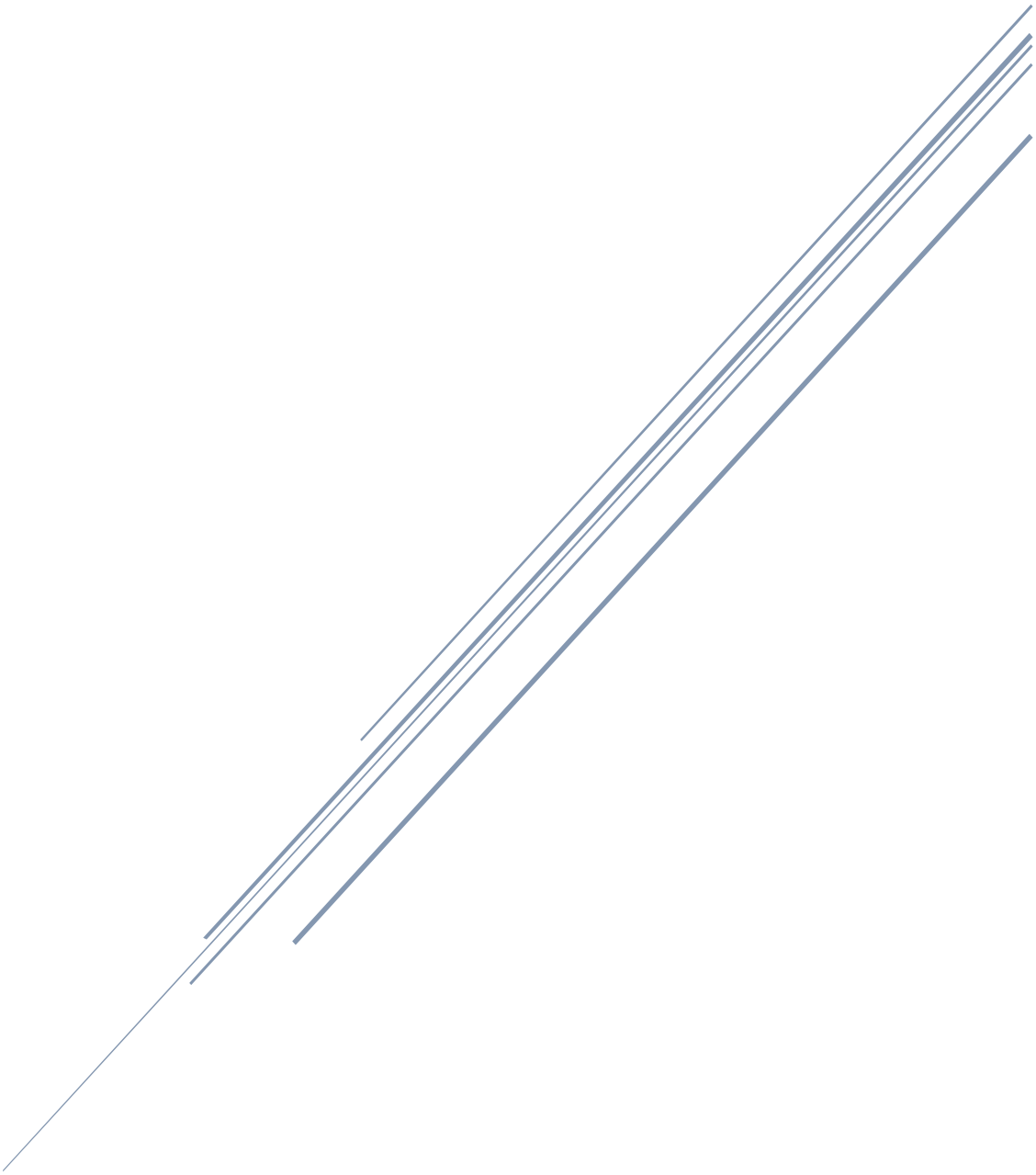
PROGRAMMATION DE SOCKETS

TP1 de Réseau avancé

Pierre Chevalier & Benoît Garçon



ISIMA
Réseau Avancé

Table des matières

Introduction

Le but de se TP est d'appréhender l'utilisation des sockets en C comme en Java au travers d'une petite application serveur-client pour exécuter des statistiques sur des chaînes de caractères simples.

I – Compilation

Pour compiler l'ensemble du projet (tous les différents programmes) il suffit de se placer à la racine du projet et d'utiliser la commande :

```
$ make
```

Il est possible de nettoyer l'ensemble du projet avec la commande :

```
$ make wipe
```

La compilation peut aussi être individualisée en utilisant en option le nom des programmes à compiler (client, serveur, serveurJava, clientSecu ou serveurSecu) :

```
$ make nomProgramme1 nomProgramme2 …
```

II – Exécution

A – Serveur C

Pour utiliser le serveur C il suffit de se placer à la racine du projet et d'exécuter le programme serveur avec le numéro de port d'écoute en argument et le nombre maximum de clients en option (le nombre de clients maximal par défaut est 1024) :

```
$ ./serveur <port> [NB_MAX_CLIENTS]
```

Ce serveur ne peut s'éteindre que lorsque l'extinction est demandé par le client grâce à la commande « . » et la connexion avec un client ne se termine que lorsque le client envoie (automatiquement) le caractère « / ».

B – Serveur Java

Pour utiliser le serveur Java il suffit de se placer à la racine du projet et d'exécuter le script bash serveurJava.sh avec le numéro de port d'écoute en argument (ce script appelle en fait la classe JServeur.class de classes/) :

```
$ ./serveurJava.sh <port>
```

Sur ce serveur c'est le serveur lui-même qui met fin à la communication avec le client en fermant la socket juste après avoir répondu au client.

C – Serveur sécurisé

Le serveur sécurisé s'utilise de la même manière que le serveur normal hormis le fait qu'il faille un certificat généré par le script init_ssl.sh :

```
$ ./init_ssl.sh
$ ./serveurSecu <port> [NB_MAX_CLIENTS]
```

D – Client C

Le client C utilise GET pour le nombre de caractères et POST pour la valeur des chaînes. Pour utiliser le client C il suffit de se placer à la racine du projet et d'exécuter le programme client d'une des façons suivantes :

1- Avec menu

Pour utiliser la version verbeuse il suffit d'ajouter l'option -v seule :

```
$ ./client -v
```

Le programme demande ensuite les différentes informations nécesaires.

2- En ligne de commande

Pour utiliser la version rapide en ligne de commande il suffit d'entrer toutes les informations nécessaires à la connexion suivies de la ou des commandes à exécuter :

```
$ ./client <adresse serveur> <port> [commande1 [commande2 […]]]
```

Une commande commence par un symbole désignant la commande à effectuer suivi de la chaîne à traiter. Si la chaîne contient des espaces, ne pas oublier de mettre la commande entre double ou simple quote. Voici la liste des commandes :

- + : nombre de consonnes
- – : nombre de voyelles
- ? : nombre de caractères
- = : valeur de la chaîne

Si la commande n'est pas mentionnée elle sera demandée par le programme.

3- En ligne de commande avec seulement des requêtes http

Pour utiliser la version rapide en ligne de commande sans avoir à mettre un nom de serveur et un port pour seulement faire des requêtes http il suffit de mettre l'option -o :

```
$ ./client -o [commande1 [commande2 […]]]
```

E – Client sécurisé

Tout comme le client simple, le client sécurisé peut être utilisé des trois façons différentes précédentes en utilisant cette fois la commande clientSecu :

```
$ ./clientSecu -v
$ ./clientSecu <adresse serveur> <port> [commande1 [commande2 […]]]
$ ./clientSecu -o [commande1 [commande2 […]]]
```

III – Exemple d'utilisation

```
$ ./serveur 1111
Socket created.
Waiting for clients...
Received localhost from 56010.
Client's request :
>>> +Bonjour les amis !

Client's request:
>>> /
Session ended.
```

```
$ ./client localhost 1103
Enter your command:
>>> +Bonjour les amis!
Connexion established.

Server's message:
>>> Number of consonants: 8
```

```
$ ./serveur 1111 10
Socket created.
Waiting for clients...

$ ./serveurJava.sh 1111
Waiting for clients...

$ ./serveurSecu 1111
Socket created.
Waiting for clients...
```

```
$ ./client -v

$ ./client localhost 1111 +br
"+blanc et bleu" =Rouge -
abcde ?cococococococococo

$ ./client -o "?Hello world"

$ ./clientSecu -v
```

IV – Avancement

Nous avons fait tout ce qui est demandé dans le TP, la question bonus inclue. On peut noter que le serveur en C est plus complexe que le serveur Java puisque dans le premier cas nous avons géré l'extinction du serveur et la possibilité de gérer un nombre maximum de clients pour éviter la surcharge.

Aucun problème ne semble se présenter.