ENSSAT LANNION

UNIVERSITÉ DE RENNES 1

TP : Serveur HTTP haute disponibilité (4h)

LSI2 :RES

Version du : 2 décembre 2013

Objectifs *A l'issue de cette séance de TP, vous serez capable de*
- *Décrire le cycle de vie d'une socket TCP*
- *Implémenter ce cycle de vie en langage C*
- *Forger des réponses HTTP*
- *Interpréter des requettes HTTP*

Délivrables A l'issue des 4h de TP, vous enverrez un compte rendu individuel de ces séances de TP, accompagné de vos sources commentés à `barreaud@enssat.fr`. Vous y décrirez votre tâche, le degré d'avancement, les problèmes rencontrés et vos solutions. Vous justifierez vos choix d'implémentation. L'objectif de ce compte rendu est de faciliter la tâche d'un hypothétique LSI2 voulant refaire le TP en 2h au lieu de 4.

1 But du TP

1.1 Le cadre

L'archive donnée en annexe contient un squelette de serveur en langage C. Ce serveur gère un pool de sous serveurs HTTP ayant pour but de servir sous forme HTML des fichiers "texte" contenus dans un répertoire pré-établi.

1.2 Votre tâche

La gestion du pool de sous-serveur est déjà prise en compte. Ce qui vous reste à faire, c'est établir les connexions et gérer leurs cycle de vie.

2 Déroulement

2.1 La base

Vous n'interviendrez que dans le fichier `reseau.c` (commentaires `TODO`). A chaque fois, une seule ligne (ou deux) est à rajouter.

- TODO 1 : création de la prise (création de la socket). Il s'agit d'une socket TCP.
  Voir `http://pubs.opengroup.org/onlinepubs/000095399/functions/socket.html`
  Vous récupérerez le descripteur de la socket dans `fd` et testerez la bonne création (utiliser la commande FATAL en cas d'erreur).
- TODO 2 : associer un nom à la socket.
  Voir `http://pubs.opengroup.org/onlinepubs/000095399/functions/bind.html`
  et test de la bonne réallisation (FATAL)
- TODO 3 : ouverture du service avec un backlog de 4.
  Voir `http://pubs.opengroup.org/onlinepubs/000095399/functions/listen.html`

Pour tester votre serveur, vous pouvez utiliser un navigateur web (qui forgera pour vous vos requêtes GET), ou bien l'utilitaire en ligne

```
curl -X GET 'http://localhost:8000/index.txt'
```

Par la suite, construisez votre propre client HTTP en C en vous appuyant sur les mêmes mécanismes d'établissement de socket.

2.2 Pour aller plus loin

Pour le moment, seule la méthode `GET` est implémentée... étendez votre serveur aux autres méthodes en commençant par "PUT" (déposer un fichier).

```
curl -H 'Content-Type: text/html'  -X POST -d 'Voici le contenu du fichier test2.txt' 'http://localh
```

Autre fonctionnalité : automatiser la rédaction des entêtes des réponses avec le code retour et le content-type.