

TE Lab2 Cheatsheet

SYE
-

Résumé du document

Definition

Table des matières

- 1. Laboratoire pipelining 2
- 2. Laboratoire sockets 3
 - 2.1. Sockets 3
 - 2.1.1. Création d’un serveur TCP 3

1. Laboratoire pipelining

2. Laboratoire sockets

Le laboratoire concerne la création d'un jeu de bataille navale en traitant les sujets de **sockets** ainsi que la gestion des **signaux**.

2.1. Sockets

2.1.1. Création d'un serveur TCP

Voici le code permettant de créer la partie serveur

```
int server_create(const int port)
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    // Création de la socket
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenfd < 0) {
        return -1;
    }

    // Initialisation de l'adresse du serveur
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr = (struct sockaddr_in) {
        .sin_family = AF_INET,
        .sin_addr.s_addr = htonl(INADDR_ANY),
        .sin_port = htons(port),
    };

    // Association de la socket à l'adresse
    if (bind(listenfd, (struct sockaddr *)
&serv_addr, sizeof(serv_addr)) < 0) {
        close(listenfd);
        return -1;
    }

    // Mise en écoute de la socket
    // listen(listenfd, 1) correspond a la
    longueur de la file d'attente des connexions
    if (listen(listenfd, 1) < 0) {
        close(listenfd);
        return -1;
    }

    printf("%s %d...\n", strings[SERVER_LISTEN],
port);

    // Acceptation d'une connexion
    connfd = accept(listenfd, (struct sockaddr
*) NULL, NULL);
    if(connfd < 0) {
        close(listenfd);
        return -1;
    }

    close(listenfd);

    printf("%s\n",
strings[SERVER_CLIENT_CONNECT]);

    return connfd;
}
```

1. Créer des variables contenant le file descripteur pour lire les connexions entrantes `listenfd` et pour gérer la connexion **spécifique** avec un client `connfd`.
2. **SOCKET**: créer la socket sur le fd `listenfd` dans le but de lire les tentatives de connexions.
3. Permet de définir la structure de l'adresse du serveur grâce à la struct `sockaddr_in`.
 - `sin_family` : spécifie IPv4.
 - `sin_addr.s_addr` : accepte les connexions sur n'importe quelle interface réseau (adresse locale) grâce à `htonl(INADDR_ANY)`.
 - `sin_port` : convertit le numéro de port en ordre réseau (big-endian) via `htons(port)`.
4. **BIND**: l'appel à `bind` associe la socket représentée par le descripteur `listenfd` à une **adresse IP** et un **port** spécifiés dans la structure `serv_addr`. Cela permet au système d'identifier sur quelle interface réseau et sur quel port le serveur doit écouter les connexions.
5. **LISTEN**: l'appel à `listen` place la socket en mode écoute pour les connexions entrantes. Elle prépare le serveur à accepter des connexions client avec `accept`.
6. **ACCEPT**: l'appel à `accept` permet au serveur d'accepter une connexion entrante d'un client. Il utilisera le descripteur de fichier (`connfd`) pour la communication avec ce client spécifique.
7. On ferme le file descriptor `listenfd` car notre connexion est désormais établie et celui-ci ne sert plus. Une fois les affichages fait, on retourne donc la variable `connfd` qui permettra au serveur de discuter avec le client connecté.