Magistère STIC — 2006–2007
Module *Réseaux* — Feuille de TP n° 6

Un vérificateur de liens

Dans ce TP nous allons réaliser un logiciel qui trouve dans un site web les liens « cassés ».

Les pages « web » sont écrit dans le langage HTML (Hypertext Markup Language). La seule chose à savoir sur HTML pour ce TP est la syntaxe des liens :

`<a href="`*url*`">`

En vérité, la balise `<a >` peut avoir des autres options mais celles-là ne nous intéressent pas ici. Les guillemets autour de l'*url* sont souvent omis.

Un *url* (pour Unique Resource Locator) indique la destination du lien. Dans une forme simplifiée, un tel url consiste en trois parties :
- le protocole (par exemple http, ftp)
- le nom du serveur
- le nom du fichier

Par exemple

`http://www.dptinfo.ens-cachan.fr/STIC/`

Pour ce TP, vous pouvez supposez que le nom du protocole est terminé par `:`, le nom du serveur est entre `//` et `/`, et que tout le nom du fichier est tout le reste qui commence sur le `/`.

On trouve sur des pages web aussi des *url relatives* : Il est possible d'omettre le protocole et le nom de serveur (dans ce cas on utilise le protocole et le nom du serveur avec lesquels on a accède la page actuelle), et dans ce cas il est de plus possible que le nom du fichier ne commence pas sur `/` (dans ce cas, c'est un nom de fichier dans le répertoire qui contient la page actuelle).

Exercice 1
Écrire une fonction qui extrait d'une chaîne de caractères tous les *url* qui sont contenus dans des liens. Représenter les *url* comme des paires de nom du serveur et nom du fichier.

Utilisez la bibliothèque `Str` de OCaml qui fournit des expressions régulières et les fonctions nécessaire pour filtrer des chaînes de caractères par des expression régulières, Voir la documentation sur `http://caml.inria.fr/pub/docs/manual-ocaml/manual037.html`.

Le protocole HTTP (Hyper Text Transfer Protocol) est un protocole de la couche application qui sert à obtenir des pages web d'un serveur web. Pour nous, il n'y a que deux formes de requêtes qui sont intéressantes : GET et HEAD. Le premier sert à obtenir une page web, le deuxième à n'obtenir que l'entête d'une page web (ce qui est suffisant pour savoir si une page est disponible sur un serveur).

On peut utiliser le programme `telnet` pour exécuter « à la main » un client web :

```
% telnet www.dptinfo.ens-cachan.fr http
Trying 138.231.36.36...
Connected to www.dptinfo.ens-cachan.fr (138.231.36.36).
Escape character is '^]'.
```

```
HEAD / HTTP/1.1
Host: www.dptinfo.ens-cachan.fr

HTTP/1.1 200 OK
Date: Wed, 10 Jan 2007 07:16:13 GMT
Server: Apache/2.0.50 (Fedora)
X-Powered-By: PHP/4.3.8
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

La requête envoyé par l'utilisateur consiste en les 3 (!) lignes dont la première commence avec HEAD.

Dans la réponse envoyée par le serveur, c'est la première ligne qui donne le statut. Les numéros commençant sur 2 indiquent que la requête peut être satisfaite, et dans le cadre de ce TP nous considérons les réponses commençant sur 3 aussi comme bon. Les numéros commençant sur 4 ou 5 indiquent des erreurs. Vous trouverez une liste complète des codes de status par exemple ici :

`http://en.wikipedia.org/wiki/HTTP_status_codes`

Exercice 2
Écrire un programme qui charge une page web (en utilisant le protocole HTTP exécuté sur une connexion TCP avec un serveur Web), et qui teste pour les liens contenus dans la page obtenue s'ils sont « cassés » ou pas.

Exercice 3
Étendre le programme de l'exercice précédent pour obtenir un outil qui charge récursivement toutes la pages web d'un site (c'est-à-dire, toutes les pages accessibles à partir d'une page de départ et qui sont logées sur le même serveur), et qui teste les liens pour l'ensemble de ces pages. Attention au fait que les liens entre les pages web d'un site peuvent former des cycles.

Exercice 4
Optimiser l'utilisation du réseau dans l'exercice précédent, on regroupant toutes les requêtes au même serveur dans une seule connexion TCP. Les versions du protocole HTTP à partir de 1.1 ne ferment pas la connexion TCP toute de suite après avoir donné une réponse, mais attendent des requêtes successives.

Par contre, la version 1.0 du protocole HTTP ne permet qu'une seule requête par connexion.