PR6 – Programmation réseaux
TP n° 6 : Clients et Serveurs TCP en C

En C, tout est de bas-niveau, c'est à dire que contrairement au java beacoup de choses sont à faire "à la main", et il sera important d'avoir l'API C bien en tête, en voici un petit rappel.
Les prototypes des fonctions supplémentaires nécessaires se trouvent dans les fichiers entêtes suivants :

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
```

Les fonctions principales sont :

```
int socket(int domaine, int type, int protocole);
int bind(int socket, const struct sockaddr *adresse, socklen_t longueur);
int listen(int socket, int attente);
int accept(int socket, struct sockaddr *adresse, socklen_t *longueur);
int connect(int socket, const struct sockaddr *adresse, socklen_t longueur);
int shutdown(int socket, int how);
int close(int socket);
ssize_t send(int socket,const void *tampon, size_t longueur,int options);
ssize_t recv(int socket,void *tampon, size_t longueur,int options);
struct hostent *gethostbyname(const char *name);
int getaddrinfo(const char *hostname, const char *servname,
       const struct addrinfo *hints, struct addrinfo **res);
```

Exercice 1 : Un client TCP pour daytime en C

Le but est d'écrire un client en C pour le service `daytime` tournant sur `lucien` et affichant également l'adresse IP de `lucien`. On proposera deux implémentations, une utilisant la fonction `gethostbyname` et l'autre utilisant la fonction `getaddrinfo`.

Exercice 2 : Discussions entre serveurs Java et C

On souhaite programmer deux entités 1 et 2 discutant via TCP entre elles et chacune liée à un port et une machine.

1. L'entité 1 aura le comportement suivant. Elle attendra un message d'un client sur son port. Ce message aura la forme suivante `adresse_ip port\n` où `adresse_ip` est une chaînes de caractère représentant une adresse IP et `port` est une chaîne de caractères représentant un numéro de port. À la réception du message il ferme son serveur et il se connecte à l'adresse IP donnée sur le port fourni et envoie le message `CONFIRM\n`. Ensuite il attend un message de la forme `ACKCONFIRM\n`. Il finit en fermant la connexion.

2. L'entité 2 aura le comportement suivant. Elle enverra un message `adresse_ip port\n` à l'entité 1. Elle fermera la connexion. Elle attendra un message sur son adresse IP `adresse_ip` et sur son port `port` qui aura la forme `CONFIRM\n` et elle répondra avec un message de la forme `ACKCONFIRM\n`, puis elle fermera la connexion.

On vous demande de programmer ces deux entités en C et en Java et de les tester de façon croisée.