

Rapport de TP : Réseaux

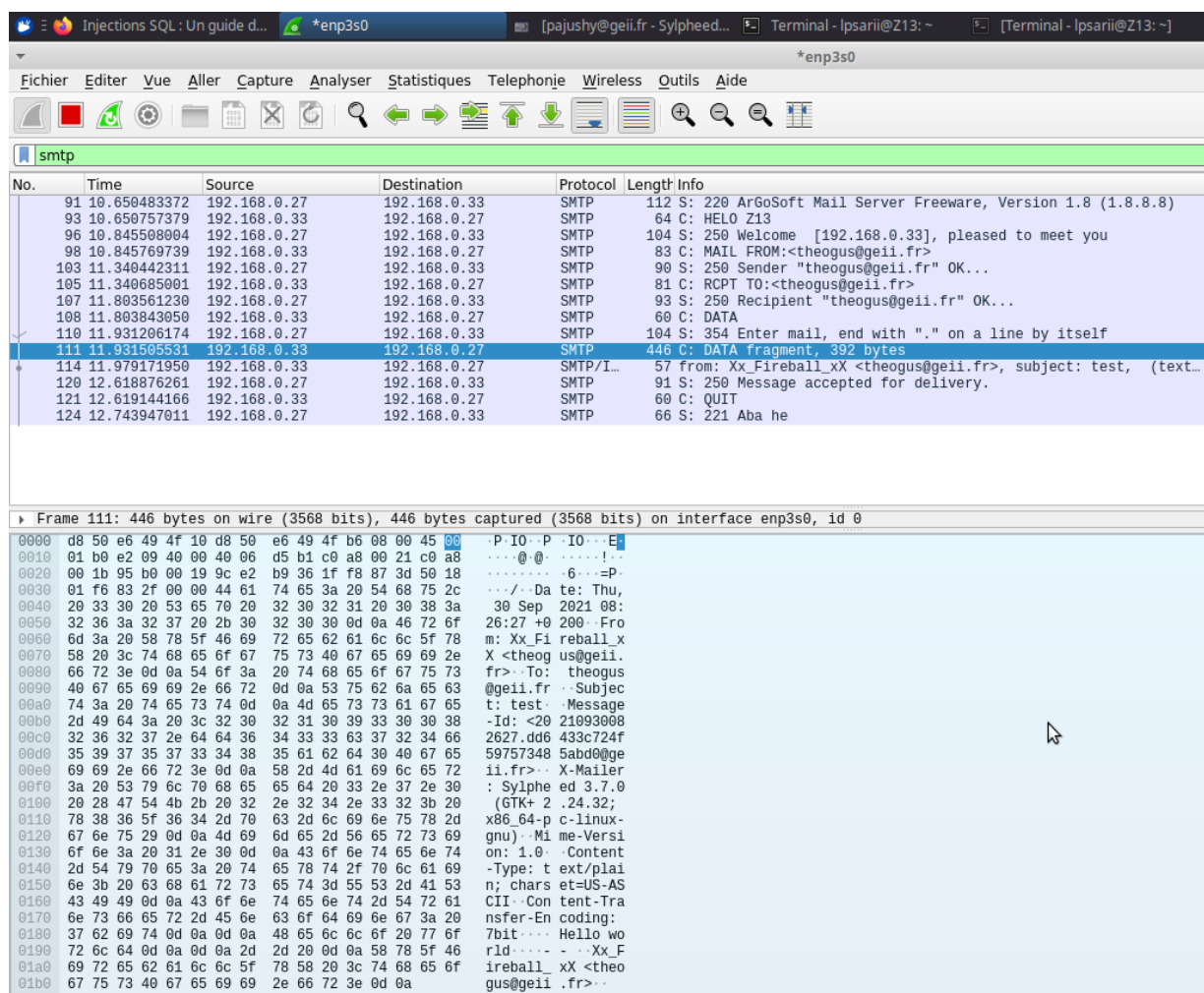
Création d'un client SMTP

Au cours de ce TP, nous avons étudié le fonctionnement du protocole SMTP, à l'utiliser via un client mail et telnet, puis à l'automatiser à l'aide d'un programme client en C.

Paramétrage et utilisation d'un client mail :

Pour commencer, nous allons utiliser un client mail classique, une « boîte mail ». Pour ce TP, nous utiliserons l'application sylpheed. En se rendant dans les paramètres, on peut configurer un nouveau compte sur cette machine, qui sera le compte theogus@gmail.com. On sélectionne le protocole POP3 (protocole permettant de récupérer des mails stockés sur un serveur), et on indique l'adresse du serveur mail, qui ici est 192.168.0.27.

On peut maintenant essayer d'envoyer notre premier mail. Pour envoyer un mail, une machine utilise le protocole SMTP (Simple Mail transfert Protocol), et le protocole POP3 (Post Office Protocol) pour les récupérer.



The screenshot displays the Wireshark interface with a packet capture on the 'smtp' filter. The packet list shows a sequence of SMTP transactions. The selected packet (No. 111) is an SMTP DATA packet, which is expanded in the packet details pane to show the email structure. The email header includes 'From: Xx_Fireball_xX <theogus@geii.fr>', 'Subject: test', and 'To: theogus@geii.fr'. The body of the email is 'Hello world'.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|--------------|--------------|-----------|--------|---|
| 91 | 10.650483372 | 192.168.0.27 | 192.168.0.33 | SMTP | 112 | S: 220 ArGoSoft Mail Server Freeware, Version 1.8 (1.8.8.8) |
| 93 | 10.650757379 | 192.168.0.33 | 192.168.0.27 | SMTP | 64 | C: HELO Z13 |
| 96 | 10.845508004 | 192.168.0.27 | 192.168.0.33 | SMTP | 104 | S: 250 Welcome [192.168.0.33], pleased to meet you |
| 98 | 10.845769739 | 192.168.0.33 | 192.168.0.27 | SMTP | 83 | C: MAIL FROM:<theogus@geii.fr> |
| 103 | 11.340442311 | 192.168.0.27 | 192.168.0.33 | SMTP | 90 | S: 250 Sender "theogus@geii.fr" OK... |
| 105 | 11.340685001 | 192.168.0.33 | 192.168.0.27 | SMTP | 81 | C: RCPT TO:<theogus@geii.fr> |
| 107 | 11.803561230 | 192.168.0.27 | 192.168.0.33 | SMTP | 93 | S: 250 Recipient "theogus@geii.fr" OK... |
| 108 | 11.803843050 | 192.168.0.33 | 192.168.0.27 | SMTP | 60 | C: DATA |
| 110 | 11.931206174 | 192.168.0.27 | 192.168.0.33 | SMTP | 104 | S: 354 Enter mail, end with "." on a line by itself |
| 111 | 11.931505531 | 192.168.0.33 | 192.168.0.27 | SMTP | 446 | C: DATA fragment, 392 bytes |
| 114 | 11.979171950 | 192.168.0.33 | 192.168.0.27 | SMTP/I... | 57 | from: Xx_Fireball_xX <theogus@geii.fr>, subject: test, (text... |
| 120 | 12.618876261 | 192.168.0.27 | 192.168.0.33 | SMTP | 91 | S: 250 Message accepted for delivery. |
| 121 | 12.619144166 | 192.168.0.27 | 192.168.0.33 | SMTP | 60 | C: QUIT |
| 124 | 12.743947011 | 192.168.0.27 | 192.168.0.33 | SMTP | 66 | S: 221 Aba he |

Frame 111: 446 bytes on wire (3568 bits), 446 bytes captured (3568 bits) on interface enp3s0, id 0

```
0000 d8 50 e6 49 4f 10 d8 50 e6 49 4f b6 08 00 45 00 .P.I0..P.I0...E
0010 01 b0 e2 09 40 00 40 06 d5 b1 c0 a8 00 21 c0 a8 ....@..
0020 00 1b 95 b0 00 19 9c e2 b9 36 1f f8 87 3d 50 18 ....6...=P
0030 01 f6 83 2f 00 00 44 61 74 65 3a 20 54 68 75 2c .../..Da te: Thu,
0040 20 33 30 20 53 65 70 20 32 30 32 31 20 30 38 3a 30 Sep 2021 08:
0050 32 36 3a 32 37 20 2b 30 32 30 30 0d 0a 46 72 6f 26:27 +0 200 -Fro
0060 6d 3a 20 58 78 5f 46 69 72 65 62 61 6c 6c 5f 78 m: Xx_Fi reball_x
0070 58 20 3c 74 68 65 6f 67 75 73 40 67 65 69 69 2e X <theog us@geii.
0080 66 72 3e 0d 0a 54 6f 3a 20 74 68 65 6f 67 75 73 fr>..To: theogus
0090 40 67 65 69 69 2e 66 72 0d 0a 53 75 62 6a 65 63 @geii.fr ..Subjec
00a0 74 3a 20 74 65 73 74 0d 0a 4d 65 73 73 61 67 65 t: test..Message
00b0 2d 49 64 3a 20 3c 32 30 32 31 30 39 33 30 30 38 -Id: <20 21093008
00c0 32 36 32 37 2e 64 64 36 34 33 33 63 37 32 34 66 2627.dd6 433c724f
00d0 35 39 37 35 37 33 34 38 35 61 62 64 30 40 67 65 59757348 5abdd0ge
00e0 69 69 2e 66 72 3e 0d 0a 58 2d 4d 61 69 6c 65 72 ii.fr>.. X-Mailer
00f0 3a 20 53 79 6c 70 68 65 65 64 20 33 2e 37 2e 30 : Sylphe ed 3.7.0
0100 20 28 47 54 4b 2b 20 32 2e 32 34 2e 33 32 3b 20 (GTK+ 2 .24.32;
0110 78 38 36 5f 36 34 2d 70 63 2d 6c 69 6e 75 78 2d x86_64-p c-linux-
0120 67 6e 75 29 0d 0a 4d 69 6d 65 2d 56 65 72 73 69 gnu)..Mi me-Versi
0130 6f 6e 3a 20 31 2e 30 0d 0a 43 6f 6e 74 65 6e 74 on: 1.0..Content
0140 2d 54 79 70 65 3a 20 74 65 78 74 2f 70 6c 61 69 -Type: t ext/plai
0150 6e 3b 20 63 68 61 72 73 65 74 3d 55 53 2d 41 53 n; chars et=US-AS
0160 43 49 49 0d 0a 43 6f 6e 74 65 6e 74 2d 54 72 61 CII..Con tent-Tra
0170 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e 67 3a 20 nsfer-En coding:
0180 37 62 69 74 0d 0a 0d 0a 48 65 6c 6c 6f 20 77 6f 7bit.... Hello wo
0190 72 6c 64 0d 0a 0d 0a 2d 2d 20 0d 0a 58 78 5f 46 rld.... -..Xx_F
01a0 69 72 65 62 61 6c 6c 5f 78 58 20 3c 74 68 65 6f ireball_xX <theo
01b0 67 75 73 40 67 65 69 69 2e 66 72 3e 0d 0a gus@geii .fr>..
```

Par défaut, le protocole SMTP n'est pas chiffré, ce qui permet de voir les données transférées en clair.

Dans le client mail, on peut maintenant cliquer sur le bouton « relever » pour récupérer le mail sur le serveur mail. Comme on utilise le protocole POP3, on active le filtre pop sur wireshark :

| Time | Source | Destination | Protocol | Length | Info |
|------|--------------|--------------|--------------|---------|---|
| 166 | 25.422951272 | 192.168.0.27 | 192.168.0.33 | POP | 112 S: +OK ArGoSoft Mail Server Freeware, Version 1.8 (1.8.8.8) |
| 168 | 25.424266631 | 192.168.0.33 | 192.168.0.27 | POP | 68 C: USER theogus |
| 170 | 25.465779163 | 192.168.0.27 | 192.168.0.33 | POP | 89 S: +OK Password required for theogus |
| 172 | 25.466107109 | 192.168.0.33 | 192.168.0.27 | POP | 68 C: PASS theogus |
| 175 | 25.515726651 | 192.168.0.27 | 192.168.0.33 | POP | 84 S: +OK Mailbox locked and ready |
| 177 | 25.515995481 | 192.168.0.33 | 192.168.0.27 | POP | 60 C: STAT |
| 179 | 25.561741073 | 192.168.0.27 | 192.168.0.33 | POP | 66 S: +OK 3 1621 |
| 180 | 25.561991818 | 192.168.0.33 | 192.168.0.27 | POP | 60 C: UIDL |
| 184 | 25.609742774 | 192.168.0.27 | 192.168.0.33 | POP | 60 S: +OK |
| 188 | 25.651870120 | 192.168.0.33 | 192.168.0.27 | POP/IMF | 117 1 scbxeka0676uu6vp , 2 zji8ub11exscjfp , 3 ia4u1wk1f34bw4c... |
| 190 | 25.652068340 | 192.168.0.27 | 192.168.0.33 | POP | 60 C: LIST |
| 191 | 25.688679797 | 192.168.0.33 | 192.168.0.27 | POP | 60 S: +OK |
| 193 | 25.704732483 | 192.168.0.27 | 192.168.0.33 | POP/IMF | 75 1 555 , 2 533 , 3 533 |
| 195 | 25.720715454 | 192.168.0.33 | 192.168.0.27 | POP/IMF | 60 . |
| 197 | 25.720951774 | 192.168.0.27 | 192.168.0.33 | POP | 62 C: RETR 3 |
| 200 | 25.768642650 | 192.168.0.33 | 192.168.0.27 | POP | 70 S: +OK 522 octets |
| 202 | 25.784961194 | 192.168.0.27 | 192.168.0.33 | POP/IMF | 587 from: Xx_Fireball_xX <theogus@geii.fr>, subject: test, (text... |
| 204 | 25.800658587 | 192.168.0.33 | 192.168.0.27 | POP/IMF | 60 . |
| 206 | 25.801128350 | 192.168.0.27 | 192.168.0.33 | POP | 60 C: QUIT |
| 208 | 25.847641207 | 192.168.0.33 | 192.168.0.27 | POP | 66 S: +OK Aba he |

On retrouve bien notre message

Reproduire une session SMTP avec le client telnet :

Maintenant que nous avons vu comment le client mail communique avec le serveur, nous allons essayer de le faire manuellement avec le client telnet. Ce client permet en effet de se connecter à un serveur et de lui envoyer des commandes. On se connecte donc au serveur mail avec la commande *telnet adresse_ip port* :

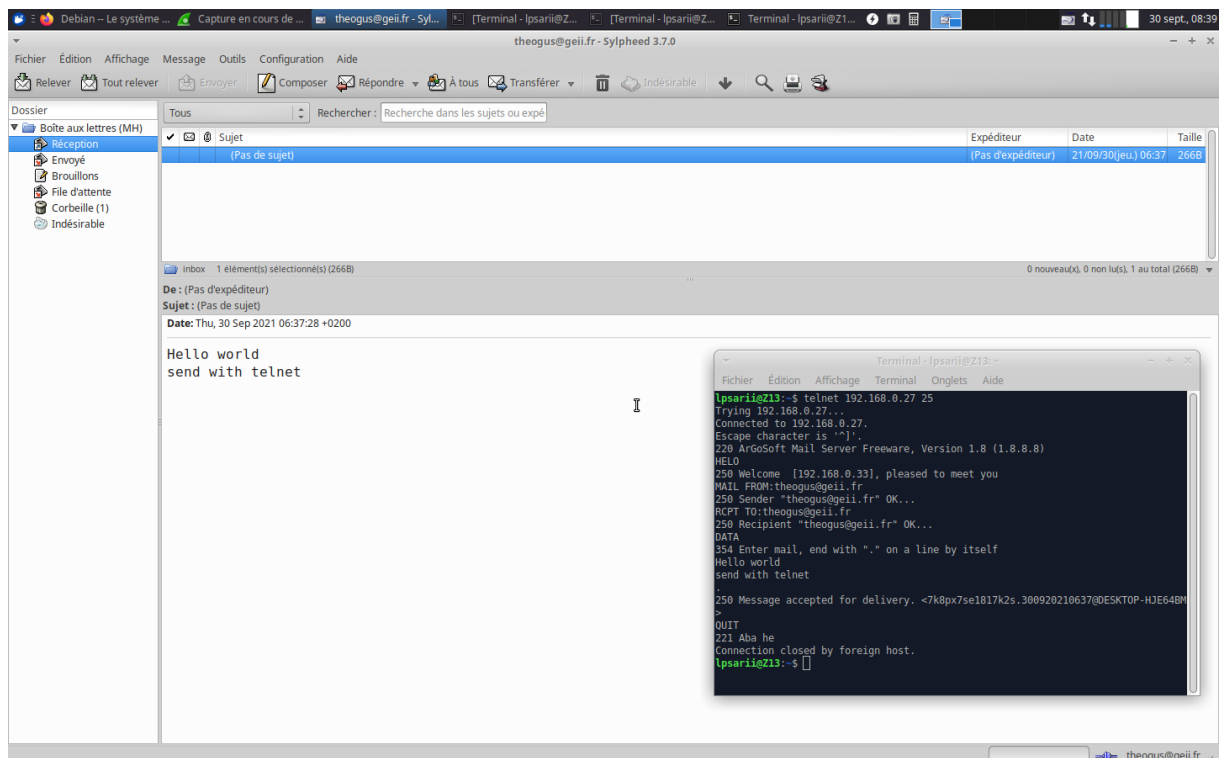
```
lpsarii@Z13:~$ telnet 192.168.0.27 25
Trying 192.168.0.27...
Connected to 192.168.0.27.
Escape character is '^]'.
220 ArGoSoft Mail Server Freeware, Version 1.8 (1.8.8.8)
HELO
250 Welcome [192.168.0.33], pleased to meet you
MAIL FROM:theogus@geii.fr
250 Sender "theogus@geii.fr" OK...
RCPT TO:theogus@geii.fr
250 Recipient "theogus@geii.fr" OK...
DATA
354 Enter mail, end with "." on a line by itself
Hello world
send with telnet
.
250 Message accepted for delivery. <7k8px7se1817k2s.300920210637@DESKTOP-HJE64BM>
>
QUIT
221 Aba he
Connection closed by foreign host.
lpsarii@Z13:~$
```

Il suffit alors de taper les commandes vues plus tôt pour envoyer notre message. Et sur Wireshark, on observera le même échange de trames entre le serveur et notre machine :

| Time | Source | Destination | Protocol | Length | Info |
|------|--------------|--------------|-----------|--------|--|
| 85 | 9.675108655 | 192.168.0.27 | SMTP | 112 | S: 220 ArGoSoft Mail Server Freeware, Version 1.8 (1.8.8.8) |
| 110 | 12.882099526 | 192.168.0.33 | SMTP | 60 | C: HELO |
| 114 | 12.990994885 | 192.168.0.27 | SMTP | 104 | S: 250 Welcome [192.168.0.33], pleased to meet you |
| 187 | 26.986122981 | 192.168.0.33 | SMTP | 81 | C: MAIL FROM:theogus@geii.fr |
| 198 | 27.355263872 | 192.168.0.27 | SMTP | 90 | S: 250 Sender "theogus@geii.fr" OK... |
| 288 | 40.506102177 | 192.168.0.33 | SMTP | 79 | C: RCPT TO:theogus@geii.fr |
| 291 | 40.918207929 | 192.168.0.27 | SMTP | 93 | S: 250 Recipient "theogus@geii.fr" OK... |
| 319 | 45.058081930 | 192.168.0.33 | SMTP | 60 | C: DATA |
| 321 | 45.172427094 | 192.168.0.27 | SMTP | 104 | S: 354 Enter mail, end with "." on a line by itself |
| 393 | 57.570106318 | 192.168.0.33 | SMTP | 67 | C: DATA fragment, 13 bytes |
| 426 | 63.338119381 | 192.168.0.27 | SMTP | 72 | C: DATA fragment, 18 bytes |
| 436 | 64.978110862 | 192.168.0.33 | SMTP/I... | 57 | Hello world , send with telnet |
| 437 | 65.003555564 | 192.168.0.27 | SMTP | 137 | S: 250 Message accepted for delivery. <7k8px7se1817k2s.300920... |
| 459 | 67.434180409 | 192.168.0.33 | SMTP | 60 | C: QUIT |
| 461 | 67.536971413 | 192.168.0.27 | SMTP | 66 | S: 221 Aba he |

Ici, nous avons envoyé un message de 2 ligne. On observe donc 2 fois la trame avec le flag « DATA ».

Et si on récupère les mails sur le client mail, on retrouve bien notre mail :



Que ce soit avec wireshark ou avec telnet, on observe bien un « dialogue » entre la machine et le serveur, puisque pour chaque commande envoyé, le serveur réponds avec un message de confirmation (ou d'erreur si le cas se présente).

Automatisation de l'envoi des messages :

Pour automatiser l'envoi des messages, on réutilise le code du dernier TP permettant de se connecter à un serveur.

Après la connexion au serveur, nous allons devoir envoyer les commandes une par une au serveur. Pour ce faire, on les prépare à l'avance dans des chaines de caracteres :

```
char* hello="HELO\r\n";
char* from="MAIL FROM:theogus@geii.fr\r\n";
char* to="RCPT TO:theogus@geii.fr\r\n";
char* DATA="DATA\r\n";
char* text="Hello world, send with C langage\r\n";
char* end=".\r\n";
char* quit="QUIT\r\n";
```

On retrouve ici les commandes vu précédemment.

Pour chaque envois de message, on va procéder de la façon suivante : la fonction `sendData()` renvoie -1 si une erreur se produit, ce qui permet de faire le test après chaque envois. S'il n'y a pas d'erreur, alors on récupère la réponse du serveur avec `receveData()`, qui elle aussi renvoie -1 en cas d'erreur, ou sinon la taille des données reçus, ce qui nous permet d'afficher la réponse du serveur. On obtient le bloc suivant que l'on utilisera pour chaque commande :

```
int sD=sendData(s,helo);
if(sD==-1)
{
    perror("error sending Data : HELO");
    return -1;
}
rD=receveData(s,data,255);
if(rD==-1)
{
    perror("error receiving Data : HELO");
    return -1;
}
printResponse(data,rD);
```

On utilise aussi la fonction `printResponse()`, prenant en argument le tableau contenant la réponse du serveur, et la taille de celui (taille renvoyé par la fonction `receveData()`).

```
void printResponse(char *data, int s)
{
    for(int i=0;i<s;i++)
    {
        printf("%c", data[i]);
    }
    printf("\n");
}
```

On répète donc le processus pour chaque commande :

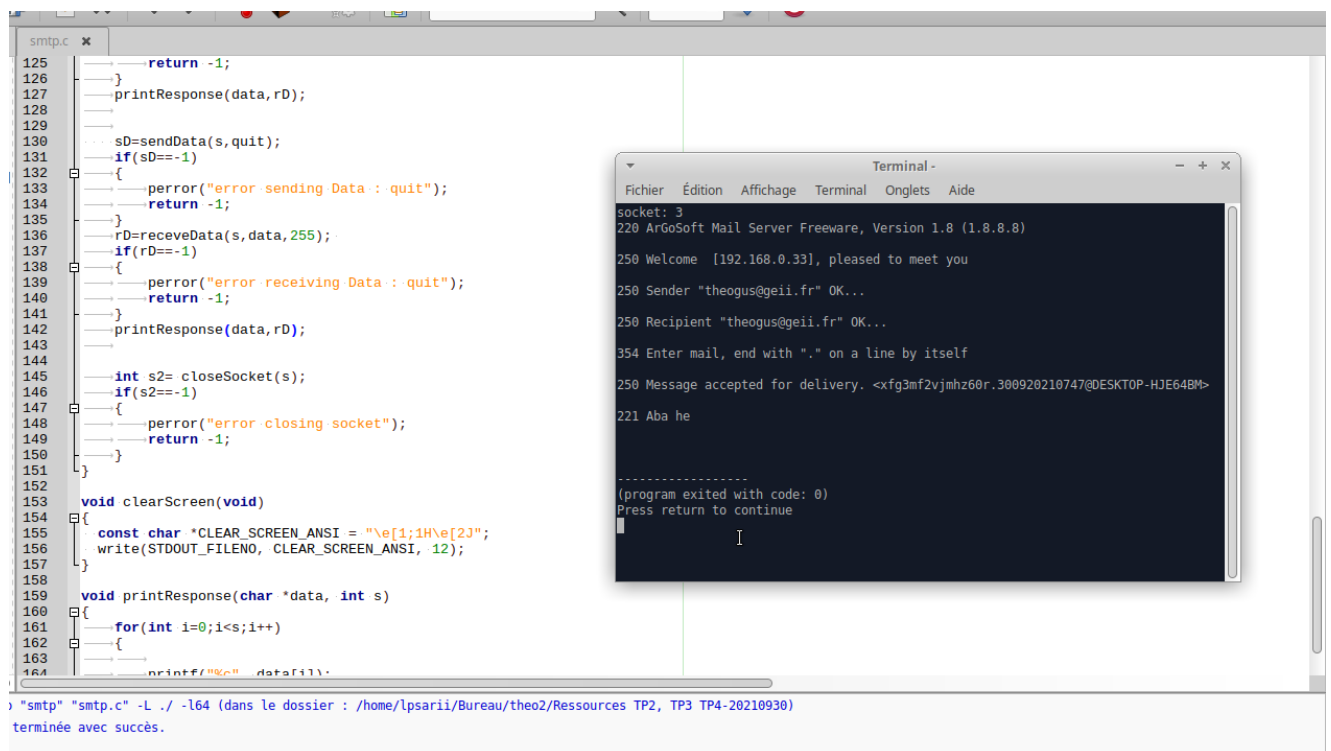
```
sD=sendData(s,from);
if(sD==-1)
{
    perror("error sending Data : FROM");
    return -1;
}
rD=receveData(s,data,255);
if(rD==-1)
{
    perror("error receiving Data : FROM");
    return -1;
}
printResponse(data,rD);
```

```

sD=sendData(s,to);
if(sD==-1)
{
    perror("error sending Data : TO");
    return -1;
}
rD=receiveData(s,data,255);
if(rD==-1)
{
    perror("error receiving Data : TO");
    return -1;
}
printResponse(data,rD);
////////////////////////////////////
sD=sendData(s,DATA);
if(sD==-1)
{
    perror("error sending Data : DATA");
    return -1;
}
rD=receiveData(s,data,255);
if(rD==-1)
{
    perror("error receiving Data : DATA");
    return -1;
}
printResponse(data,rD);
////////////////////////////////////
sD=sendData(s,text);
if(sD==-1)
{
    perror("error sending Data : text");
    return -1;
}
////////////////////////////////////
sD=sendData(s,end);
if(sD==-1)
{
    perror("error sending Data : .");
    return -1;
}
rD=receiveData(s,data,255);
if(rD==-1)
{
    perror("error receiving Data : .");
    return -1;
}
printResponse(data,rD);
////////////////////////////////////
sD=sendData(s,quit);
if(sD==-1)
{
    perror("error sending Data : quit");
    return -1;
}
rD=receiveData(s,data,255);
if(rD==-1)
{
    perror("error receiving Data : quit");
    return -1;
}
printResponse(data,rD);

```


Enfin, lors de l'exécution du programme, on retrouve les mêmes réponses vues avec telnet :



The image shows a code editor with a C program named `smtp.c` and a terminal window displaying the output of the program. The code in `smtp.c` includes functions for sending and receiving data, error handling, and printing responses. The terminal window shows the following output:

```
socket: 3
220 ArGoSoft Mail Server Freeware, Version 1.8 (1.8.8.8)

250 Welcome [192.168.0.33], pleased to meet you

250 Sender "theogus@geii.fr" OK...

250 Recipient "theogus@geii.fr" OK...

354 Enter mail, end with "." on a line by itself

250 Message accepted for delivery. <xfg3mf2vjmh260r.300920210747@DESKTOP-HJE64BM>

221 Aba he

-----
(program exited with code: 0)
Press return to continue
```

Le code complet : <https://github.com/IUT-Theophile-Wemaere/TP-reseau/blob/main/smtp.c>

Il est aussi possible d'utiliser le format MIME (Multi Internet Mail Extension) pour ajouter des données à notre mail, tel que l'expéditeur, le destinataire, le sujet du mail et bien d'autres. Comme c'est un format universel de codage de données pour les mails, tous les clients mails savent le décoder et l'afficher proprement.

Pour l'utiliser, il suffit de rajouter dans la chaîne de caractères « text » les éléments voulus :

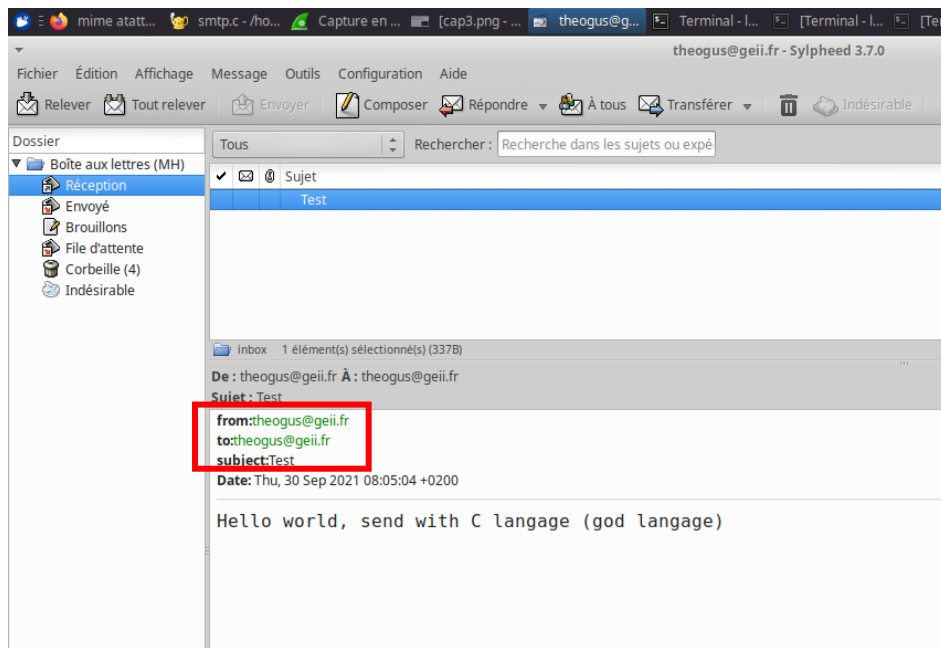
- _ « From : » => spécifie l'expéditeur
- _ « To : » => spécifie le destinataire
- _ « Subject : » => spécifie le sujet du mail

Il est aussi possible de spécifier les emails d'autres destinataires, la date, l'organisation, ajouter des pièces jointes,...

On obtient une chaîne de caractères ressemblant à ça :

```
char* text="From:theogusgmail.com\n\rTo:theogusgmail.com\n\rsubject:Test\r\n\r\nHello world, send with C language (god language)\r\n";
```

Ainsi, lors de la réception du message sur le client mail, toutes ces données s'afficheront :



La blague de la semaine :

CAPTURING PACKETS WITH WIRESHARK

The Global Internet Is Being Attacked by Sharks, Google Confirms

By WILL OREMUS

AUG 15, 2014 • 3:23 PM



YOU'RE TAKING IT WAY TOO LITERALLY

imgflip.com