Enoncé du TP 1

Printemps 2021

But du TP

Le but de ce TP est de se familiariser avec le langage de programmation D (on parle aussi de DLang) et d'écrire vos premiers programmes en langage D

(1) en analysant quelques programmes simples écrits en langage D,

(2) en écrivant des programmes en langage D pour des problèmes (relativement) simples sur des systèmes distribués ayant une topologie en anneau.

Le langage D a été choisi car il intègre la notion de concurrence qui permet de simuler un système distribué basé sur le passage de messages. Le langage D permet de générer des threads logiques qui fonctionnent de manière asynchrone entre eux et qui peuvent s'échanger des messages. Ainsi chaque thread peut être vu comme un processus qui peut échanger des messages avec d'autres processus (représentés par d'autres threads). Il est possible de définir des liens entre les threads, ce qui peut être vu comme les liens de communication entre les processus.

Des informations sur le langage D se trouvent sur les sites suivants :

- `http://dlang.org`
- `https://tour.dlang.org`
- `http://www.informit.com/articles/article.aspx?p=1609144 (généralités sur la concurrence)`
- `https://tour.dlang.org/tour/en/multithreading/message-passing (sur le passage de messages)`

Attention : l'objectif de ces TPs n'est pas que vous deveniez expert.e du langage D mais seulement que vous puissiez utiliser les fonctions permettant de programmer les algorithmes distribués considérés.

Utiliser le langage D en salle de TP

L'utilisation du langage D sur votre poste est très simple. Il suffit :

1. de télécharger le compilateur dmd
2. d'écrire votre programme avec votre éditeur de texte favori et de le sauvegarder dans un fichier portant l'extension `.d`
3. de compiler le programme dans un `xterm` avec la commande suivante :
   `dmd -of=nomExecutable nomDuPrograme.d`
4. d'exécuter l'exécutable généré (`./nomExecutable`)

1 Exercice - Prise en main du langage D (1 heure)

Question 1

1. Téléchargez le fichier basicSendReceive.d via `http://perso.ens-lyon.fr/isabelle.guerin-lassous/MIF12/TP1/basicSendReceive.d` (identifiants : MIF12 / 12M1AD).
2. Etudiez ce code et exécutez le.
3. Décrivez ci-dessous, en quelques lignes, ce que ce programme effectue et identifiez les fonctions importantes.

Question 2

En vous inspirant du programme précédent, écrivez un programme où le **processus fils**, c'est-à-dire le thread créé par le processus principal (processus associé à la fonction *main* appelé aussi **processus père**), acquitte, auprès du processus père, la bonne réception d'un nombre (n'importe quel entier) envoyé par ce dernier.

Aide : utilisez la variable **thisTid**, de type **Tid**, qui est l'ID du processus principal et qui peut être passée en variable dans la fonction **spawnedFunc**, ainsi que les fonctions **send** et **receive**.

Vous êtes donc maintenant en mesure de faire communiquer un processus père et son processus fils. Comment peut-on caractériser la communication mise en place entre ces deux processus dans cette question ?

Question 3

En vous inspirant des programme précédents, écrivez un programme où le processus fils va acquitter la bonne réception des données auprès du processus principal après la réception de trois messages du processus père.

Question 4

En vous inspirant des programme précédents, écrivez un programme où le processus père peut envoyer plusieurs valeurs à son processus fils et où le processus fils acquitte dès qu'il

reçoit la valeur 196. Le programme doit fonctionner quel que soit le nombre de valeurs envoyées avant la valeur 196.

2 Exercice - Système distribué en anneau (2 heures)

Il est maintenant temps de passer à des systèmes plus complexes. Dans cet exercice, on s'intéresse à un système distribué qui a une topologie en anneau unidirectionnel (le processus 0 a pour voisin le processus 1 qui a pour voisin le processus 2, ..., qui a pour voisin le processus $n-1$, qui a lui-même pour voisin le processus 0). Un processus de l'anneau est aussi appelé nœud par la suite.

Question 1

1. Téléchargez le fichier skeleton_forFirstTopology.d via `http://perso.ens-lyon.fr/isabelle.guerin-lassous/MIF12/TP1/skeleton_forFirstTopology.d` ;
2. Etudiez ce code ;
3. Décrivez ce que ce programme effectue et identifiez les fonctions importantes.

Question 2

Corrigez le programme skeleton_forFirstTopology.d, pour qu'il construise une topologie en anneau unidirectionnel entre les processus fils créés par le processus père de telle sorte que le nœud 0 a pour voisin dans l'anneau le nœud 1, le nœud 1 a pour voisin dans l'anneau le nœud 2, etc. Il est important de noter que le processus père n'appartient pas à la topologie en anneau. Il sert seulement à créer la topologie distribuée.

Question 3

Complétez le programme précédent de telle sorte que chaque nœud envoie son identifiant logique à son nœud voisin. Quelle est la plus grande taille d'anneau que votre programme est

capable de traiter ?

Question 4

Modifiez le programme afin que chaque nœud détermine le nombre de nœuds existants dans la topologie en anneau, sans passer par le processus père. Pour cela, l'algorithme sera le suivant :

- vous choisissez un nœud qui sera l'initiateur du comptage ;
- ce nœud envoie à son voisin un compteur avec la valeur 1 ;
- à chaque fois qu'un nœud (différent du nœud initiateur) reçoit le compteur, il l'incrémente de 1 et l'envoie à son voisin ;
- lorsque le nœud initiateur reçoit de nouveau le message, il connait le nombre de nœuds dans le système ;
- il informe alors tous les nœuds du système de la valeur obtenue.

Combien de messages sont échangés avec cette technique ?

Question 5

Ajoutez à votre programme une horloge de Lamport (horloge logique scalaire) associée à chaque nœud et vérifiez que ces horloges fonctionnent correctement sur vos programmes précédents (par exemple sur le programme associé à la question 3 et sur le programme associé à la question 4).

Question 6

Modifiez le programme précédent pour qu'il manipule les horloges vectorielles et vérifiez que ces horloges fonctionnent correctement sur vos programmes précédents (par exemple sur le programme associé à la question 3 et sur le programme associé à la question 4).

3 Exercice optionnel

Reprenez l'exercice précédent et modifiez le pour manipuler un anneau où les liens sont bidirectionnels.