TP - ASR5 système d'exploitation

Lecture sur une socket

5 mars 2017

Ce TP est noté, vous devez rendre votre travail à la fin de la séance. Il peut se faire en binôme (veillez bien à ce que votre binôme soit connu de votre chargé de TP, aucun binôme non déclaré ne sera formé après aujourd'hui).

Le but final du TP est de faire un programme C++ qui s'envoie lui même par mail à votre chargé de TP, vous mettrez au point ce programme en utilisant un script de test, puis un serveur jouet et *lorsqu'il sera au point* vous utiliserez l'adresse de votre chargé de TP ainsi que l'un des serveurs officiels de l'université.

Selon mon estimation, le sujet est trop long pour 2h. Ne vous inquétez pas, la note tiendra compte de l'endroit où vous êtes arrivé à la fin de la séance (notamment les résultats du script de test). Suivez les indications de ce script et à la fin de la séance, déposez votre travail dans la case tomuss `TP5Note`. *Le programme fourni doit au minimum compiler.*

I Envoi de mail

Le protocole SMTP (simple mail transfert protocole) est un protocole qui date de 1982 et (comme son nom l'indique) il permet l'envoi de mails. Il sert à transmettre un message à un serveur ou entre les serveurs.

I.1 Description des échanges

Le client et le serveur de mail échangent des données via des messages sous forme de textes formés de une ou plusieurs lignes séparées par des fins de lignes \r\n (ou endl en C++). Le client envoie des ordres sur une ligne formée de la commande elle même qui est un mot de quatre lettres majuscules (au début de la ligne) et de ses paramètres. La seule exception est la commande `DATA` qui signale que la suite sera le contenu du mail. À chaque commande le serveur renvoie une ou plusieurs lignes de réponses. Chaque ligne contient un code sur 3 chiffres suivi d'un tiret ou d'une espace puis d'une description. La dernière ligne de la réponse est la seule ligne dont les 4 premiers caractères sont 3 chiffres et une espace cela permet de la reconnaitre.

Les commandes les plus utilisées sont :

HELO *domaine*	permet au client de s'identifier en fournissant un nom de domaine (cela doit être la première
MAIL FROM : *adresse*	signale l'adresse mail de l'expéditeur du message qui va suivre
RCPT TO : *adresse*	signale le destinataire du message qui va suivre
DATA	début des données du mail
HELP	demande au serveur de lister les commandes disponibles
QUIT	demande au serveur de se déconnecter

À chaque commande la réponse accompagnée d'un code à 3 chiffres. Ce code, s'il commence par 2, signifie que la commande est acceptée et réussie. S'il commence par 3, cela signale que le serveur attend d'autres données. Les codes commençant par 4 et 5 sont des codes d'erreur, 4 signale une erreur temporaire (surcharge du serveur, nombre de connexions limitées ...). 5 au début du code signale une erreur définitive (mail refusé, destinataire inconnu, ...).

I.2 Échanges

Supposons un utilisateur gaston@spirou.be qui envoie un message à un ami jule@schmitt.ef. Le client de Gaston contacte son serveur sortant. Vous pouvez voir le chronogramme des échanges à la figure 1. Le serveur parle en premier, à chaque envoie de données (sauf lors de l'envoie du mail, il répond par un

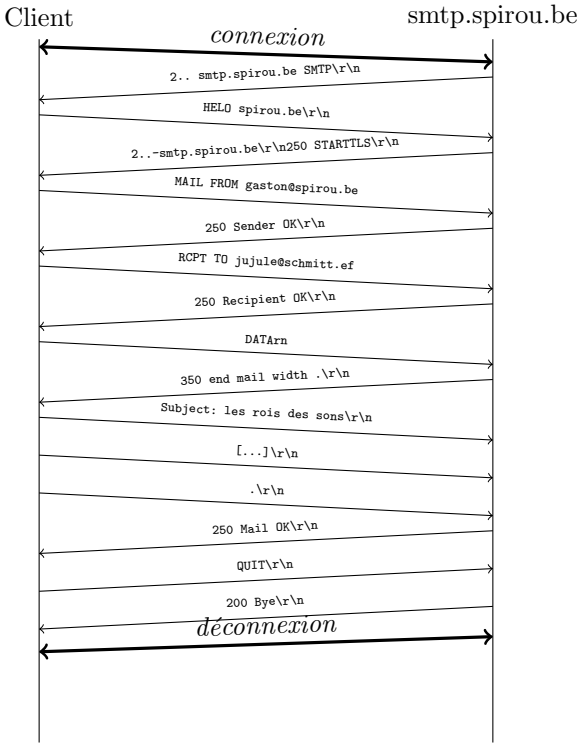

Figure 1 – Échange entre un client et un serveur mail

message sur une ou plusieurs lignes. La dernière ligne du message contient un code (erreur ou acceptation) et un petit message descriptif.

I.3 Composition d'un mail

Un mail est formé par ce qui est envoyé après la commande `DATA`. Il est composé d'un entête et d'un corps de mail. L'entête est une série de lignes sous la forme `Champ: Valeur\r\n` par exemple :

```
From: tragicomix@thapsus.tu
To: falbala@gallorum-oppidum.ga
Subject: Vale
```

Enfin le corps du mail est un texte qui se termine par une ligne contenant seulement un point : `\r\n.\r\n`

I.4 Travail à faire

Je rappelle qu'un serveur mails répond par une ou plusieurs lignes dont la dernière est celle qui commence par 3 chiffres suivi d'une espace.

Q.I.1) - Vous devez faire une fonction capable de lire la réponse d'un serveur mail :

```
void lire_rep(socklib::BufferedReaderWriter &serv, string &reponse, string &code, string &message)
```

Cette fonction doit lire la totalité de la réponse, retrouver la dernière ligne et la décomposer. Les 3 paramètres `reponse`, `code` et `message` sont les résultats de la fonction. Après son exécution, `reponse` doit contenir la totalité du texte répondu par le serveur, `code` doit contenir le code de 3 chiffres et `message` le petit message explicatif à la fin de la dernière ligne.

Q.I.2) - Vous devez faire un programme qui prend 4 paramètres (dans l'ordre) :
- l'adresse mail de l'envoyeur ;
- l'adresse mail du destinataire.
- un nom de serveur mail ;
- son port ;

Le programme doit vérifier le nombre de paramètres. Si exactement 4 parmètres ne sont pas fournis, il doit utiliser utiliser la fonction `usage` pour afficher une erreur (puis sortir). Cela doit vous permettre de valider le premier test.

Q.I.3) - Modifiez le programme pour qu'il contacte le serveur désigné par les 2 derniers paramètres. Cela doit permettre de valider le 2ème test.

Q.I.4) - Si vous lisez le message envoyé par le serveur à la connexion et que vous l'afficher en utilisant la fonction `affiche_reponse`, vous devez valider le 3ème test.

Q.I.5) - Vous devez faire en sorte que le programme suive le protocole pour envoyer un mail, c'est à dire envoyer
- `HELO` *nom du serveur*
- `MAIL FROM:` *mail envoyeur*
- `RCPT TO:` *mail destinrataire*
- `DATA`
- *texte du mail*
- `.`
- `QUIT`

à chaque envoie, le client doit lire la réponse du serveur, Pour que le script de test accepte le travail, vous devez absolument utiliser la fonction `affiche_reponse` pour afficher le retour du serveur. De plus vous devez tester la réponse du serveur et sortir en cas de refu.

Q.I.6) - Sur le serveur jouet `systeme.univ-lyon1.fr` via le port `8080` vous devez faire en sorte que le programme envoie un message avec un sujet, un champ `from`, un champ `to` et le message :

```
Bonjour,
voila le code de mon travail
##DEBUT CODE#############
...
##FIN CODE#############
```

Ce serveur accepte uniquement les mails à destination du domaine `asr5.univ-lyon1.fr` et pour les utilisateur `p1...` (votre login). Vous devez donc envoyer des mails à l'adresse `p...@asr5.univ-lyon1.fr`. Vous pourrez regarder les mails sur le webmail de ce serveur `http://systeme.univ-lyon1.fr` (votre login et le mot de passe `plop`).

Q.I.7) - Vous devez faire en sorte que le programme envoie son propre code dans le corps du message. Pour cette dernière étape, utiliser le serveur mail de l'université `smtp.univ-lyon1.fr:25`, votre propre mail dans le champs `From` et celui de votre chargé de TP :
- fabien.rico@univ-lyon1.fr
- adil.khalfa@cc.in2p3.fr
- yves.caniou@univ-lyon1.fr
- jacques.bonneville@univ-lyon1.fr

N'oubliez pas de déposer votre travail sur tomuss !.

II Script de test

Un script de test est à votre disposition dans l'archive : `test.py`. Il se lance avec la commande :

```
python3 test.py NOMEXECUTABLE
```

Même s'il n'est pas parfait, il essaye d'exécuter le protocole pas à pas et de signaler les erreurs. La notation en tiendra compte, il est donc important de l'essayer régulièrement.