TP4 - Le Protocole HTTP

Conseil avant de commencer : n'hésitez pas à installer l'extension Firefox "Live HTTP Headers".

1 Une première utilisation de telnet

Telnet est un protocole simple de connexion à distance : il permet de transmettre des caractères entre une machine locale (écran + clavier) et une machine distante [1].

Par défaut, un client telnet se connecte au service TCP *telnet* mais il est possible de préciser le service TCP de la façon suivante : `telnet M S`. Dans ce cas le client telnet se connecte au service S de la machine M.

Quel est l'effet de la commande suivante : `telnet time-a.timefreq.bldrdoc.gov daytime` ?

Décrire le protocole *daytime* d'après le résultat de cette commande.

2 Protocole HTTP

Pour le reste du TP, nous travaillerons sur le protocole HTTP. Quelle est le port utilisé par ce protocole ?

Nous utiliserons `telnet` pour envoyer directement des commandes aux serveurs « à la main ». N'utilisez donc pas simplement la commande `HEAD` depuis le shell, utilisez d'abord `telnet` pour vous connecter au port www avant d'envoyer du texte.

2.1 Méthode `HEAD`

Pour obtenir l'en-tête d'une page web sans pour autant la télécharger complètement, on peut utiliser simplement la méthode HEAD :

```
HEAD / HTTP/1.0
Host: www.le.domaine.voulu.com
```

N'oubliez pas de taper deux fois sur entrée pour que le serveur sache que vous avez fini votre requête

Observez les différences entre les en-têtes de la page de garde de www.emi.u-bordeaux1.fr, www.labri.fr, www.inria.fr, www.cnrs.fr ?

Pour chaque exemple, sur quelle machine est implanté le serveur et quel est le type de ce serveur ? Quelle est la classe de réponse ?

2.2 Classes de réponse

En utilisant la méthode `HEAD`, essayez d'obtenir les différentes classes de réponse.

— Succès. `/* facile */`

— Erreur client. `/* vraiment facile */`

— Inchangé. `/* plus dur */` [2]

— Redirection. `/* vous ne pouvez pas l'inventer */`.
Sous Firefox ouvrez l'URL `http://dept-info.labri.fr/~thibault/test`. D'après vous, que se passe-t-il ?

1. À essayer depuis chez vous : `telnet towel.blinkenlights.nl`
2. après le `HEAD` et l'en-tête `Host`, envoyez un en-tête du type `If-Modified-Since: Sat 05 Nov 2005 23:23:59 GMT`

2.3 Méthode `GET` simple et entêtes

Reprenez la question précédente, pour obtenir le contenu du document `/~thibault/test` sur le serveur `dept-info.labri.fr`. Obtenez vous la même chose que sous Firefox ?

3 Utilisation d'un proxy

Comme vous utilisez régulièrement Firefox comme navigateur, savez-vous si vous avez configuré ce dernier en connexion directe sur internet, ou bien en passant par un proxy ? Si vous ne le savez pas, essayez de trouver cette option de configuration sous votre navigateur.

Depuis l'université, vous pouvez utiliser le proxy `cache.u-bordeaux.fr` sur le port 3128. Avec *telnet*, récupérez la page d'accueil de `http://www.inria.fr` au travers du proxy. Il faut cette fois donner l'URL en entier à la méthode GET. Expliquer les différences entre une connexion avec et sans proxy. Quels sont les intérêts d'un proxy ?

Essayons d'utiliser un autre proxy : 140.77.1.58 (port 3128). Rechargez `http://www.inria.fr`. Qu'en concluez vous ?

Remarque : *après cet exercice, n'oubliez pas de revenir à la configuration initiale.*

4 Vos traces

La Commission Nationale de l'Informatique et des Libertés a été instituée par la loi n° 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés qui la qualifie d'"autorité administrative indépendante". Cette commission a bien entendu développé son propre serveur HTTP : `http://www.cnil.fr`. La CNIL met à disposition des pages d'informations sur les droits, les devoirs, et les risques encourus par les usagers sur internet, illustrés dans la rubrique « Vos traces » (Parmi les liens "Vos droits"), qui vous propose quelques expériences. Consultez celle-ci en mode accès direct puis en accès via le proxy de l'université, et comparez le résultat de la première expérience (le nom d'hôte, notamment).

Avec accès direct, votre machine a-t-elle été identifiée par le site de la CNIL ?

Même question que précédemment, mais en utilisant l'accès via votre proxy.

5 Cookie et formulaire

Pour aller à l'essentiel, un cookie est un enregistrement d'informations par le serveur sur l'ordinateur client (le vôtre), informations que ce même serveur peut aller relire et modifier ultérieurement. Plus précisément, un cookie se compose d'un ensemble de variables (ou de champs) que le client et le serveur s'échangent lors de transactions HTTP, lesquelles variables sont tout simplement stockées sur la machine cliente. Un cookie est obligatoirement rattaché à un nom de domaine et un ensemble d'URL de telle sorte que seule une requête provenant du même serveur pourra y accéder. Par exemple, grâce à un programme CGI, le serveur a la possibilité de mettre à jour ou d'effacer un cookie. Mais pour cela, il doit spécifier tous les attributs du cookie, par conséquent seul le serveur qui a créé un cookie peut le modifier ou le supprimer.

Sachez que leur fonctionnement est assez simple sous Firefox : les cookies sont stockés dans un fichier binaire qui est une base de données SQLite. Vous pouvez ouvrir ce fichier en utilisant `sqlitebrowser` depuis la ligne de commande.

Examinez votre fichier de cookies (`$HOME/.mozilla/firefox/xxx.default/cookies.sqlite`) Les tuples ont la structure suivante :

- **id** : clé primaire
- **name** : le nom laissé par le serveur qui a déposé le cookie
- **value** : ça parait évident !
- **host** : le nom du serveur qui a laissé ce cookie
- **path** : restriction sur le chemin d'accès du serveur
- **expiry** : date d'expiration du cookie en secondes depuis le 1/1/70
- **lastAccessed** : dernière utilisation en microsecondes depuis le 1/1/70

— **isSecure** : si égal à 1, le cookie ne peut transiter que *via* une connexion SSL
— **isHttpOnly** : si égal à 1, le cookie n'est pas lisible par un script côté client

Le formulaire `http://perso.aquilenet.fr/~samuel/cookies/` traite deux cookies « nom » et « fruit » et permet de les mettre à jour. Après avoir soumis le formulaire, constatez que son effet a bien été pris en compte dans la base de données des cookies de Firefox.

Après combien de temps les cookies expirent-ils ?

À la main via `telnet`, récupérez directement cette page en utilisant la méthode GET. Dans un premier temps passez-lui l'état de deux cookies « nom » et « fruit » en ajoutant par exemple dans l'en-tête `Cookie: nom=toto`.

Lorsque vous validez le formulaire dans le navigateur, observez l'URL obtenue : les champs du formulaires sont ajoutés à l'URL. Essayez de faire de même à la main avec `telnet` : il suffit d'ajouter juste derrière `/cookies/`

Maintenant, essayons avec la méthode POST. Dans l'en-tête, en plus de `Host:`, il faut fournir `Content-type: application/x-www-form-urlencoded` et `Content-Length: 19`, puis introduire une ligne vide avant d'envoyer les valeurs postées sous la forme `nom=test&fruit=test`. Il faudra bien sûr adapter 19 selon la taille des valeurs postées). Vérifiez que la réponse fournie par le serveur concorde avec ce qui est attendu.

6 Vraiment à la main

Écrivons en C un outil qui récupère la page d'accueil d'un domaine, que l'on utilisera ainsi : `httpget www.ledomaine.com`. Dans votre programme, vérifiez bien tous les codes de retour pour être sûr de remarquer les erreurs, ou bien utilisez `strace` pour bien vérifier ce qui se passe. N'effectuez **que** les *casts* indiqués ci-dessous. Si vous ressentez le besoin d'effectuer d'autres *casts* que ceux-là, c'est que vous faites fausse route ! Compilez avec `-Wall -g -O2` pour que le compilateur relève bien toutes les erreurs potentielles. Le plus efficace est donc de créer un petit `Makefile` contenant simplement :

```
CFLAGS=-Wall -g -O2
all: httpget
```

les règles implicites de `make` font le reste.

6.1 Allons-y !

Puisque l'on veut ouvrir une connection TCP sur le port 80, il faut donc dans l'ordre
— utiliser `gethostbyname()` pour traduire le nom de domaine (récupéré depuis `argv[1]`, donc) en adresse IP. Dans la réponse passée sous forme de structure `struct hostent`, vérifiez que le champ `h_addrtype` vaut bien `AF_INET`. Il suffit alors de *caster* le pointeur stocké dans le champ `h_addr` de la `struct hostent` en `struct in_addr *`. Vous pouvez utiliser `printf("%x\n"` pour afficher en hexadécimal le champ `s_addr` de la `struct in_addr` pour être sûr du résultat (testez avec `10.0.0.1` pour voir un cas simple).
— créer la socket à l'aide de la fonction `socket`, en type `SOCK_STREAM` puisque l'on veut du TCP, laisser le protocole à 0 pour que le système le choisisse automatiquement.
— préparer une variable de type `struct sockaddr_in` (documenté dans `man 7 ip`) : il suffit d'initialiser les trois champs `sin_family`, `sin_port` et `sin_addr`. Pour ce dernier il suffit donc de copier ce qui a été obtenu avec `gethostbyname`(). Pour le port, pensez bien à traduire depuis l'ordre des octets machine vers l'ordre des octets réseau à l'aide de `htons()`.
— utiliser `connect()` pour se connecter réellement. Ici, il faut *caster* l'adresse de la `struct sockaddr_in` préparée ci-dessus en `struct sockaddr *`.
— utiliser un simple write pour écrire la requête `GET /` Pensez bien à écrire deux retour chariot, pour introduire la ligne blanche.
— utiliser un simple read pour récupérer le résultat. Pensez bien à ajouter un `\0` à la fin avant de l'afficher à l'aide d'un simple `printf`.

6.2 Raffinements

Traitez correctement les cas d'erreur : essayez par exemple avec `localhost` (sur lequel aucun serveur web ne tourne) et avec `trucbidule` (qui n'existe pas).

Essayez d'utiliser `fdopen` pour utiliser `fprintf` plutôt que `write`. Est-ce que cela fonctionne ?

Souvenez-vous du cours de Programmation Système, il faut flusher le tampon de la libc à l'aide de `fflush`, sinon rien n'est écrit (vérifiez-le à l'aide de `strace`).