

TP d'IE sur la sérialisation - Vincent Monot

Sérialisation

La méthode pour sérialiser les threads est la suivante :

Chaque thread du programme est "associé" à un mutex. Au démarrage du programme, les threads situés en début de ligne ont leurs mutex initialisés à 1. Ceux qui ne sont pas en début de ligne ont leurs mutex initialisés à 0.

Lorsqu'une tâche est finie, le programme vérifie si la deadline de la ligne a été atteinte.

- Si ce n'est pas le cas, le mutex de la tâche suivante sur la ligne est libéré.
- Si c'est le cas, le mutex du début de ligne est libéré

Déroulement du programme

Le fichier **threads.c** est le fichier contenant le programme principal (avec le fichier de headers correspondant **threads.h**). Les fichiers **tasks.c** et **tasks.h** contiennent les fonctions appelées par les threads et chargées à chaud par ceux-ci.

Lors du parsing du fichier, chaque nouvelle tâche à appeler est enregistrée dans un tableau de **TaskInfo** :

```
typedef struct {  
    // numéro de la tâche à exécuter (fonction appelée)  
    int task;  
    // numéro du thread (pour le tableau de mutex)  
    int threadNb;  
    // numéro du thread à exécuter ensuite  
    int nextThread;  
    // numéro de la ligne  
    int line;  
} TaskInfo;
```

Pour récupérer les deadlines à respecter, une structure **LineInfo** est utilisée de la manière suivante :

```
typedef struct {  
    // nombre de tâches dans la ligne  
    int taskNb;  
    // numéro de la première tâche sur la ligne  
    int firstTask;  
    // temps de début et de fin de la ligne  
    struct timespec start, end;  
    // deadline à atteindre  
    int deadline;  
} LineInfo;
```

Après avoir récupéré toutes les informations relatives aux tâches à exécuter, les threads sont lancés et c'est la fonction **startThread()** qui est donnée en tant que paramètre que **pthread_create()** :

```
// pthread_t* tasks: tableau des threads (1 thread pour 1 tâche)  
// TaskInfo* taskInfo: tableau des infos sur les tâches  
pthread_create(&tasks[i], NULL, startThread, &taskInfo[i]);
```

Le déroulement de la fonction `startThread()` est le suivant :

- Récupération des informations sur la tâche à effectuer (mutex et autres valeurs)
- Boucle principale (`while(1)`) :
- Attente de la libération (puis prise) du mutex avec `sem_wait`
- Début du timer si cette tâche est en début de ligne
- Appel de la fonction à exécuter
- Fin du timer.
 - Si l'échéance est dépassée, on libère le mutex avec `sem_post` de la tâche en début de ligne.
 - Sinon, on libère le mutex de la tâche suivante.

Utilisation du programme

Le fichier **taskList.txt** contient la liste des tâches à effectuer. Il est sous la forme :

```
TASK_NB:<nombre total de tâches>
LINE_NB:<nombre de lignes>
<nb de tâches sur la ligne>:<num. des fcts à appeler>-...-END-<échéance de la ligne>
```

Pour lancer le programme :

```
cd src
make
./threads.out
```

A noter que le programme est compilé avec les options `-Wall` et `-pedantic` .