UNIVERSITÉ DE ROUEN	ANNÉE 2015-2016
FACULTÉ DES SCIENCES	RÉSEAUX
LICENCE 3 INFORMATIQUE	FICHE DE TD/TP N° 6

Création d'une bibliothèque pour la manipulation des sockets Internet pour le protocole UDP

Nous allons créer une bibliothèque de manipulation des sockets Internet pour le protocole UDP. L'ensemble des structures de données sera défini dans `socketUDP.h`. Les fonctions seront écrites dans `SocketUDP.c`. Votre bibliothèque utilisera la bibliothèque d'adressage du TP précédent.

Exercice 1

1) Rappeler le schéma général d'un serveur (d'un client) internet en mode non connecté.

2) Rappeler les paramètres à passer à l'appel système `socket` pour créer une socket Internet en mode non connecté.

3) Définir un type

`SocketUDP`

permettant de manipuler une socket BSD en mode non connecté.

Exercice 2

1) Écrire en C un client et un serveur multi-threads de transfert de fichiers courts (au plus 65535 octets) basé sur le protocole UDP. Le client demande un fichier en envoyant pour requête un nom de fichier, le serveur répond en envoyant le contenu du fichier dans un paquet. Prévoir le fait que le client ne reçoit pas nécessairement le paquet. Un paquet pouvant se perdre, on estimera dans ce protocole qu'il n'est pas nécessaire que le fichier soit reçu.

2) Modifier le protocole et donc le code du client et du serveur afin de garantir la réception du fichier.

Création d'une bibliothèque pour la manipulation des sockets en mode non connecté

Nous allons créer une bibliothèque `SocketUDP` pour la programmation des sockets Internet pour le protocole UDP. Vous penserez à structurer vos fichiers source.

Vous rendrez un zip contenant tous vos fichiers source et un Makefile permettant de compiler votre bibliothèque, ainsi qu'un programme de test testant toutes les fonctions.

Exercice 3

1) Définir un type `SocketUDP` permettant de manipuler une socket en mode non connecté.

2) Définir la fonction

```
int initSocketUDP(SocketUDP *socket);
```

qui initialise la socket `socket` (préalablement alouée).

Valeur de retour : 0 en cas de succès, -1 sinon.

3) Définir la fonction

```
int attacherSocketUDP(SocketUDP *sock, const char *address,
                      uint16_t port, int flags);
```

qui attache la socket `sock` à une adresse local donnée par `adresse` (IP ou nom DNS) et le port `port`. Si `adresse` est `NULL`, et que l'entier `flag` vaut 0, la socket est attachée à toutes les interfaces. Si l'entier `flag` vaut `LOOPBACK`, la socket est attachée à l'interface locale. Si `adresse` n'est pas `NULL`, l'entier `flag` est ignoré.

Valeur de retour : 0 en cas de succès, -1 sinon.

4) Définir la fonction

```
int estAttachee(SocketUDP *socket);
```

qui renvoie 0 si la socket a une adresse local, -1 sinon.

5) Définir la fonction

```
int getLocalName(SocketUDP *socket, char *buffer, int taille);
```

qui remplie `buffer` (préaloué de longueur `taille`) avec le nom local de la SocketUDP. Si le buffer est trop petit, le nom est tronqué (et le 0 final n'est pas écrit). Si besoin, `socket` est mis à jour.

Valeur de retour : le nombre d'octets écrits dans `buffer`, ou -1 en cas d'erreur.

6) Définir la fonction

```
int getLocalIP(const SocketUDP *socket, char *localIP, int tailleIP);
```

qui remplie `localIP` (préaloué de longueur `tailleIP`) avec l'IP locale de la SocketUDP. Si le buffer est trop petit, le nom est tronqué (et le 0 final n'est pas écrit).

Valeur de retour : le nombre d'octets écrits dans `buffer`, ou -1 en cas d'erreur.

7) Définir la fonction

```
uint16_t getLocalPort(const SocketUDP *socket);
```

qui retourne le port local de la SocketUDP.

8) Définir les fonctions suivantes :

```
ssize_t writeToSocketUDP(SocketUDP *sock, const AdresseInternet
          *address, const char *buffer, int length);
```

qui écrit sur la socket `sock` vers `adresse` un bloc d'octets `buffer` de taille `length` et retourne la taille des données écrites ou -1 s'il y a erreur.

```
ssize_t recvFromSocketUDP(SocketUDP *socket, char *buffer, int
        length, AdresseInternet *address, int timeout);
```

qui lit sur `socket` les données envoyées par une machine d'adresse `adresse`. La fonction est bloquante pendant `timeout` secondes. Elle lit et place dans `buffer` un bloc d'octets de taille au plus `length`. La fonction `recvFromSocketUDP` retourne la taille des données lues ou -1 s'il y a erreur.

9) Définir la fonction

```
int closeSocketUDP(SocketUDP *socket);
```

qui ferme la connexion (dans les 2 sens) et libère la SocketUDP.

Valeur de retour : 0 en cas de succès, -1 en cas d'erreur.