td7.md

Exercice 1 - Une fenêtre sur le monde

Sur vos machines, une `struct sockaddr_in` a la forme suivante:

```
struct sockaddr_in{
    sa_family_t    sin_family;          /* famille d'adresse : AF_INET, ...*/
    in_port_t      sin_port;            /* No port.            */
    struct in_addr sin_addr;            /* Adresse internet    */
    unsigned char  sin_zero[sizeof (struct sockaddr) -   /* Remplissage */
                            sizeof (sa_family_t) -       /* jusqu'a     */
                            sizeof (in_port_t) -         /* la taille de*/
                            sizeof (struct in_addr)];    /* sockaddr */
};

struct in_addr{
    uint32_t   s_addr;
};
```

- Afin de créer une socket en écoute, il faut remplir une `struct sockaddr_in`, de famille `AF_INET`. Écrire une fonction qui retourne une telle structure remplie avec un port donné en argument, au dessus de 1024.
  Attention : il faut utiliser `htons` pour mettre le numéro de port sous le bon format.
  Dans le champ adresse de la structure, on mettra `INADDR_ANY` qui signifie qu'on veut écouter sur toutes les interfaces de la machine.
- Écrire une fonction qui étant donné un port, crée une socket en utilisant la fonction précédente pour récupérer une `struct sockaddr_in`. Pour écouter sur ce port, relie la socket à l'adresse grâce à `bind()`, puis met la socket en écoute avec `listen()`. La fonction retournera la socket en écoute.
- Mettre la socket ainsi créée en attente de connection avec `accept`. Cet appel est bloquant jusqu'à ce que quelqu'un se connecte sur le port spécifié; il retourne alors un descripteur sur une nouvelle socket qui permet de communiquer, l'ancienne socket restant en écoute.
- Écrire un programme qui écoute sur le port passé en argument, de sorte que lorsque l'on se connecte (avec la commande `nc` par exemple), le message "Bonjour" s'affiche. Essayez de vous connecter au programme tournant sur la machine voisine.

Exercice 2 - Fil d'attente

Faire un serveur de numéro de passage (comme à la sécu, dans les préféctures, tout ça) multi-guichets. Plusieurs guichets peuvent se connecter, et démander, quand ils ont terminé leur travail, le numéro du suivant.

Le protocol guichet

1. un guichet (client) se connecte au serveur en indiquant son numéro de guichet
2. quand le guichet n'a rien a faire, il demande au server le prochain numéro à traiter
3. quand le serveur lui annonce un numéro à traiter, le guichet s'en occupe (simulé par une attente de 1-5 secondes, par hazard, `sleep(1 + rand() % 5)` par example), puis demande le prochain numéro

Le protocol serveur

Utiliser `select` ou `poll` pour attendre qu'un (ou plusieurs) des choses suivant se passe:

1. un nouveau guichet s'enregistre:
   - prendre son numéro guichet
2. un guichet connu demande un nouveau numéro
   - lui donner le premier numéro à traiter
   - annonce sur `stdout` le message "le numéro X est attendu au guichet Y"

Testez le programme avec au moins 3 guichets (soit lancé dans 3 terminals differents, soit lancé avec `&` comme ça: `./guichet &` (n'oubliez pas les arreter à la fin du test: `killall guichet`)