Objectifs du TD :

1 TP5 et TP6 : Client/Serveur en TCP

Exercice 1

1. Créer deux programmes permettant d'engendrer des processus communiquant par le protocole TCP, selon le modèle *Client - Serveur*. La base de travail sera les classes `C++` distribuées précedemment et à votre disposition.

   Le Client `clientTCP` expédie un message contenant le numéro d'utilisateur ayant lancé ce processus client (voir l'appel système `getuid()`).

   Le serveur `serverTCP` lui répondra *Tiens, bonjour* suivi du nom d'utilisateur correspondant au numéro transmis (voir l'appel système vu en cours de *systèmes d'exploitation*, `getpwuid()`).

   Le serveur traite les requêtes une à une, c'est-à-dire qu'on répond complètement à un client, on ferme la connexion puis on passe au client suivant.
2. Nous avons vu que le protocole TCP était orienté *flot de caractères*. Il n'y a pas de limite de message qui soit assurée par le protocole. Du coup, il n'y a pas de correspondance entre le nombre d'envois (écritures) et le nombre de réceptions (lectures) pour un *message*.

   Est-ce que les processus que vous avez prévus tiennent compte de cette caractéristique ?Sinon, modifier vos programmes.

2 Appels système pratiques

Exercice 2

— L'appel `getsockname()`, déjà vu dans un TD précédent, permet de récupérer l'adresse de la socket (boîte réseau) dont le descripteur est `s`.

```
int  getsockname(int   s, struct sockaddr * name, int * namelen )
```

— L'appel `getpeername()` permet de récupérer l'adresse de la socket connectée à la socket `s`.

```
 int  getpeername(int  s, struct sockaddr *name, int *namelen);
```

1. Un client en mode connecté peut laisser l'appel `connect()` compléter la demande d'allocation de sa boîte réseau. Écrire un programme permettant d'afficher le numéro de port réellement obtenu après avoir fait la demande `connect()`. On pourra afficher le numéro avant cette demande, pour constater que l'appel système a fait son travail.
2. Afficher sur un serveur en mode connecté les caractéristiques du client demandant une connexion.

3 Un serveur concurrent et un problème de performances

Exercice 3

On veut mettre en place un serveur de fichiers, à la manière de `ftp`, afin de faire des mesures et comparer ses performances aux outils classiques de transfert de fichiers (`ftp` ou ce même protocole sous un navigateur par exemple).

1. Ecrire un serveur et un client pour cette application. Le client ne peut que demander un fichier et le serveur le lui expédie lorsqu'il existe, ou lui annonce son inexistence : on ne prévoit pas d'autres services que le transfert de fichiers. Bien sûr plusieurs clients sont possibles et peuvent travailler simultanément. Chaque client peut demander plusieurs fichiers.
2. Est-ce que le client peut savoir s'il a reçu un fichier intégralement (est-ce prévu dans votre protocole) ?Sinon, modifier votre programme.
3. Si on veut limiter le nombre de clients simultanés, quelle solution peut-on proposer ? Mettez les en place