TP4 : Appels système

CE TP DOIT ÊTRE RENDU.

Dans cette dernière partie, notre but est de code des nouvelles fonctions pour les appels système. Par exemple, pour la création de threads, pour endormir les processus pendant un certain temps, etc..

1 Création de threads

Dans un premier temps, vous pouvez ajouter un nouvel appel système (`SYSC Ri, Rj, SYSC_NEW_THREAD`) qui duplique le thread courant pour en créer un nouveau. Chez le père (l'appelant), le système renvoie le numéro du thread fils dans le registre `Ri` (et `AC`). Chez le fils, ce même registre est forcé à zéro (ainsi que `AC`). Le père et le fils continuent leur exécution à la première instruction qui suit l'appel au système :

```
 0 : SUB R0, R0, 0
 1 : SYSC R1, R1, SYSC_NEW_THREAD
 2 : IFGT R0, 10
 3 : code du fils
     . . . . . .
10 : code du pere
     . . . . . .
```

1.1 Modification case mémoire

Programmez un exemple dans lequel le fils incrémente le contenu d'une case mémoire et le père l'affiche sans la modifier. Vous aurez sans doute besoin d'une nouvelle instruction `STORE` :

```
instruction STORE Ri, Rj, k
    | AC = Rj + k
    | si (AC < 0) ou (AC >= SS) <erreur adressage>
    | mem[SB + AC] = Ri
    | AC = Ri
    | PC += 1
```

2 Endormir des threads

On se propose d'implanter l'appel système `SYSC Ri, Rj, SLEEP` qui va endormir le thread courant pendant `Ri` seconde(s). Pour ce faire, vous devez :

— ajouter un état endormi ;
— ajouter une date de réveil (voir `man 2 time`) ;
— endormir le thread courant ;
— faire en sorte de le réveiller.

2.1 Test SLEEP

Vous pouvez tester cette fonction en créant deux threads (le père et le fils). Le premier affiche toujours le même entier et le second un entier différent toutes les 3 secondes.

3 La fonction getchar

On se propose d'implanter l'appel système `SYSC Ri, Rj, GETCHAR` qui va lire un caractère sur l'entrée standard et le placer dans `Ri` ou attendre l'arrivée d'un caractère.

Nous ne pouvons pas réellement utiliser le clavier car cela impose de contrôler parfaitement les arrivées de caractères. Nous allons donc simuler la frappe au clavier en demandant au système de placer un caractère toutes les trois secondes dans le tampon.

Pour ce faire, vous devez :

— prévoir la définition du tampon (capacité un caractère) :

```
char tampon = '\0';    /* le '\0' indique le vide */
```

— ajouter un nouvel état `GETCHAR` (endormi en attente de caractère) et un compteur de processus dans cet état ;

— endormir le processus courant si le tampon est vide ;

— à l'arrivée d'un caractère, vous devez réveiller un processus qui serait dans un état `GETCHAR` ou stocker le caractère dans le tampon.

3.1 Lecture caractère

Vous pouvez tester cette fonction en créant un thread qui tente de lire un caractère toutes les secondes ou toutes les quatre secondes

4 Création de processus (question bonus)

On se propose d'implanter l'appel système `SYSC Ri, Rj, FORK` qui va créer un nouveau processus par recopie de la mémoire d'un processus père et recopie de l'état processeur du thread courant. Pour ce faire, vous devez :

— ajouter à votre système un pointeur vers la dernière case mémoire utilisée ;

— prévoir la duplication de la partition courante.

Partons du principe que toutes les zones mémoire **ont la même taille**. Comment gérer la fin d'un processus et la libération de sa mémoire ?

4.1 Création fils

Vous pouvez tester cette fonction en créant un processus qui va créer un processus fils toutes les quatre secondes. Ce processus fils réalise deux affichages et meurt au bout de 4 secondes.