Réseaux : modèles, protocoles, programmation

Pablo Rauzy
pablo.rauzy@univ-paris8.fr
pablo.rauzy.name/teaching/rmpp

UFR MITSIC / L2 informatique

Séance a

Programmation réseau (un peu plus) avancée

Programmation réseau (un peu plus) avancée

- Avec notre programme `ncat`, nous avons déjà vu comment établir une connexion client-serveur et échanger des messages.
- Aujourd'hui nous allons voir comment un serveur peut gérer plusieurs clients simultanément.
- Deux problématiques :
  - Comment discuter avec plusieurs clients en même temps ?
  - Comment continuer pendant ce temps à accepter des nouveaux clients ?

Multiplexage d'entrées/sorties

- Dans notre programme `ncat`, on a déjà vu comment utiliser `poll` pour attendre des données à la fois d'un client et sur l'entrée standard.
- Le même mécanisme peut être utilisé pour "écouter" sur plusieurs sockets.

Accepter plusieurs clients

- On peut aussi utiliser `poll` sur une socket passive.
- Dans ce cas, l'évènement `POLLIN` signifiera qu'on peut `accept`er un nouveau client.
- Attention toutefois, `accept` est un appel bloquant.

Accepter plusieurs clients

- On peut aussi utiliser `poll` sur une socket passive.
- Dans ce cas, l'évènement `POLLIN` signifiera qu'on peut `accept`er un nouveau client.
- Attention toutefois, `accept` est un appel bloquant.
- Il est donc nécessaire de rendre la socket du serveur *non bloquante*.
  - On peut le faire avec un appel à `fcntl`.
  - Dans ce cas, `accept` retournera -1 et mettre `errno` à la valeur `EWOULDBLOCK` à la place de bloquer.

Code

- Codons ensemble un serveur d'écho qui broadcast les messages à tous ses clients.

- Il est généralement important de s'assurer de la fiabilité des communications.
- TCP fait en partie cela pour vous, mais il ne peut pas faire de magie non plus.
- C'est à vous de gérer les envoies partiels, et de vous assurer en réception d'avoir bien reçu l'intégralité de ce qui vous a été envoyé.

Gestion des envois partiels

- Exercice :
  - Vous avez un buffer `buf` d'une longueur `len`.
  - Vous devez vous assurer de son envoi complet avec `send` ou `write` (au choix).

  → Écrivez une fonction `int sendall (int fd, void *buf, int *len);` qui agit comme `write` mais s'assure d'avoir tout envoyé.
  - Si elle échoue, la fonction renvoie -1 et place dans l'argument `len` le nombre d'octets effectivement envoyés, sinon elle renvoie 0.

- Exercice :
  - En réception, comment être sûr qu'on a bien reçu l'intégralité d'un message ?
  - Quels mécanismes connaissez vous pour arriver à ce but ?

- Voyons comment gérer IPv6 en plus d'IPv4.

Adresse de socket

- Pour IPv6 :
  - remplacer

```
1 struct sockaddr_in sa4;
2 sa4.sin_family = AF_INET;
3 sa4.sin_addr.s_addr = INADDR_ANY;
4 sa4.sin_port = htons(atoi(argv[1]));
```

  - par

```
1 struct sockaddr_in6 sa6;
2 sa6.sin6_family = AF_INET6;
3 sa6.sin6_addr = in6addr_any;
4 sa6.sin6_port = htons(atoi(argv[1]));
5 /* in6addr_any est une struct in6_addr
6    initialisée à IN6ADDR_ANY_INIT */
```

Manipulation d'adresses IP

- À la place de `inet_ntoa`, utiliser `inet_ntop` :
  - remplacer

```
1 printf("%s\n", inet_ntoa(sa4.sin_addr));
```

  - par

```
1 char str_addr[INET6_ADDRSTRLEN];
2 inet_ntop(AF_INET6, &(sa6.sin6_addr), str_addr, INET6_ADDRSTRLEN);
3 printf("%s\n", str_addr);
```

- À la place de `inet_aton`, utiliser `inet_pton` :
  - remplacer

```
1 inet_aton("127.0.0.1", &(sa4.sin_addr));
```

  - par

```
1 inet_pton(AF_INET6, "::1", &(sa6.sin6_addr));
```

Gérer indifféremment les IPv4 et IPv6

- Les nouvelles fonctions `inet_ntop` et `inet_pton` sont capables de gérer aussi bien IPv6 que IPv4 (premier paramètre), et sont à utiliser de préférences par rapport aux anciennes.
- Il existe une fonction `getaddrinfo` et une structure `struct addrinfo` qui permettent d'utiliser tout ça de manière transparente.
- La fonction `getaddrinfo` fait beaucoup de choses :
  - remplace l'appelle à `gethostbyname`,
  - est capable de remplir correctement une `struct sockaddr` côté serveur,
  - peut récupérer les différentes adresses possibles (IPv4 et IPv6) pour un hôte,
- Le mieux est de voir un exemple : utilisons la ensemble.