TCP, I/O, Buffers, send/recv et read/write sur Linux

Introduction

Ce micro-cours aborde les fondamentaux de TCP, des opérations d'entrée/sortie (I/O), des buffers, ainsi que des fonctions send/recv et read/write sur Linux, avec un accent sur leur application pratique.

1. TCP (Transmission Control Protocol)

Concepts Clés

- **Orienté Connexion :** Établissement d'une connexion fiable entre deux hôtes.
- **Contrôle de Flux et de Congestion :** Gestion de l'intégrité des données et prévention de la surcharge du réseau.
- **Transmission Ordonnée et Fiable :** Livraison des paquets dans l'ordre et sans erreur.

2. I/O sur Linux

Types d'I/O

- **Bloquant :** L'opération bloque le processus jusqu'à ce que les données soient disponibles.
- **Non-Bloquant :** Retour immédiat, même sans disponibilité des données.

3. Buffers

Gestion des Buffers

- **Buffer de Réception :** Stockage temporaire des données entrantes.
- **Buffer d'Envoi :** Accumulation des données sortantes avant transmission.

Raison d'Être des Buffers

Les buffers permettent de gérer les différences de vitesse entre la production et la consommation de données, évitant la perte de données et réduisant les opérations d'I/O constantes.

4. send/recv vs read/write

send/recv

- **Spécificité :** Pour les sockets, avec des options pour les transmissions réseau.

read/write

- **Généralité :** Pour tout descripteur de fichier, y compris les sockets.

Exemple Pratique : Buffer de Réception

Serveur TCP (Utilisant read)

```
char buffer[10];
int bytesRead;

// Première lecture
bytesRead = read(new_socket, buffer, 5);
printf("1ère réception: %.*s\n", bytesRead, buffer);

// Seconde lecture
bytesRead = read(new_socket, buffer, 5);
printf("2ème réception: %.*s\n", bytesRead, buffer);
```

Client TCP (Utilisant write)

```
char *message = "HelloWorld";
write(sock, message, strlen(message));
```

Explication :

- Le client envoie "HelloWorld" (10 octets).
- Le serveur lit d'abord 5 octets, puis les 5 octets restants dans une seconde lecture.
- Cette séquence démontre comment le buffer de réception stocke les données non lues pour une lecture ultérieure.