Théorie et pratique de la concurrence – Master 1 Informatique

TP 1 : Rappel de threads en C

1 Threads en C

Un *thread* est un processus léger, c'est-à-dire un processus qui a sa pile d'exécution, son contexte d'exécution, et accès direct aux variables de sa portée.

En C, pour programmer avec des threads, il faut utiliser la librairie POSIX. En particulier pour ce TP il suffit de connaître la fonction *pthread_create*.

Dans les exercices ci-dessous on se propose d'observer le comportement concurrent de certains threads écrit en C. Pour pouvoir observer des exécutions différentes on pourra utiliser de façon intelligente la fonction *nanosleep*, qui arrête temporairement l'exécution d'un thread.

Exercice 1:

Dans une première version, on s'intéresse qu'à la concurrence, pour observer l'entrelacement. Écrire deux threads qui affichent de façon complètement indépendante les nombres de 1 à 100.

Observer le comportement de votre programme.

Exercice 2:

Dans une deuxième version, on s'intéresse également à observer une possible interférence. écrire deux threads qui incrémentent chacun à son tour une variable partagée, et qui ensuite affichent son contenu.

Observer le comportement de votre programme.

Exercice 3:

Dans cet exercice on programmera un producteur et plusieurs consommateurs, pour observer l'absence d'exclusion mutuelle et la famine. On producteur vérifie qu'un drapeau soit `false`,incrémente une variable partagée, met à `true` le drapeau et recommence du début. Chaque consommateur attend que le drapeau soit à `true`, ensuite il affiche son nom (passé en paramètre à la création) et le contenu de la variable, met le drapeau à `false` et recommence du début.

Observer le comportement de votre programme. Il faudra trouver un moyen d'observer l'absence d'exclusion mutuelle et la famine d'un des consommateur.

Exercice 4:

Dans cet exercice on programmera des threads pour observer l'interblocage. Les deux threads ont accès à deux drapeaux, `x1` et `x2`. Le premier thread attend que `x1` est `true`, et le mets à `false`, ensuite attend que `x2` soit `true` et le mets à `false`, et ensuite affiche un message, remet les drapeaux à `true` et il recommence. Le deuxième fait la même chose avec les deux drapeaux inversés.

Trouvez le moyen d'observer l'interblocage (livelock).