

# Rapport Projet Réseau

## Messagerie en C



Polytech Montpellier IG4  
Chenyang Gao  
Marc Billy

# Partie I : Explications des fonctionnalités



Le côté client :

## Local

### **void show\_inbox(list\* mail\_list)**

La fonction permet d'afficher le contenu de la liste inbox, en montrant les numéros des mails, source, destination , objet, date et leurs status (0==pas lu, 1==lus)

### **void show\_sendbox(list \*mail\_list)**

La fonction permet d'afficher le contenu de la liste sendbox, en montrant les numéros des mails, source, destination , objet, date. Sendbox enregistre les mails qui ont été émit au serveur.

### **void show\_composebox(list\* mail\_list)**

La fonction permet d'afficher le contenu de la liste composebox, en montrant les numéros des mails, source, destination , objet, date. Composebox enregistre les brouillons de mails

### **list\* consult\_inbox(int i, list\* l)**

La fonction permet d'afficher le contenu d'un mail de la liste inbox (La source, destination, objet, date et message ). L'utilisateur donne le numéro qu'il veut consulter.

### **list\* consult\_sendbox(int i, list\* l)**

La fonction permet d'afficher le contenu d'un mail de la liste sendbox. (La source, destination, objet, date et message ). L'utilisateur donne le numéro qu'il veut consulter.

### **list\* delete\_mail(list\* l, int i)**

La fonction permet à l'utilisateur de supprimer un mail d'une liste (Inbox ou sendbox) en choisissant le numéro du mail et le nom de liste.

### **void compose()**

La fonction permet de composer un mail en fournissant le choix de sauvegarder soit dans composebox soit d'envoyer directement au serveur et puis de sauvegarder dans la liste sendbox

### **void quit()**

La fonction permet à l'utilisateur de se déconnecter.

### UDP:

#### **void subscribe()**

La fonction permet à l'utilisateur de faire son inscription sur le serveur.

#### **void unsubscribe()**

La fonction permet à l'utilisateur de se désinscrire sur le serveur.

#### **void getUsernameList()**

La fonction permet de demander au serveur de nous envoyer la liste des utilisateurs inscrits sur le serveur.

#### **void blacklist()**

La fonction permet à l'utilisateur de demander au serveur d'interdire à <pseu> de lui envoyer des mails.

### TCP:

#### **void connection\_tcp()**

La fonction permet à l'utilisateur de se connecter au serveur s'il est déjà inscrit, sinon la fonction lui demande de s'inscrire et de se connecter au serveur. Une variable CONNECTE est utilisée dans le programme client pour contrôler globalement le statut de la connection TCP avec le serveur.

#### **void sendMail(mail\* mail\_content)**

En état connecté, la fonction permet à l'utilisateur d'envoyer les mails en attente. En état déconnecté, la fonction demande à l'utilisateur de se connecter dans un premier temps.

#### **list\* receiveMailList(list\* list)**

En état connecté, la fonction permet à l'utilisateur de recevoir les nouveaux mails stockés sur le serveur. Il y a 2 sous-commandes: get <pseu> et check <pseu>. check <pseu> permet à l'utilisateur de savoir combien il y a de nouveaux mails pour lui sur le serveur. get <pseu> permet de retirer les mails et les mettre dans la liste INBOX locale. (l'utilisateur peut décider du nombre de nouveaux mails qu'il veut récupérer) Si l'utilisateur n'est pas connecté, la fonction demande à l'utilisateur de se connecter au serveur dans un premier temps.



### Le côté serveur :

### Local

#### **void purge(char \* str)**

Permet de remplacer le retour chariot d'une chaîne de caractère par le caractère '\0'. La fonction est utilisée pour les transactions UDP.

#### **int openUDP(int port, struct sockaddr\_in addr, int \* problem, int debug)**

Ouvre le serveur UDP sur le port 'port' avec pour paramètre 'addr'. S'il y a un problème, met 'problem' à 1. Affiche des informations de debuggage si 'debug' n'est pas à zéro. Retourne la socket du serveur.

**int openTCP(int port, struct sockaddr\_in addr, int \* problem, int debug, int maxUser)**

Ouvre le serveur TCP sur le port 'port' avec pour paramètre 'addr' avec 'maxUser' comme maximum d'utilisateur. S'il y a un problème, met 'problem' à 1. Affiche des informations de debugage si 'debug' n'est pas à zéro. Retourne la socket du serveur.

**user \* finduser(list \* userlist, char \* nick)** (récursive)

Recherche puis retourne l'utilisateur correspondant au pseudonyme 'nick' dans la liste 'userlist'

**list \* deleteuser(list \* userlist, char \* nick)** (récursive)

Supprime l'utilisateur correspondant au pseudo 'nick' dans 'userlist', retourne la liste modifiée.

**int isblack(list \* black, char \* nick)** (récursive)

Retourne 1 si le pseudonyme 'nick' est dans la blacklist 'black', 0 sinon.

**list \* deleteblack(list \* black, char \* nick)** (récursive)

Supprime le pseudo 'nick' de la blacklist passée en paramètre, retourne la blacklist modifiée.

**void getnicklist(list \* userlist, char \* nicklist)** (récursive)

Modifie la chaine de caractère 'nicklist' en y ajoutant tous les pseudonymes contenus dans la liste 'userlist'.

**void disconnect(user \* u, unsigned short \* nbUser)**

Déconnecte l'utilisateur 'u' s'il était connecté, sinon, ne fait rien.

**list \* processKeyboard (**

**list \* userlist, char \* command, int cmd\_sz,  
int udp, int \* sockUDP, struct sockaddr\_in \* addrUDP,  
int tcp, int \* sockTCP, struct sockaddr\_in \* addrTCP, unsigned short \* maxUser  
int debug, char \* retour, unsigned short \* nbUser, int \* ret\_sz, int \* problem)**

Traite une commande 'command' saisie au clavier. La fonction va modifier certains des paramètres envoyés si besoin est. Elle retourne la liste 'userlist' modifiée ou non.

**list \* processUDP (**

**int sockUDP, char \* command, int cmd\_sz,  
struct sockaddr \* exp, socklen\_t exp\_sz,  
list \* userlist, unsigned short \* nbUser, int \* problem)**

Traite une commande 'command' reçue sur la socket UDP. La fonction va modifier certains des paramètres envoyés si besoin est. Elle retourne la liste 'userlist' modifiée ou non.

**list \* processTCP (**

**list \* userlist, char \* command, int cmd\_sz,  
user \* u, unsigned short \* nbUser, int debug)**

Traite une commande 'command' reçue sur la socket TCP. La fonction va modifier certains des paramètres envoyés si besoin est. Elle retourne la liste 'userlist' modifiée ou non.



**En commun:**

**void get\_text\_from\_stdin(char \*dst, int sz, char \*mess)**

La fonction permet d'obtenir une chaîne de caractères saisie au clavier.

**get\_field\_by\_number(char \*a, char \*dst, int field, char del)**

La fonction permet d'obtenir le champ numéro i de la chaîne src formatée via le caractère del.

**mail\_to\_str(const mail \*m, char \*s)**

La fonction permet de convertir une structure mail en chaîne de caractères formatée.

**void str\_to\_mail(char \*s, mail \*m)**

La fonction permet de convertir une chaîne de caractères formatée en structure mail.

**list\* create\_list()**

La fonction permet la création d'une liste vide

**list\* append\_top(list \*l, void \*data)**

La fonction permet d'ajouter un élément dans la tête d'une liste.

**list\* get\_position(list \*l, int i)**

La fonction retourne le pointeur de la position i dans la liste l.

**list\* delete\_position(list\* l, int i)**

La fonction permet de supprimer un élément dans une liste et retourne la tête de la liste l.

**list\* clear(list \*l)**

La fonction vide la liste.

**int get\_size(list \*l)**

Retourne la taille de la liste.

**void strlwr(char \*str)**

Transforme tous les caractères de la chaîne de caractères en minuscule.

## Partie II :Un cas d'utilisation et sa démonstration



### Manuel de l'utilisateur :



#### le côté client

**CONNECTION:** Se connecter au serveur. (Mode TCP)

--Tapez "CONNECTION". Attendez que le serveur réponde. Ensuite tapez "connect <pseu> <password> ". Attendez que le serveur réponde. Si vous êtes bien connecté, vous pourrez ensuite utiliser les commandes "INBOX" et "COMPOSE" qui dépendent de la connexion TCP. Sinon, vous devrez premièrement vous inscrire au serveur. (Voir : "SUBSCRIBE")

**USERLIST:** Obtenir la liste d'utilisateur du serveur.

--Tapez "USERLIST". Attendez que le serveur vous réponde.

**INBOX:** Obtenir les nouveaux mails depuis le serveur et les afficher.

--Tapez "INBOX". (Si non connecté, voyez le guide "CONNECTION". ) Attendez que le serveur vous réponde. Il y a 2 choix : "get <pseu>" et "check <pseu>". Vous tapez "check <pseu>" dans le but de consulter le nombre de nouveau(x) message(s) sur le serveur. Vous tapez "get <pseu>" afin de récupérer les nouveaux messages depuis le serveur. Ensuite, vous tapez le nombre de nouveaux messages que vous voulez récupérer depuis le serveur dans INBOX local.

**SENDERBOX:** Afficher la liste des mails locaux que vous avez déjà envoyé au serveur.

--Tapez "SENDERBOX". Le programme vous affiche la suite.

**COMPOSEBOX:** Afficher les mails que l'utilisateur avait stocké dans composebox.

--Tapez "COMPOSEBOX". Le programme vous affiche la suite.

**COMPOSE:** Ecrire un nouveau mail puis l'envoyer.

--Tapez "COMPOSE". (Si non connecté, voyez le guide "CONNECTION". ) Tapez les informations qui vous sont demandées dans le but d'écrire au pseudo. Quant au choix, si vous voulez seulement stocker le mail sans l'envoyer, tapez "N". Sinon tapez "Y". Attendez que le serveur vous réponde. Vous pourrez ensuite taper "SENDERBOX" permettant de consulter les mails correctement envoyés.

**CONSULT:** Afficher le contenu d'un mail.

--Tapez "CONSULT <INBOX/SENDERBOX> <numéro>" pour obtenir le contenu d'un mail. (source, destination, objet, date et message). Par exemple, "CONSULT INBOX 3".

**DELETE:** Supprimer un mail dans une liste.

--Tapez "DELETE <INBOX/SENDERBOX> <numéro>" pour supprimer un mail. Par exemple, "DELETET INBOX 3".

**SUBSCRIBE:** Inscription au serveur.

--Tapez "SUBSCRIBE". Ensuite tapez : "subscribe <pseu> <password>". Attendez que le serveur vous réponde.

**UNSUBSCRIBE:** Désinscription du serveur.

--Tapez "UNSUBSCRIBE". Ensuite tapez : "unsubscribe <pseu> <password>". Attendez que le serveur vous réponde.

**BLACKLIST:** Blacklist une personne afin d'interdire cette personne de vous écrire.

--Tapez "BLACKLIST". Ensuite tapez "blacklist <pseu> <password> <unwanted>". Attendez le serveur vous réponde.

**QUIT:** Fermer le client.

--Tapez "QUIT".



**le côté serveur :**

**HELP:** Obtenir la liste des commandes disponibles.

--Tapez "HELP".

**WHOIS:** Obtenir des informations sur un utilisateur.

--Tapez "WHOIS <pseu>". Affiche le pseudo, le nombre de mail ainsi que l'IP de <pseu>.

**CLOSE/OPEN:** Fermer ou Ouvrir un serveur d'écoute.

--Tapez "CLOSE/OPEN <serv>". Ferme/Ouvre le serveur d'écoute <serv>, où <serv> est soit 'subscription', soit 'connection' correspondant respectivement à UDP et à TCP.

**MAXUSER:** Obtenir ou modifier le nombre maximum d'utilisateur.

--Tapez "MAXUSER <new>". Renvoi la nouvelle valeur du nombre maximum d'utilisateur. (Si <new> n'est pas donné ou est erroné, alors ne modifie pas la valeur)

**QUIT:** Fermer le serveur.

--Tapez "QUIT".

Dans la partie suivante nous vous présenterons une démarche normale, qui montre un utilisateur qui **se connecte** au serveur, qui **compose** un mail, qui **télécharge** ses mails, qui **consulte** son INBOX et qui finalement se **déconnecte**. (Note: l'utilisateur "gao" consulte son propre inbox, le mail qui est dedans est celui venant d'un autre utilisateur déjà enregistré sur le serveur).

Voici les différentes démarches du cas d'utilisation présenté ci-dessus :

Utilisateur	Programme Client	Programme Serveur
<b>Etape 1. CONNECTION</b>		
Taper "CONNECTION"	--> TCP : connection_tcp()-->	TCP : accept()
	--> TCP : connection_tcp()-->	TCP : processTCP():
Message: Error Wrong name/password	<-- TCP : connection_tcp()<--	TCP : close()
<b>Etape 2. SUBSCRIPTION</b>		
Taper "SUBSCRIPTION"	--> UDP:subscribe() -->	UDP : processUDP()
Message: Subscription Success	<-- UDP: subscribe() <--	UDP : sendto()
<b>Etape 3. CONNECTION</b>		
Taper "CONNECTION"	--> TCP: connection_tcp() -->	TCP : accept()
	--> TCP: connection_tcp() -->	TCP : processTCP()
Message: Now Connected!	<-- TCP: connection_tcp() <--	TCP: send()
<b>Etape 4. COMPOSE MAIL</b>		
Taper "COMPOSE" Saisir les informations demandées et choisir l'action d'envoyer la lettre	compose(), TCP : sendmail() -->	TCP : processTCP()
Message: Waiting/Mail successfully registered	<-- TCP : sendmail() <--	TCP : send()



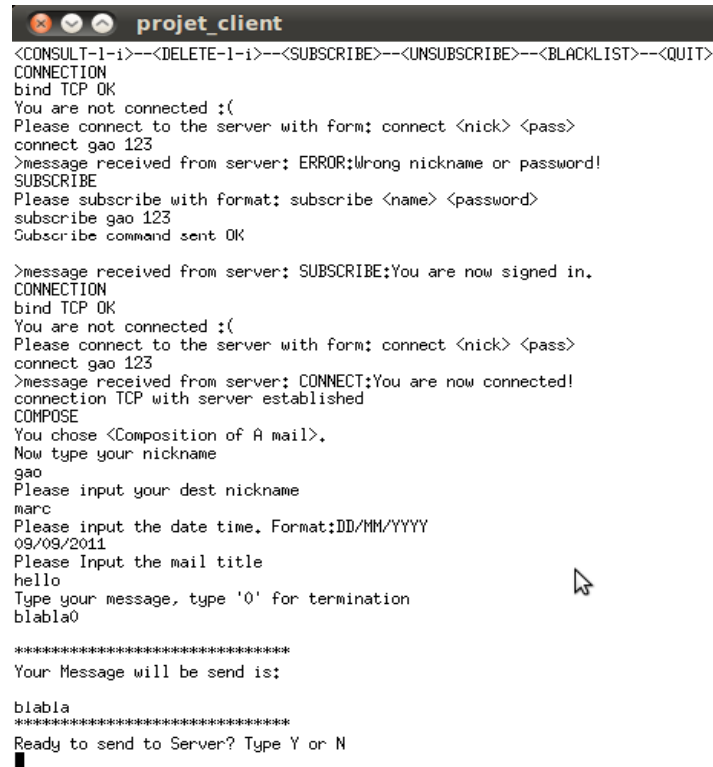
<b>Etape 5. GET MAIL</b>		
Taper "INBOX", choisir le nombre de nouveaux mails que l'utilisateur veut obtenir.	TCP: receiveMailList()-->	TCP : processTCP()
List: MailList	<--TCP:receiveMailList() show_inbox()	TCP : send()
<b>Etape 6: CONSULT INBOX</b>		
Taper "CONSULT INBOX 1"	-->consult_inbox()	N/A
Message: information sur mail 1	<--consult_inbox()	N/A
<b>Etape 7: DECONNECTION</b>		
Taper "QUIT"	close(sock_TCP), close(sock_UDP), quit()	TCP : main() Client TCP Transaction catch

## ANNEXE:

### Illustration par des images (screenshots) :

#### Côté client :

##### 1. CONNECTION, SUBSCRIBE, COMPOSE



```
<CONSULT-1-i>--<DELETE-1-i>--<SUBSCRIBE>--<UNSUBSCRIBE>--<BLACKLIST>--<QUIT>
CONNECTION
bind TCP OK
You are not connected :(
Please connect to the server with form: connect <nick> <pass>
connect gao 123
>message received from server: ERROR;Wrong nickname or password!
SUBSCRIBE
Please subscribe with format: subscribe <name> <password>
subscribe gao 123
Subscribe command sent OK

>message received from server: SUBSCRIBE:You are now signed in.
CONNECTION
bind TCP OK
You are not connected :(
Please connect to the server with form: connect <nick> <pass>
connect gao 123
>message received from server: CONNECT:You are now connected!
connection TCP with server established
COMPOSE
You chose <Composition of A mail>.
Now type your nickname
gao
Please input your dest nickname
marc
Please input the date time. Format:DD/MM/YYYY
09/09/2011
Please Input the mail title
hello
Type your message, type '0' for termination
blabla0

*****
Your Message will be send is:

blabla
*****
Ready to send to Server? Type Y or N
█
```

##### 2. SENDMAIL, INBOX, SENDBOX



```
hello
Type your message, type '0' for termination
blabla0

*****
Your Message will be send is:

blabla
*****
Ready to send to Server? Type Y or N
Y
SENDMAIL command sent correctly
>message received from server: MAIL:Waiting.
mail_str is :
mail|marc|gao|09/09/2011|hello|
blabla, and sizeof it is 500
>message received from server: MAIL:Mail successfully registered!
Your message has been stored in the Send box
You can type <SENDBOX> to check

Wrong command, try again
SENDBOX
*****LIST OF SENDBOX*****
[Number]:1 [Source]:gao [Destination]:marc
[Object]: hello [Date]:09/09/2011
*****
INBOX
You are connected :)
Get the maillist from server as the following format: get <nick> or check <nick>
get gao█
```

### 3. CONSULT INBOX, QUIT

```
INBOX
You are connected :)
Get the maillist from server as the following format: get <nick> or check <nick>
get gao
Input how many new messages you want to receive in INBOX
1
The mail in your inbox are:
*****LIST OF INBOX*****
[Number]:1 [Source]:marc [Destination]:gao
[Object]: Premier mail [Date]:24/03/2012 [Status]:0
[Number]:2 [Source]:test1 [Destination]:test1
[Object]: test1 [Date]:test1 [Status]:0
*****
Wrong command, try again
CONSULT INBOX 1
You have chosen <INBOX>, mail <1>
*****DETAIL OF MAIL*****
De : gao
A : marc
Date : 24/03/2012
Objet : Premier mail
Message :
Salut ! Comment vas-tu Chenyang ?
QUIT
```

---

## Côté serveur :

(En mode debug, permettant de voir l'activité du serveur)

Lancement ; Connexion ; Inscription ; Connexion, Inscription ;

Déconnexion d'un utilisateur non authentifié ; Connexion ;

Authentification de gao ; Envoi d'un mail de gao pour marc ;

Authentification de marc ; Envoi d'un mail de marc pour gao ;

Réception d'un mail pour gao ; Déconnexion de gao.

```
ubuntu@ubuntu:~/reseau$ ./serveur debug

Running Serveur's version: 1.0.0

Port TCP: 13284
Port UDP: 13285
Max user number: 10

Creating UDP socket:      OK
Running UDP server:      OK
Creating TCP socket:      OK
Running TCP server:      OK
Receiving TCP packet:     OK:NewUser 5
Receiving client TCP packet: RECV100:connect gao 123/SEND:ERROR:Wrong nickname or password!
Receiving UDP packet:     OK:subscribe gao 123
Receiving TCP packet:     OK:NewUser 5
Receiving UDP packet:     OK:subscribe marc pass
Receiving client TCP packet: Connection closed:TMP
Receiving TCP packet:     OK:NewUser 6
Receiving client TCP packet: RECV100:connect gao 123/SEND:CONNECT:You are now connected!
Receiving client TCP packet: RECV60:SENDMAIL gao marcMAIL:mail|marc|gao|09/09/2011|hello|
blabla/SEND:MAIL:Mail successfully registered!
Receiving client TCP packet: RECV40:connect marc pass/SEND:CONNECT:You are now connected!
Receiving client TCP packet: RECV256:sendmail marc gaoMAIL:mail|gao|marc|24/03/2012|Premier mail|Salut !
Comment vas-tu Chenyang ?/SEND:MAIL:Mail successfully registered!
Receiving client TCP packet: RECV60:get gao/SEND:mail|gao|marc|24/03/2012|Premier mail|Salut ! Comment va
s-tu Chenyang ?
Receiving client TCP packet: Connection closed:gao
```

```
ubuntu@ubuntu:~/reseaubis$ ./serveur

Running Serveur's version: 0.0.2

Port TCP: 13284
Port UDP: 13285
Max user number: 10

bindTCP(): Address already in use
>open connection
Connections are opened.
>whois gao
Unknown user...
>whois marc
User: marc
Mails: 0
IP: 81.220.155.151
>whois gao
User: gao
Mails: 0
IP: Not connected
>maxuser
You have to close connections before. 'close connection'
>close connection
Connections are closed.
>maxuser
Max user: 10
>maxuser 8
Max user: 8
>maxuser tsetdf
Max user: 8
>close subscription
Subscriptions are closed.
>quit
Closing server...
```

(en mode normal)

Lancement, problème TCP ;

Ouverture TCP ;

WHOIS gao ;

WHOIS marc ;

Enregistrement de gao ;

WHOIS gao ;

MAXUSER ;

MAXUSER 8 ;

MAXUSER erroné ;

Fermeture UDP ;

Fermeture serveur.