TP4

contexte général

On considère une application client-serveur, comprenant un programme serveur et un programme client. Il n'y aura qu'une seule instance du programme serveur, en revanche il pourra y avoir *k* instances du programme client (un programme pour chaque client désirant se connecter au serveur).

Cette application permet de réserver des places pour un ensemble de spectacles.

Des clients émettent deux types de requêtes à destination du serveur :

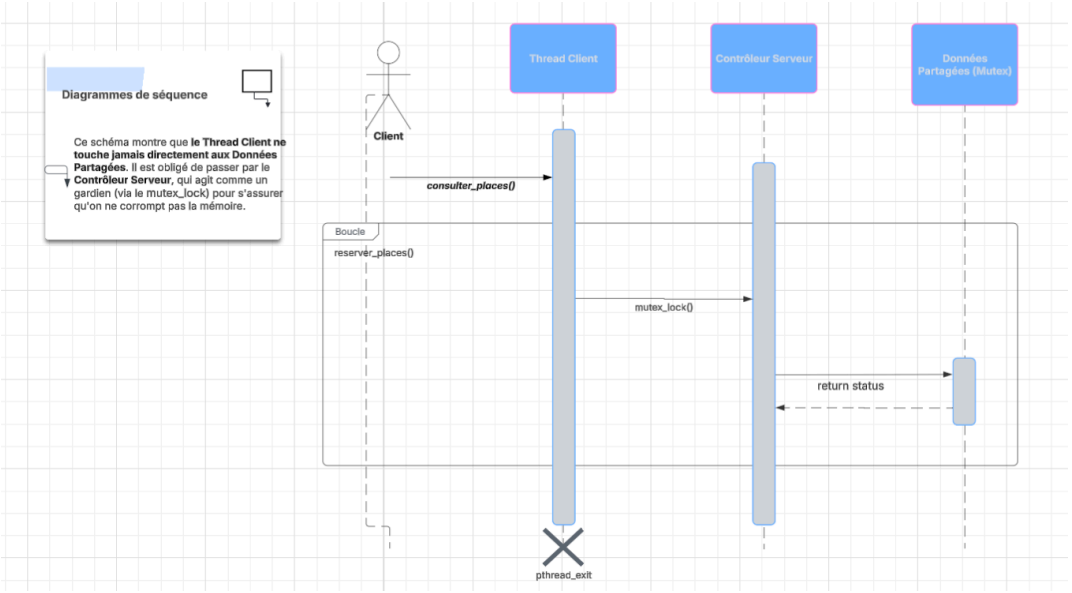
- Requête de consultation : permettant de consulter le nombre de places restantes pour un spectacle donné.
- Requête de réservation : permettant de réserver *n* places pour un spectacle donné.

Le serveur de gestion de places de spectacles est composé de deux parties :

- Partie Consultation : cette partie prend en compte les requêtes de consultation. Pour chacune d'elle, cette fonctionnalité renvoie le nombre de places disponibles pour le spectacle spécifié dans la requête.
- Partie Réservation : cette partie prend en compte les requêtes de réservation. Pour chacune d'elle, cette fonctionnalité effectue la réservation si cela est possible (suffisamment de places disponibles pour la date et le spectacle indiqué). Dans le cas où la réservation a pu être faite, un acquittement de réservation est renvoyé au client et sinon un message d'erreur.

Les informations concernant les spectacles sont stockées dans une table en mémoire centrale du côté du serveur, qui est accessible par les deux parties du serveur (parties Consultation et Réservation).
Une entrée de la table concerne un spectacle et donne les informations suivantes : Intitulé du spectacle, nombre de places restantes.
Chaque spectacle est joué une seule fois.

diagramme de sequences :



1. Architecture des Processus

Nous avons opté pour une architecture Client-Serveur centralisée créée via un fork().
Processus Père (Serveur) : *Il est unique. Il détient la "vérité" (la table des réservations) et tourne en boucle infinie pour traiter les demandes.*

Processus Fils (Client) : *Il interagit avec l'utilisateur (affichage du menu, saisie clavier). Il ne prend aucune décision, il se contente de transmettre les ordres.*

2. Communication : Nombre de Tubes (Pipes)

```c
int main() {
    int pipe_req[2];
    int pipe_res[2];
```

Pour permettre une communication (dialogue) entre le client et le serveur, nous avons besoin de 2 tubes anonymes.

Tube pipe_req (Client → Serveur) : Utilisé par le client pour envoyer ses commandes (Consultation, Réservation).

Tube pipe_res (Serveur → Client) : Utilisé par le serveur pour renvoyer le résultat (Succès, Échec, Nombre de places).

3. Maintien du Tableau des Réservations

Les données sont stockées dans un tableau global Spectacle table_spectacles[].

```c
void init_spectacles() {
    table_spectacles[0] = (Spectacle){0, "Le Roi Lion", 100};
    table_spectacles[1] = (Spectacle){1, "Notre Dame de Paris", 50};
    table_spectacles[2] = (Spectacle){2, "Moliere", 10};
    table_spectacles[3] = (Spectacle){3, "Starmania", 0};
    nb_spectacles = 4;
}
```

Localisation : Bien que déclaré globalement, seule la mémoire du processus Serveur est considérée comme valide.

Accès : Le Client n'a jamais le droit de modifier ce tableau directement (même s'il en a une copie technique au moment du fork, cette copie devient obsolète dès la première modification).

Mise à jour : Toute modification (ex: décrémenter le nombre de places) se fait exclusivement par le Serveur après réception d'une requête.

4. Structure des Données Échangées

Au lieu d'envoyer du texte brut (difficile à analyser), nous échangeons des structures binaires (struct) de taille fixe.

A. La Requête (struct Requete) Envoyée par le Client. Elle contient :

```c
typedef struct {
    TypeRequete type;
    int id_spectacle;
    int nb_places;
} Requete;
```

Type : L'action à faire (1=Consulter, 2=Réserver, 3=Lister).
ID Spectacle : Un entier simple (0, 1, 2...) pour identifier le spectacle (plus simple et rapide que d'envoyer le titre complet).
Nombre de places : La quantité demandée (pour les réservations).

B. La Réponse (struct Reponse) Envoyée par le Serveur. Elle contient :

```c
typedef struct {
    int status;
    char message[MAX_BUFFER];
    int val_retour;
    Spectacle liste[MAX_SPECTACLES];
} Reponse;
```

Status : 1 (Succès) ou 0 (Échec).
Message : Une chaîne de caractères expliquant le résultat (ex: "Réservation confirmée").
Liste : Une copie complète de la table des spectacles (uniquement envoyée si le client demande la liste).