Séance 3
Communications réseau – Sockets TCP/IP

Gestion des erreurs sur les appels système

Appels système : retour de -1 en cas d'échec
- code d'erreur précis dans variable errno
- fonction perror : affichage message d'erreur correspondant à errno

Tester tous les retours et si échec appeler **erreur_IO** (fonction de la bibliothèque modules).

void erreur_IO (char *messageProgrammeur) utilise perror et fait exit :
affichage du message programmeur + affichage du message d'erreur système + exit

```
fd = open("monfifo", …);
if (fd ==  -1)
   erreur_IO("ouverture fifo");          si monfifo n'existe pas affiche : "ouverture fifo: No such file or directory"
```

Socket TCP/IP : communication client/serveur

Socket TCP/IP : mécanisme de communication entre applications sur des machines distantes.

Mécanisme dissymétrique pour la connexion :

- une application jour le rôle de serveur (en attente de connexion)
- une application jour le rôle de client (se connecte au serveur)

Séquence pour la connexion côté serveur (1)

1) Création d'un socket

socket*(famille, type, protocole)*

retourne un socket (un entier du type descripteur de fichier) ou -1

```
int soc1;
soc1 = socket(AF_INET, SOCK_STREAM, 0);
```

2) Lier le socket à une adresse réseau (celle de la machine hôte) et un port

bind*(socket, structure contenant adresse et port, taille structure)*

```
struct sockaddr_in  monadr;
monadr.sin_family = AF_INET;
monadr.sin_addr.s_addr = INADDR_ANY;            // toutes les adresses de la machine
monadr.sin_port = htons(2000);                  // port à choisir dans la plage libre
bind (soc1, (struct sockaddr*) &monadr, sizeof(monadr));
```

retourne 0 ou -1

Séquence pour la connexion côté serveur (2)

3) <u>Se mettre à l'écoute des connexions</u>

listen *(socket, taille file d'attente des connexions)*

ne se met pas en attente de connexion mais demande au noyau de se mettre à l'écoute des connexions entrantes

```
listen(soc1, 5);
```

retourne 0 ou -1

4) <u>Accepter une connexion entrante</u>

accept*(socket, structure où ranger l'adresse IP du client se connectant, entier où ranger la taille de la structure)*

```
struct sockaddr_in adrcli;
unsigned int lgadr = sizeof(adrcli);
int soc2;
soc2 = accept (soc1, (struct sockaddr*) &adrcli, &lgadr);
```

accept <u>bloque</u> en attente d'une demande de connexion par un client,
ou prend la 1ère connexion dans la file d'attente
en sortie de accept une connexion est établie

accept retourne un socket qui sera utilisé pour les échanges, ou -1

Séquence pour la connexion côté client

1) <u>Création d'un socket</u>
***socket**(…)*

```
int soc;
soc = socket(AF_INET, SOCK_STREAM, 0);
```

2) <u>Connexion au serveur</u>
connect (socket, structure contenant l'adresse du serveur et le port, taille de la structure)

```
struct sockaddr_in *adrserv = resolv("www.emse.fr", "2000");
connect(soc, (struct sockaddr *) adrserv, sizeof(struct sockaddr_in));
```

retourne 0 ou -1

Envoi-réception de données

Serveur

```
soc2 = accept(…);
connexion établie

pour recevoir et envoyer des données
read(soc2, …) et write(soc2, …)


close(soc2);    // fermeture de la connexion
```

Client

```
connect(soc, …);
connexion établie

pour recevoir et envoyer des données
read(soc, …) et write(soc, …)


close(soc);    // fermeture de la connexion
```

- **read bloquant** si pas de données
- les donnues lues sont extraites

Mémorisation d'une demande de connexion

Si le client se connecte (connect) alors que le serveur n'est pas en attente de connexion (accept) :

- la demande de connexion est mémorisée dans une file d'attente (taille donnée sur listen),
- connect sort immédiatement et sans erreur,
- des write qui suivent sont acceptés et mémorisés