Encrypted Header + Body
  - Header
    - Status Code
  - Body
    - Content

檔案說明

src

---

source code

底下有client、server、packet、util四個資料夾分別放置

1. client的source code
2. server的source code
3. pakcet的source code(about the implementation of request packet and responce packet)
4. util的source code(about some common function and macro)

db

---

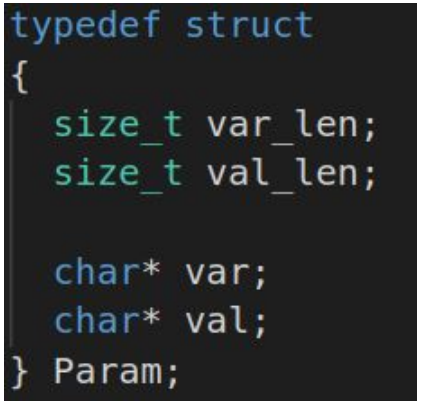
database，用來放置CA、server與client的資料

- client一旦註冊成功，就會在db/client上面建立資料夾，並且會用來紀錄client的動作
- server用來放置server的private key、certificate
- root用來放置root的private key和certificate

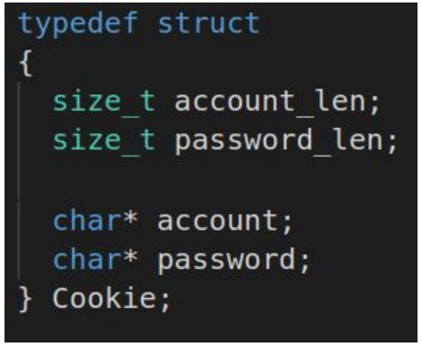
其餘

- root.crt
    - 用來給client讀取的第三方公正 certificate
- Makefile
    - compile
- readme.md
    - 說明

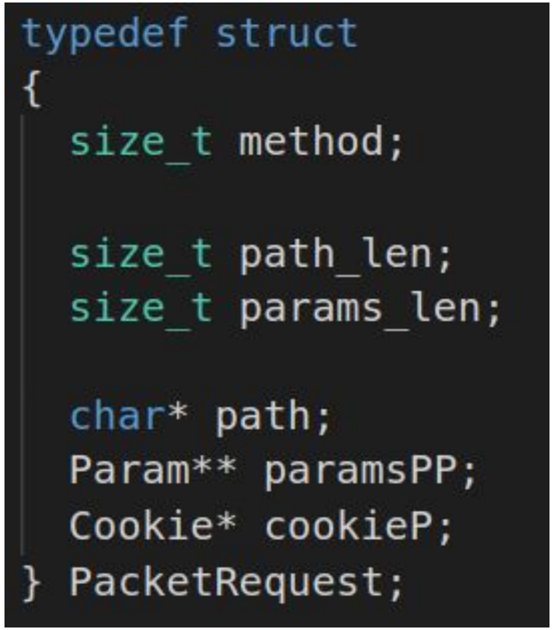
packet/packet.h - struct



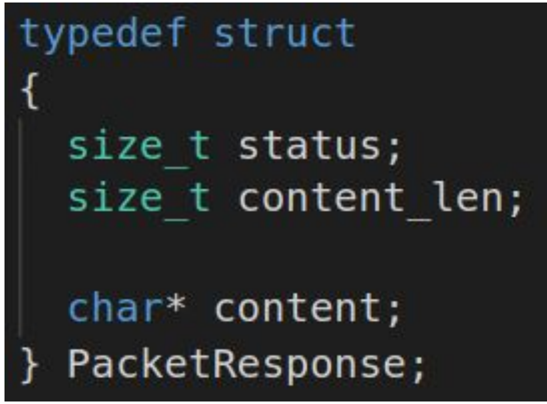
```c
typedef struct
{
  size_t var_len;
  size_t val_len;

  char* var;
  char* val;
} Param;
```



```c
typedef struct
{
  size_t account_len;
  size_t password_len;

  char* account;
  char* password;
} Cookie;
```

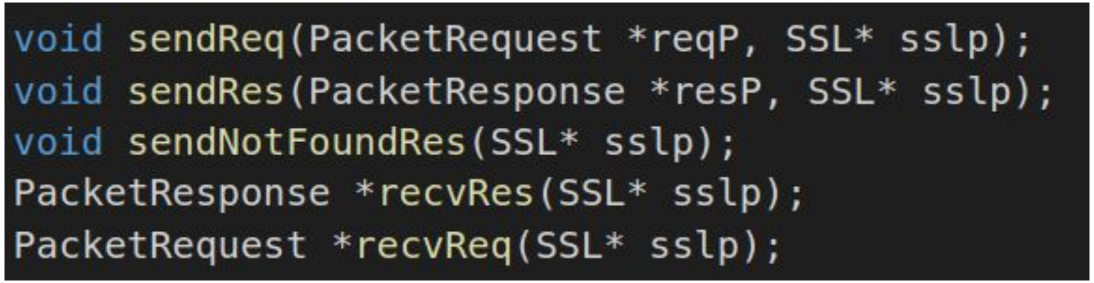


```c
typedef struct
{
  size_t method;

  size_t path_len;
  size_t params_len;

  char* path;
  Param** paramsPP;
  Cookie* cookieP;
} PacketRequest;
```

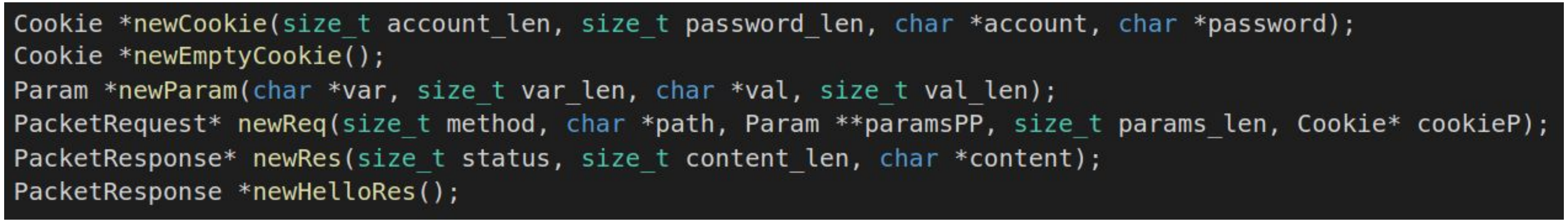


```c
typedef struct
{
  size_t status;
  size_t content_len;

  char* content;
} PacketResponse;
```

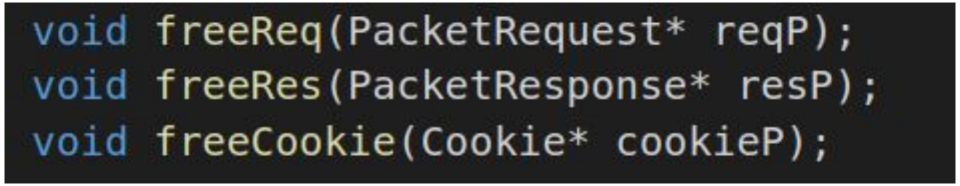
packet/packet.h - function



```c
void sendReq(PacketRequest *reqP, SSL* sslp);
void sendRes(PacketResponse *resP, SSL* sslp);
void sendNotFoundRes(SSL* sslp);
PacketResponse *recvRes(SSL* sslp);
PacketRequest *recvReq(SSL* sslp);
```



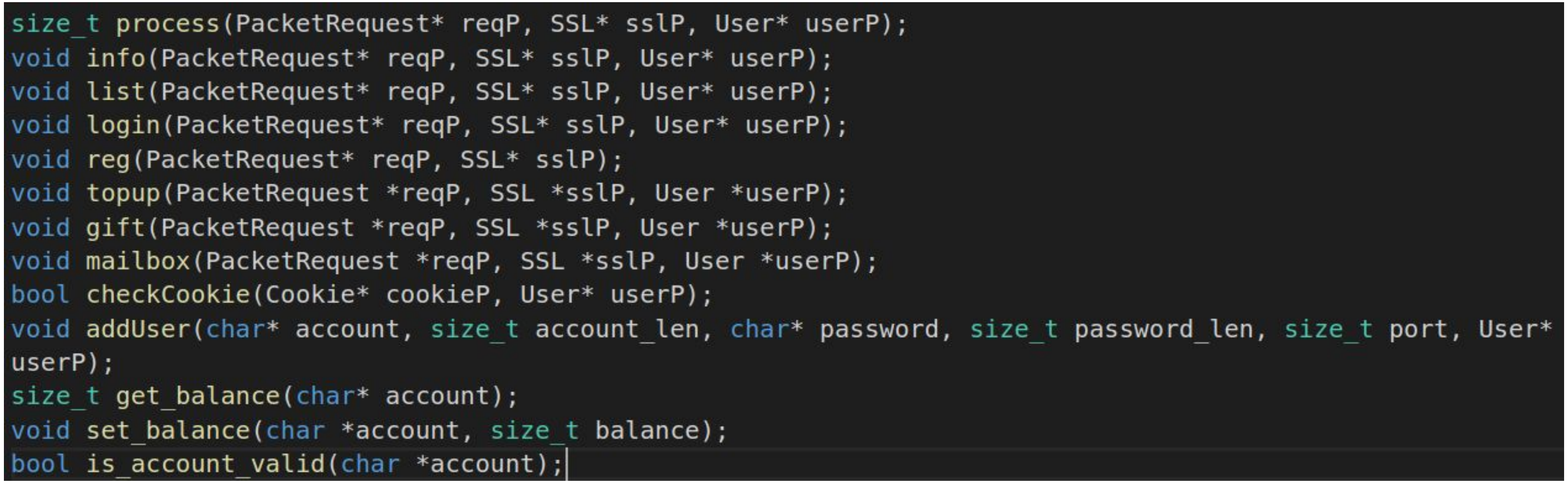
```c
Cookie *newCookie(size_t account_len, size_t password_len, char *account, char *password);
Cookie *newEmptyCookie();
Param *newParam(char *var, size_t var_len, char *val, size_t val_len);
PacketRequest* newReq(size_t method, char *path, Param **paramsPP, size_t params_len, Cookie* cookieP);
PacketResponse* newRes(size_t status, size_t content_len, char *content);
PacketResponse *newHelloRes();
```



```c
void freeReq(PacketRequest* reqP);
void freeRes(PacketResponse* resP);
void freeCookie(Cookie* cookieP);
```

server/process.h

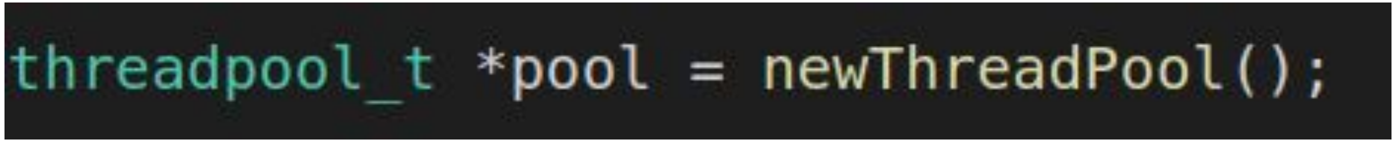
//according to the path to process the request packet



```c
size_t process(PacketRequest* reqP, SSL* sslP, User* userP);
void info(PacketRequest* reqP, SSL* sslP, User* userP);
void list(PacketRequest* reqP, SSL* sslP, User* userP);
void login(PacketRequest* reqP, SSL* sslP, User* userP);
void reg(PacketRequest* reqP, SSL* sslP);
void topup(PacketRequest *reqP, SSL *sslP, User *userP);
void gift(PacketRequest *reqP, SSL *sslP, User *userP);
void mailbox(PacketRequest *reqP, SSL *sslP, User *userP);
bool checkCookie(Cookie* cookieP, User* userP);
void addUser(char* account, size_t account_len, char* password, size_t password_len, size_t port, User*
userP);
size_t get_balance(char* account);
void set_balance(char *account, size_t balance);
bool is_account_valid(char *account);
```

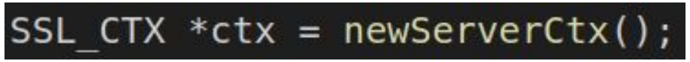
所實做各項功能

threadpool

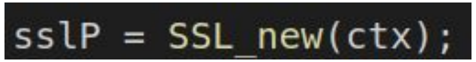


```
threadpool_t *pool = newThreadPool();
```

security connection

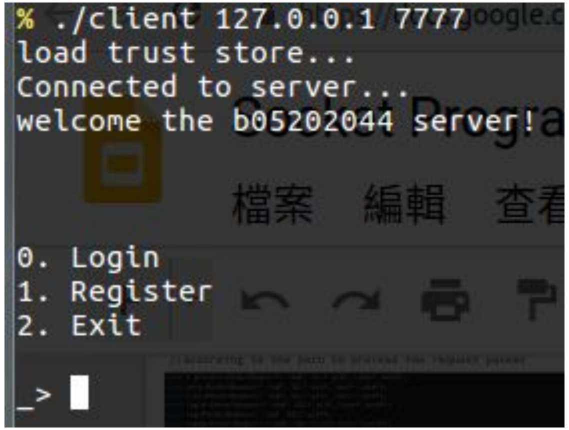


```
SSL_CTX *ctx = newServerCtx();
```



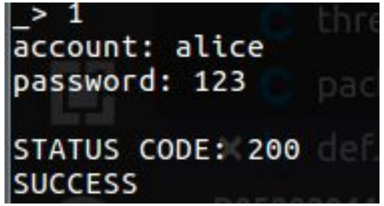
```
sslP = SSL_new(ctx);
```

(before login) login/register/exit

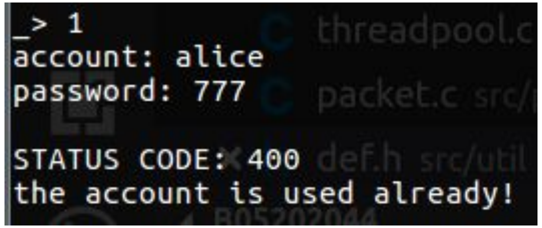


```
% ./client 127.0.0.1 7777
load trust store...
Connected to server...
welcome the b05202044 server!


0. Login
1. Register
2. Exit

_> 
```

register

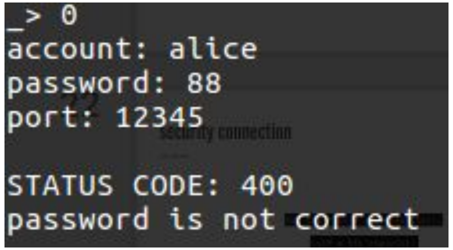


```
_> 1
account: alice
password: 123

STATUS CODE: 200
SUCCESS
```

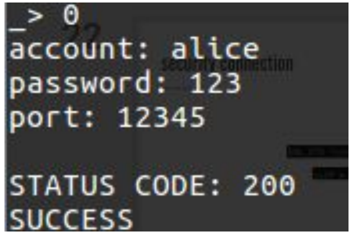


```
_> 1
account: alice
password: 777

STATUS CODE: 400
the account is used already!
```

login

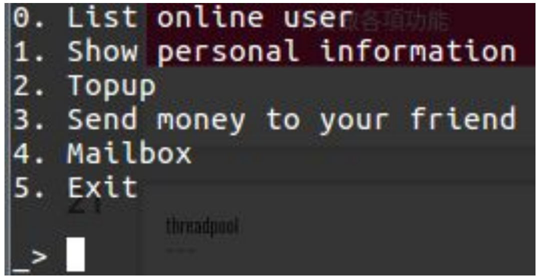


```
_> 0
account: alice
password: 88
port: 12345

STATUS CODE: 400
password is not correct
```



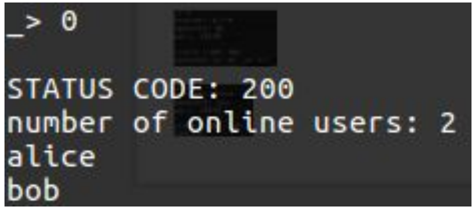
```
_> 0
account: alice
password: 123
port: 12345

STATUS CODE: 200
SUCCESS
```

(after login) list/show/topup(儲值)/gift/exit



```
0. List online user
1. Show personal information
2. Topup
3. Send money to your friend
4. Mailbox
5. Exit

_> 
```

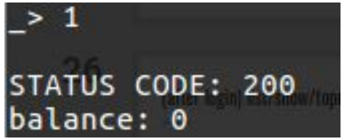
list



```
_> 0

STATUS CODE: 200
number of online users: 2
alice
bob
```

show

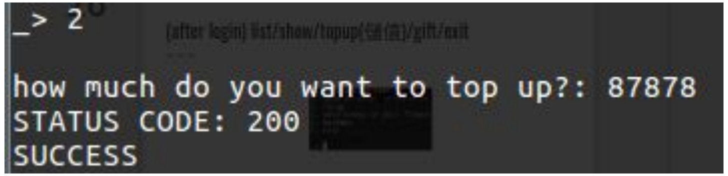
Now, Alice’s balance is 0.



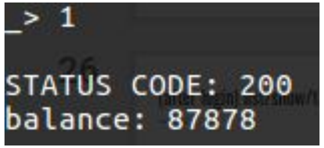
```
_> 1

STATUS CODE: 200
balance: 0
```

topup

After Alice topup 87878, Alice’balance is 87878.



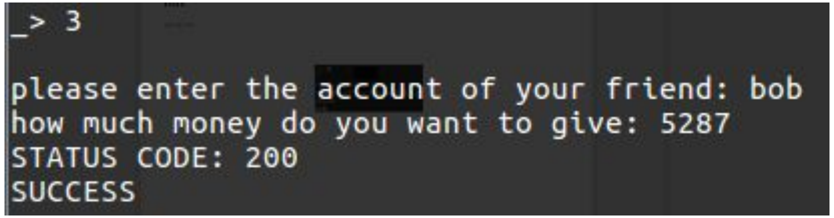
```
_> 2

how much do you want to top up?: 87878
STATUS CODE: 200
SUCCESS
```



```
_> 1

STATUS CODE: 200
balance: 87878
```

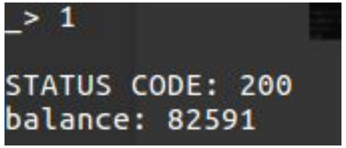
gift (send money to your friend)

Alice send 5287 to Bob.



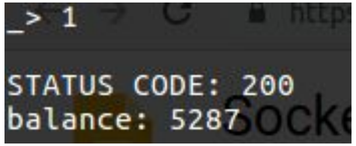
```
_> 3

please enter the account of your friend: bob
how much money do you want to give: 5287
STATUS CODE: 200
SUCCESS
```

Hence, Alice’balance is 82591.



```
_> 1

STATUS CODE: 200
balance: 82591
```

And, Bob’s balance become 5287 (原本是0)

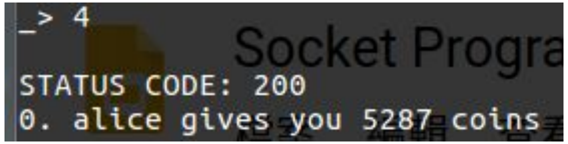


```
_> 1

STATUS CODE: 200
balance: 5287
```

mailbox

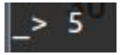
---

bob check his mailbox, then he can make sure 5287 is from alice



```
_> 4

STATUS CODE: 200
0. alice gives you 5287 coins
```

exit



```
_> 5
```

Bonus

interactive user interface/exception handling

如同前面截圖所示，我有提供interactive user interface給使用者使用，用以提示使用者如何操作以及提示錯誤訊息。

對於server端與client端，我都有做exception handling。

- 如果使用者輸入錯誤，使用者會得到清楚的錯誤訊息。
- server做了許多error handling
  - 包括防範他人傳送惡意 request
  - 使用者unexpected behavior(非預期離線、輸入過長訊息)
  - 當連線人數>可接受的最大上線人數 (可以更改server/def.h中的THREAD來測試)
  - 使用regex過濾各種injection
  - etc
- 因為error handling做在各種小地方上，因此不另外截圖，歡迎助教檢查。