Université Paris Diderot – Paris 7 **Programmation réseau**
L3 Informatique **Anne 2013-2014, 2ème semestre**

TP n°9

HTTP

Exercice 1 [Un serveur web en ligne de commande]

1. Lancez la commande `nc` avec l'option `-l 8080`, cela ouvre un serveur sur votre machine locale sur le port 8080. Connectez-vous à ce serveur avec votre navigateur préféré. Tapez une ligne dans votre terminal. Que se passe-t-il ? Quittez votre serveur `nc`, que se passe-t-il alors dans votre navigateur ?
2. Dans un fichier `test.html` tapez un exemple de code HTML. Lancez de nouveau `nc` comme pour la question précédente en lui donnant en entrée (grâce à un pipe et à la commande `cat`) votre fichier html. Connectez vous de nouveau à votre serveur avec un navigateur web, que se passe-t-il ? Pouvez-vous rafraichir la page ?
3. De façon à pouvoir rafraichir la page depuis la navigateur, écrivez un script bash bouclant à l'infini et exécutant la commande de l'exercice précédent. Testez avec votre navigateur le résultat.

Exercice 2 [Un serveur simple]
Écrivez en Java un programme qui écoute sur le port 8080. Lorsqu'il a accepté une connexion, votre programme fermera immédiatement cette dernière. Testez votre serveur à l'aide d'un navigateur web et assurez-vous que votre programme est capable d'accepter plusieurs connections de suite.

Exercice 3 [Lecture de requêtes]
Modifiez votre programme pour que, lorsqu'il reçoit une connexion, il lise des données jusqu'à une ligne vide CR-LF-CR-LF. Une fois la ligne vide trouvée, il enverra une réponse de code 501, par exemple la suivante (où chaque ligne se termine par CR-LF) avant de fermé la connexion :

```
HTTP/1.0 501 Pas encore implemente
 Content-Type: text/html; charset=utf-8

 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
   Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <body>
  <p>Desole, je n'ai pas encore fini.</p>
</body>
</html>
```

Comme avant, testez votre programme.

Exercice 4 [Analyse de requêtes]
Modifiez votre programme pour qu'il analyse la première ligne de la requête du client. Celle-ci doit être composée de trois champs séparés par des espaces :

- le premier (le type de requête) est la chaîne de caractères «GET» ;
- le deuxième (le chemin) commence par le caractère «/» ;
- le troisième (le protocole) commence par «HTTP/1.».

Exemple : `GET /chemin/page.html HTTP/1.1`
Si les trois conditions ci-dessus sont remplies, votre serveur donnera une réponse de code 200 ; dans le cas contraire, il donnera une réponse de code 400.

Exercice 5 [Affichage de pages web locales] Votre serveur web sert toujours la même page. Mettez différentes pages dans un répertoire `~/public_html/`. Modifiez votre serveur pour que après avoir vérifié que le chemin de la requête commence par "/" et ne contient pas la chaîne "/../", il retourne au client le contenu du fichier correspondant situé dans `~/public_html/`. La valeur de l'en-tête `Content-Type` dépendra du nom du fichier, il vaudra :

- `text/plain; charset=utf-8` si le nom du fichier se termine par ".txt" ou ".text" ;
- `text/html; charset=utf-8` si le nom du fichier se termine par ".htm" ou ".html" ;
- `application/octet-stream` sinon.

Exercice 6 [Longueur de pages]
Modifiez votre programme pour qu'il indique au client la longueur du corps de la réponse dans un en-tête `Content-Length`.

Exercice 7 [Serveur web concurrent]
Si ce n'est pas déjà le cas, modifiez votre serveur web de façon à ce qu'il soit capable de servir plusieurs clients simultanément.
Vérifiez que plusieurs connections simultanées sont possibles.