Compte-rendu projet système d’exploitation :

Calcul des sommes préfixes d’un tableau d’entiers

Algorithme de Hillis-Steele Scan

Ouattara Umm-Habibah
L3 Info

Table des matières

I. Manuel utilisateur

1) Il faut vous munir de 2 terminals
2) Dans le premier faire un make client
3) Dans le second faire un make demon
4) Lancer le client en faisant ./client en précisant les arguments du tableau et l'opération puis faire Entrée
5) Dans le second terminal faire ./demon puis Entrée

II. Manuel technique

But du projet

Le but de ce projet est de réaliser le calcul des sommes préfixes d'un tableau d'entiers, sous la forme d'un serveur démon qui répond aux requêtes émises par les clients. Le serveur doit pouvoir gérer plusieurs clients à la fois. Le projet est réalisé en 2 parties.

- Une partie client qui utilisera un tube nommé pour y déposer ses requêtes
- Une partie serveur qui se chargera de récupérer les requêtes et de les exécuter

Choix des structures

1) Client

   Le client pour envoyer sa requête, va utiliser une structure de nom request_t définie dans un fichier de configuration ***config.h*** commun au client et au serveur comme ci-dessous

```
typedef struct {
    char ope[OPE_TAB_SIZE];
    size_t size;
    pid_t pid;
} request_t;
```

   Cette structure qui contient :

   - un champ ope, tableau de caractère qui va contenir l'opération demandée
   - un champ size, qui sera la taille d'un tableau d'entier
   - un champ pid, qui sera le pid du client qui envoie sa demande

2) Serveur

   Le serveur, lui aussi utilise la structure request_t pour pouvoir être capable de lire la requête du client. En plus de request_t, le serveur détient une autre structure de données. Cette second structure, lui permet de créer des threads, qui auront leur propre numéro mais aussi les données relatives aux requêtes clients.

```c
// arguments transmis au thread
struct thread_arg {
  int numTh; // numéro pour le thread
  size_t size;
  pid_t pid;
  char ope[OPE_TAB_SIZE];
  char shm_name[SEGMENT_NAME_SIZE];
};
```

Fonctionnement du client

Exemple d'une ligne de commande : ./client 4 -1 3 0 1 2 -2 5 +

- Récupération des arguments du tableau

  Le client devra renseigner sur la ligne de commande les éléments du tableau dont il souhaite effectuer la somme, ainsi que l'opération souhaitée. Il va se charger ensuite de convertir les caractères argv[1] à argv[argc 1]. Si on reprend l'exemple, il va convertir les caractères 4 -1 3 0 1 2 -2 5 en utilisant la fonction ***strtol***. Cette fonction a été privilégiée au détriment de ***atoi*** car il fallait être sûr que la conversion s'était bien déroulée.

Une fois que la récupération des arguments du tableau a été faite, le client se charge de créer son propre segment de mémoire partagée et de le projeter.

- Création et projection du segment de mémoire

  Le segment de mémoire portera comme le nom le pid du client sous la forme ***/pid***. Une fois créée, le segment a besoin d'une taille, qui sera défini par ***SHM_SIZE * sizeof(req.size)*** (avec SHM_SIZE = sizeof(int) et req.size la taille du tableau). Ce qui lui permet de stocker un tableau de req.size d'entier. Après que la taille du SHM ait été définie, on peut en faire une projection.

- Projection du segment

  Pour ce qui est de la projection en mémoire du segment, elle a été faite avec la fonction ***mmap*** et avec l'attribut ***PROT_WRITE*** pour pouvoir y écrire le contenu du tableau où se fera le calcul de somme préfixe. Et pour ce faire, j'ai utilisé le fonction ***memcpy*** comme ceci : ***memcpy(ptr_shm, &tab, SHM_SIZE * sizeof(req.size));***

Dés lors que la copie a été faite, je crée le tube nommé.

- Création du tube nommé et écriture de la requête

La création du tube nommé s'effectue avec la fonction ***mkfifo***. Le tube nommé crée, don ouverture doit se réaliser. Pour se faire, il faut utiliser la fonction ***open***. Ces deux actions ont permis de créer le tube nommé de nom ***tube***. Lorsque les deux actions précédentes sont faites, on peut enfin y écrire la requête comme suit : ***write(d, (request_t *) &req, sizeof(req)***

Fonctionnement du serveur

Pour ce qui est du serveur, lui, va ouvrir le tube nommé en mode lecture pour y lire la demande du client. Quand la lecture est finie, il va créer un processus avec la fonction ***fork***. C'est le fils qui va être à l'origine de la création des threads qui vont effectuer en parallèle le calcul de somme préfixe et c'est également à lui de renvoyer le résultat au client. Le père, lui, va se mettre en attente d'une autre requête à lire.

- Threads créer par le processus fils

  Comme dit plus haut, le fils se charge de créer les threads qui vont faire la somme. L'algorithme de Hillis-Steele Scan a été appliquée. Pour se faire, N-1 threads ont été crées, et, sont joins, chacun avec comme argument la structure de données vu plus haut ***struct thread_arg***. Comme cela, chaque thread aura les informations dédiées à un client en particulier.

  Chaque thread va ouvrir le segment de mémoire partagée en mode lecture et y faire le calcule. Une fois le calcule terminé, le processus fils le récupère grâce à la fonction ***pthread_join***.

- Envoie du résultat au client

  Le processus fils, envoie le résultat au client via le segment de mémoire partagée.

  Le fait que le client et le serveur ont accès tous les deux à la mémoire partagée, cela oblige à être vigilent et à en protéger l'accès.

Gestion des sémaphores

Pour gérer les accès au shm, j'utilise les sémaphores afin de bloquer un des acteurs quand l'un travaille sur la mémoire. 2 sémaphores ont été utilisés comme suit :

sem_t *S = 0 et sem_t *T = 0

Dans le serveur :

a) P(s)
b) lire le shm
c) écrire dans le shm
d) V(T)

Dans le client :

a) V(S)

b) écriture dans la mémoire

c) P(T)

d) lecture dans le hm

Difficultés rencontrées

1) La compréhension des algorithmes présentés était très compliquée, car je ne savais pas s'il fallait utiliser les deux, Hillis-Steele et Blelloch ou un des deux. Du coup j'ai opté pour celui de Hillis-Steele.

2) Au niveau du calcul de la somme, je n'ai pas réussi à afficher le bon résultat à la fin. Cela vient probablement du fait que j'utilise un tableau temporaire qui n'est pris en compte que dans la boucle du parcours du tableau

3) Du mal à comprendre l'utilisation des sémaphores pour la gestion de la mémoire partagé