Résumé et QCM Sockets

Voici un résumé du cours ainsi qu'un QCM pour t'aider à réviser.

Résumé du cours : Programmation Client-Serveur avec Sockets

1. Notions Générales

- La programmation client-serveur permet la communication entre processus sur un même ordinateur ou sur un réseau.
- Un **protocole** définit les règles de communication (format des messages, détection de fin, acheminement, etc.).
- Le **modèle client-serveur** repose sur un serveur qui répond aux requêtes d'un client.

2. Sockets

- Une **socket** est un point de communication permettant l'échange de données entre processus.
- Se situe juste au-dessus de la couche transport (TCP/UDP) du modèle OSI.
- Utilise des **ports** pour identifier les applications en communication.

3. Types de Sockets

- **SOCK_STREAM (TCP)** : Communication fiable, mode connecté.
- **SOCK_DGRAM (UDP)** : Communication rapide, mode non connecté.

4. Primitives des Sockets

- **Création** : `socket()`
- **Attachement à une adresse** : `bind()`
- **Écoute des connexions** (mode connecté) : `listen()`
- **Connexion du client au serveur** : `connect()`
- **Acceptation d’une connexion** : `accept()`

- **Envoi et réception de données :** `send()`, `recv()`, `write()`, `read()`
- **Fermeture de la connexion :** `close()`

5. Mode Connecté vs Non Connecté

Mode	Connecté (TCP)	Non Connecté (UDP)
Connexion préalable	Oui	Non
Fiabilité	Oui	Non garantie
Performance	Plus lent	Plus rapide
Utilisation	Communication stable	Transmission rapide

6. Gestion des Processus

- `fork()` : Création d'un processus fils pour gérer plusieurs clients en parallèle.
- `getpid()`, `getppid()` : Récupération des identifiants de processus.

7. Résolution d'Adresse

- `inet_addr()` : Convertir une adresse IP en format numérique.
- `gethostbyname()`, `gethostbyaddr()` : Résolution DNS pour obtenir des informations sur une machine.

QCM – Programmation Client-Serveur avec Sockets

1. Quel est le rôle principal d'un serveur dans une architecture client-serveur ?
a) Envoyer des requêtes aux clients
b) Répondre aux requêtes des clients
c) Se connecter à Internet
d) Intercepter les communications

2. Quelle fonction est utilisée pour créer une socket en C ?
a) `bind()`
b) `connect()`
c) `socket()`
d) `listen()`

3. Quel protocole garantit une communication fiable entre client et serveur ?

a) UDP

b) TCP

c) HTTP

d) FTP

4. Quelle est la fonction utilisée pour lier une socket à une adresse IP et un port ?

a) `socket()`

b) `bind()`

c) `listen()`

d) `connect()`

5. Quelle primitive permet à un serveur d'attendre les connexions entrantes ?

a) `listen()`

b) `accept()`

c) `connect()`

d) `send()`

6. Quelle est la différence principale entre TCP et UDP ?

a) TCP est plus rapide qu'UDP

b) UDP est plus fiable que TCP

c) TCP assure une connexion stable, UDP non

d) UDP est utilisé uniquement pour le web

7. Quelle fonction est utilisée pour envoyer des données en mode connecté ?

a) `recv()`

b) `send()`

c) `bind()`

d) `accept()`

8. Quelle fonction est utilisée pour gérer plusieurs clients en parallèle ?

a) `fork()`

b) `accept()`

c) `bind()`

d) `socket()`

9. Que retourne `fork()` dans le processus fils ?

a) -1

b) 0

c) L'identifiant du processus père

d) Une valeur aléatoire

10. Quelle fonction est utilisée pour convertir un nom de domaine en adresse IP ?

a) `gethostbyname()`

b) `inet_addr()`

c) `getpid()`

d) `fork()`

Si tu veux des réponses détaillées aux questions, dis-moi ! 😊

Voici les réponses détaillées du QCM avec explications :

---

1. Quel est le rôle principal d'un serveur dans une architecture client-serveur ?

✅ **Réponse : b) Répondre aux requêtes des clients**

🔍 **Explication :**

Dans une architecture client-serveur, le serveur est un programme qui attend et répond aux requêtes envoyées par un ou plusieurs clients. Il peut fournir des services comme l'accès à une base de données, l'envoi de fichiers ou l'exécution de calculs à distance.

---

2. Quelle fonction est utilisée pour créer une socket en C ?

✅ **Réponse : c)** `socket()`

🔍 **Explication :**

La fonction `socket()` permet de créer une socket et retourne un descripteur de fichier qui la représente. Elle prend en paramètre le domaine (ex: `AF_INET` pour IPv4), le type (ex: `SOCK_STREAM` pour TCP) et le protocole (ex: `0` pour laisser le système choisir le bon protocole).

Exemple :

```c
```

```c
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
    printf("Erreur de création de la socket\n");
}
```

---

3. Quel protocole garantit une communication fiable entre client et serveur ?

✅ **Réponse : b) TCP**

🔍 **Explication :**

TCP (Transmission Control Protocol) assure une transmission fiable et ordonnée des données en garantissant qu'aucun paquet n'est perdu ou corrompu. Il utilise un mécanisme de contrôle d'erreur et de retransmission. UDP, en revanche, est plus rapide mais ne garantit pas la fiabilité des transmissions.

---

4. Quelle est la fonction utilisée pour lier une socket à une adresse IP et un port ?

✅ **Réponse : b)** `bind()`

🔍 **Explication :**

La fonction `bind()` associe une socket à une adresse IP et un numéro de port spécifique. Elle est utilisée principalement du côté du serveur pour être en écoute sur un port précis.

Exemple :

```c

struct sockaddr_in server_addr;
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(8080);
server_addr.sin_addr.s_addr = INADDR_ANY;
bind(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr));
```

5. Quelle primitive permet à un serveur d'attendre les connexions entrantes ?

✅ **Réponse : a)** `listen()`

🔍 **Explication :**

Une fois la socket créée et liée à un port avec `bind()`, le serveur doit appeler `listen()` pour indiquer qu'il est prêt à accepter des connexions entrantes.

Exemple :

```c
c

listen(sockfd, 5); // 5 est la taille de la file d'attente
```

---

6. Quelle est la différence principale entre TCP et UDP ?

✅ **Réponse : c) TCP assure une connexion stable, UDP non**

🔍 **Explication :**

- TCP établit une connexion avant d'envoyer des données, garantissant leur livraison et leur ordre.
- UDP envoie les paquets directement sans vérification, ce qui le rend plus rapide mais moins fiable.
- UDP est utilisé pour le streaming et les jeux en ligne où la rapidité est plus importante que la fiabilité.

---

7. Quelle fonction est utilisée pour envoyer des données en mode connecté ?

✅ **Réponse : b)** `send()`

🔍 **Explication :**

La fonction `send()` est utilisée pour envoyer des données via une socket connectée en TCP.

Exemple :

```c
char message[] = "Hello Client!";
send(sockfd, message, sizeof(message), 0);
```

- Elle retourne le nombre d'octets envoyés ou -1 en cas d'erreur.

---

8. Quelle fonction est utilisée pour gérer plusieurs clients en parallèle ?

✅ **Réponse : a)** `fork()`

🔍 **Explication :**

La fonction `fork()` permet au serveur de créer un processus fils pour gérer chaque client individuellement, permettant ainsi de traiter plusieurs connexions en parallèle.

Exemple :

```c
int pid = fork();
if (pid == 0) {
    // Processus fils (client)
    handle_client(newsockfd);
}
```

- Le processus père continue d'accepter de nouveaux clients.

---

9. Que retourne `fork()` dans le processus fils ?

✅ **Réponse : b) 0**

🔍 **Explication :**

- `fork()` crée un processus fils qui exécute le même code que son père.

- **Dans le processus fils,** `fork()` retourne `0`.
- **Dans le processus parent,** `fork()` retourne l'identifiant (PID) du processus fils.

---

10. Quelle fonction est utilisée pour convertir un nom de domaine en adresse IP ?

✅ **Réponse : a)** `gethostbyname()`

🔍 **Explication :**

Cette fonction permet d'obtenir l'adresse IP associée à un nom de domaine.

Exemple :

```c
struct hostent *host;
host = gethostbyname("www.google.com");
printf("Adresse IP : %s\n", inet_ntoa(*(struct in_addr*)host->h_addr));
```

✅ **Avec ces explications, tu as toutes les clés pour réussir ton examen !** 😊

Si tu veux d'autres exercices ou des précisions, n'hésite pas à demander. 🚀