ENSEIRB-MATMECA

PROGRAMMATION RESEAU : LES SOCKETS

Patrice KADIONIK
kadionik.enseirb-matmeca.fr

TABLE DES MATIERES

PENDANT LA SEANCE :

	On ne boit pas dans la salle. Les pauses sont faites pour cela.
	On ne mange pas dans la salle. Les pauses sont faites pour cela.
	On n'utilise pas son téléphone pour se distraire ou réaliser des activités personnelles. Les pauses sont faites pour cela.
	Soyez en forme
	N'oubliez pas que vous travaillez en binôme

09/09/2025	Révision 6.6	Olivier VILLAIN

1. BUT DES TRAVAUX PRATIQUES

Le but de ces TP est de maîtriser la programmation réseau par `socket` en langage C sous UNIX.

2. COMPTE RENDU / NOTATION

Vous remettrez par binôme :

- un compte rendu au format PDF (de 4 à 10 pages maxi) dans lequel vous expliquerez votre **méthode de travail** et les **tests** qui vous permettent de valider votre conception

- un zip avec les fichiers sources
- la méthode de notation tient particulièrement compte de la qualité des codes sources : lisibilité, justesse des commentaires…

3. UTILISATION DE L'INTELLIGENCE ARTIFICIELLE

L'utilisation de l'IA générative dans les buts suivants est PENALISEE :

- Générer ou corriger un rapport
- Générer du code source à partir d'un énoncé dans le but de résoudre un exercice
- Introduire dans l'IA votre propre code source
- Mettre en forme, rajouter/vérifier/corriger des commentaires
- Corriger un bug

L'utilisation de l'IA générative dans les buts suivants n'est pas PENALISEE :

- Répondre à une question particulière sur un point mal compris (une fonction, un paramètre, l'origine d'un dysfonctionnement …)

L'utilisation de Mistral AI sera préférée à ChatGpt

Vos codes sources et vos rapports seront automatiquement inspectés afin de vérifier que vous avez respecté ces règles

La pénalité est la note minimale.

4. QUELQUES OUTILS

man : https://linux.die.net/man

A tout moment on peut avoir des informations en ligne sur une commande via la commande *man*. Exemple :

```
% man  socket
```

netstat : https://linux.die.net/man/8/netstat

La commande **netstat** permet d'afficher l'activité réseau. Voici quelques usages :

- netstat –pl : affiche les Processus à l'écoute (Listening) (donc TCP)
- netstat –apt : affiche tous (All) les Processus avec un socket Tcp
- netstat –pu : affiche les Processus ayant ouvert un socket UDP

grep : https://linux.die.net/man/1/grep

La commande **grep** effectue une recherche sur la sortie standard. On l'utilisera avec la commande netstat dans l'exemple ci-dessous. Cette notation a pour effet de mettre entrée de la commande grep le résultat de la commande netstat :

- netstat –pl | grep 22222 : affiche toutes les lignes de netstat contenant le texte 22222

telnet : https://linux.die.net/man/1/telnet

La commande **telnet** est un client TCP pouvant se connecter à tout serveur TCP. Les paramètres sont le nom du serveur ou son adresse IP, et son port d'écoute. Par exemple :

- telnet brahmane.enseirb.fr 21 : se connecte au serveur FTP de l'Enseirb. Il comprend les requêtes : USER, PASS, HELP, QUIT …

- telnet 127.0.0.1 80 : se connecte à un serveur web local dans la machine

curl : https://linux.die.net/man/1/curl

La commande curl envoie des requêtes sur un serveur. On l'utilisera ci-dessous pour envoyer une requête sur un serveur web :

- curl –http1.0 https://www.qwant.com/?l=fr&q=helsinki&t=web

& : lancement de commande en parallèle

Le caractère « & » permet de lancer en parallèle plusieurs commandes. Par exemple :

- curl –http1.0 https://www.qwant.com/?q=helsinki & curl –http1.0 https://www.qwant.com/?q=oslo & …..

5. ENVIRONNEMENT DE TRAVAIL

Les stations Linux de l'école permettent le développement en C.

Les élèves qui apportent leur PC portable peuvent se connecter à leur console Linux par SSH : Sous Windows lancer « cmd », ou sous Linux lancer « terminal » :

6. COPIE DE FICHIER

La commande scp permet de copier des fichiers d'une machine à l'autre

- Scp fichier.c mon_login@ssh.enseirb.fr:/mon_repertoire

7. EDITEUR DE FICHIER SOURCE : VI

L'éditeur « vi » fourni par Linux permet l'édition de vos fichiers source. Quelques commandes utiles :

- « vi monsource.c » : édite le fichier

- « i » : entrer dans le mode édition
- « esc » : sortir du mode édition
- « :w » : sauvegarder le fichier
- « :q » : quitter vi
- « :wq » : sauvegarder et quitter

8. COMPILATION

Pour travailler de façon homogène, vous utiliserez la command ./cur pour compiler vos projets. Si votre fichier source s'appelle mon_serveur.c, vous lancerez la commande :

- ./cur mon_serveur

Si vous avez le message « permission denied » enexécutant cur, changez les droits d'exécution :

- chmod +x cur

Si la compilation réussi, le fichier exécutable « mon_serveur » sera généré.

9. RESSOURCES

Dans le répertoire : /net/npers/ovillain001/Public/RE223
Copiez les fichiers présents dans votre répertoire de travail :

- cp␣/net/npers/ovillain001/Public/RE223/*␣votre_répertoire

10. PROJET 1 : CLIENT/SERVEUR TCP

Chaque binôme réalisera une paire client/serveur TCP.
Un élève réalise le client et l'autre élève réalise le serveur.
Les 2 élèves travaillent en coopération pour que le projet final soit fonctionnel.

10.1. Spécifications du serveur

Le serveur écoute sur le premier port disponible à partir du numéro 22000 et en ordre croissant.

Le serveur sera une calculatrice capable de réaliser les opérations de base (addition, soustraction, multiplication, division), sur 2 opérandes.

Chaque opérande pourra être un nombre à virgule flottante dont on ne garde que 2 décimales après la virgule (réaliser un arrondi par défaut) compris entre 0 et 10000

Les résultats seront également des nombres à virgule flottante avec 2 décimales après la virgule (réaliser un arrondi par défaut)

Les opérateurs +,-,/,* seront autorisés. La virgule sera le point '.'

Les cas d'erreur seront gérés et remontés au client.

Le serveur sera capable de traiter 10 opérations simultanées en provenance de clients différents.

Pour simuler un traitement long, on pourra utiliser l'instruction sleep(n) ; qui attend n secondes. Cela permettra de lancer plusieurs clients simultanément.

10.2. Spécifications du client

Le processus client sera exécuté en ligne de commande depuis le terminal
On l'exécutera de la manière suivante :

```
$ ./clientCalc  Nom_serveur  Port_serveur
```

Le client attend les requêtes de l'utilisateur et affiche le résultat :

```
>> 31+7
=38
>> 10001*5
Out of range
>> 2.442-0.978
=1.47
```

Le client se termine avec la commande quit

```
>> quit
Bye
$
```

11. PROJET 2 : CLIENT/SERVEUR UDP

On réalise le même projet que précédemment avec un client et un serveur UDP