TD 1 - LIF12 système d'exploitation

Passage d'informations sur les pipes

13 mars 2017

I Protocole SMTP

Le protocole SMTP (simple mail transfert protocole) est un protocole qui date de 1982 et qui permet l'envoie de mails. Il sert à transmettre un mail à un serveur qui va se charger de l'acheminer (via le même protocole) ou le stocker si c'est le serveur final. Au départ c'était le seul protocole utilisé pour la gestion des mails. Les logiciels de mail contactaient directement les serveurs et les utilisateurs se connectaient sur le serveur final pour lire directement les fichiers contenant les mails.

I.1 Description des échanges

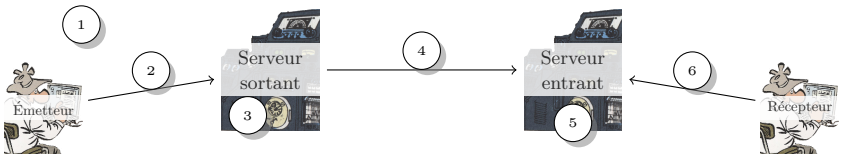
Le client et le serveur de mail échangent des données via des messages sous forme de textes formés de une ou plusieurs lignes séparées par des fins de lignes \r\n (ou endl en C++). Le client envoie des commandes sur une ligne formée de la commande qui est un mot de quatre lettres majuscules (au début de la ligne) et de ses paramêtres. La seule exception est la commande `DATA` qui signale que la suite sera le contenu du mail. A 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'un espace puis d'une description. La dernière ligne de la réponse est la seule ligne dont les 4 premier caractères sont 3 chiffres et un espace.

Les commandes les plus utilisées sont :

HELO *domaine*	permet au client de s'identifier en fournissant un nom de domaine (cela doit être la prem
EHLO *domaine*	peut remplacer la précedante et signifie qu'on souhaite utiliser un jeu de commandes éte
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 une aue 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 Parcours d'un mail



1. L'utilisateur écrit un mail.
2. Il est envoyé au serveur via SMTP.
3. Ce dernier le stocke dans sa liste de messages sortants.

4. Le serveur l'envoie au serveur distant lorsqu'il est disponible VIA smtp.
5. Qui le délivre dans la boite de l'utilisateur.
6. Lorsque le destinataire lit ses nouveaux mails, son agent contacte le serveur pour obtenir le mail.

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
```

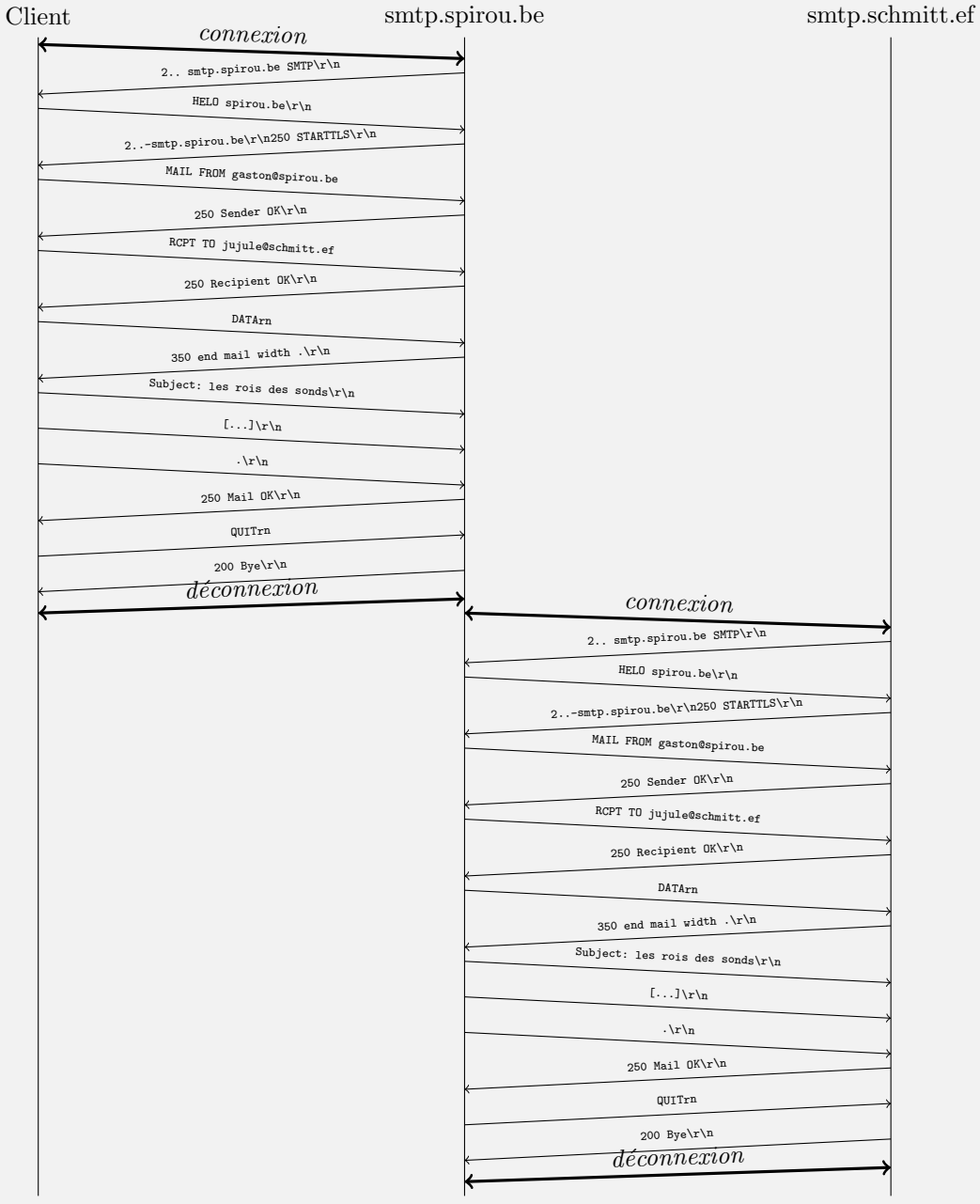
Enfin le corps du mail est un texte qui se termine par une ligne contenant seulement un point : \r\n.\r\n

I.4 Échanges

Supposons un utilisateur gaston@spirou.be qui envoie un message à un ami jule@schmitt.ef. Les deux serveurs de mails utilisés sont `smtp.spirou.be` et `smtp.schmitt.ef`.

Q.I.1) - Dessinez le chronograme des échanges ocasionnés par un envoie de mail jusqu'à son arrivée sur le serveur du destinataire (point 1 à 4 du shéma). Détaillez chaque commande envoyée par le client mail, le serveur sortant et le serveur entrant. plop

Je n'ai pas trop décrit les réponses du serveur dans le sujet. La seule chose importante pour le moment est le premier caractère (le code d'erreur). Le reste n'aura pas beaucoup d'importance en projet, et s'il y en a, le texte explicatif est suffisant pour comprendre. Donc, il faut juste se souvenir que la réponse existe et que 2.. est le signe que tout se passe bien.



I.5 Algorithme

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

— Donnez l'algorithme permettant de lire la réponse d'un serveur.
— Donnez le code C/C++ qui fait cela et renvoie un entier : 0 si tout va bien, 1 s'il faut une suite à la commande et -1 en cas d'erreur.

Données : s : socket
début
  resultat → Chaine vide
  cont → Vrai
  tant que *cont* **faire**
    ligne → lire une ligne sur s
    ajoute ligne à la fin de resultat
    si *ligne commence par 3 chiffres et un espace* **alors**
      cont → Faux
fin

J'utilise les BufferedReaderWriter qui permettent entre autre de lire une ligne depuis le réseau en s'assurant de la lire complètement et seulement elle.
Pour ceux qui ne sont pas famillié avec les expressions régulières (`regex`, il est possible de tesgter si une ligne est la dernière autrement avec quelque `if`. Mais je conseille de vite maitriser ces outils qui existent dans tous les langages et sont très pratiques.

```
int lire_reponse(socklib::BufferedReaderWriter &sock, string &resultat) {
  // on annule le résultat
  resultat = "";
  string ligne;

  while (true) {
    ligne = sock.read_line();
    resultat += ligne;
    // test avec une regex si la ligne commence par 3 chiffre et un espace.
    if (regex_search(ligne, std::regex("^[0-9][0-9][0-9] "))) {
      break;
    }
  }
  if (ligne[0] == '2') {
    return 0;
  } else if (ligne[0] == '3') {
    return 1;
  } else {
    return -1;
  }
}
```

II Droits sous unix

Sur le système considéré, il y a trois utilisateurs :
— gmw qui fait partie du groupe utilisateurs ;
— asw qui fait partie des groupes utilisateurs et developpeurs ;
— scw qui ne fait partie d'aucun de ces deux groupes.

Q.II.1) - Représentez dans une matrice les possibilités d'accès des fichiers et répertoires suivants pour chaque utilisateur.

```
-rw-r-----. 1 gmw developpeurs     27 janv.  2 11:06 donnees.txt
-rw-r--r--. 1 gmw utilisateurs     24 janv.  2 10:46 PPP-Notes
-rwxr-sr-x. 1 asw developpeurs 152392 janv.  2 10:47 prog1
-rw-rw----. 1 asw utilisateurs 164488 janv.  2 10:55 project.t
-rw-r-----. 1 asw developpeurs 118581 janv.  2 10:49 splash.png
```

	donnees.txt	PPP-Notes	prog1	project.t	splash.png
gsw	rw	rw	rx	rw	-
asw	r	r	rwx	rw	rw
scw	-	r	rx	-	-

Q.II.2) - Que signifie le bit s du fichier prog1, à quoi cela peut-il servir ? donnez un exemple d'utilisation.

C'est le bit setgid : la commande est exécutée avec le gid du fichier et pas celui du lanceur. Cela permet d'acquérir temporairement les droits correspondants. Ici par exemple, quel que soit l'utilisateur lançant le programme prog1, prog1 pourra lire le fichier splash.png (pour afficher l'icone alors que ce fichier est protégé par exemple). La commande sudo, qui appartient à root, fonctionne selon un principe similaire, mais avec un bit setsuid.

Exemple :
$ whereis passwd
passwd: /usr/bin/passwd /bin/passwd /etc/passwd /usr/share/man/man5/passwd.5.bz2 /usr/share/man/man1/pas
$ l /usr/bin/passwd
lrwxrwxrwx 1 root root 11 21 oct. 2015 /usr/bin/passwd −> /bin/passwd
$ l /bin/passwd
−rws−−x−−x 1 root root 46984 21 oct. 2015 /bin/passwd
Qu'en est-il de sudo ?

Plus d'info :
`http://docs.oracle.com/cd/E19683-01/816-4883/secfile-69/index.html`

Q.II.3) - Expliquer comment partager localement des fichiers avec au moins une personne, sans qu'elle(s) ai(en)t accès à l'ensemble de vos fichiers stockés localement.

Question à laisser en suspens, pour pratique des étudiants.
Le répertoire doit être exécutable, pas readable, et les personnes doivent connaître les noms des fichiers. Peut-on mieux faire, avec les réponses ci-dessus ?