Licence 3 Informatique

Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 09

Programmation Système
Travaux pratiques n°1

Exercice 1 – Comparaison processus Unix / threads Posix

Préparation [TP1] : Structure (ou algorithme) de chaque version du programme demandé.
Rendu Moodle : Code associé à chaque version du programme demandé.
Date limite : Fin de la 1re séance de TP.
Code disponible sous Moodle : tp1_exo1-v1_base.c, tp1_exo1-v2_base.c

On veut créer N **activités parallèles** qui affichent un message personnalisé un certain nombre de fois avant de se terminer. Le nombre d'affichages à réaliser est **choisi** par l'utilisateur et doit être communiqué, ainsi que son rang de création, à l'activité parallèle lors de sa création.

Code disponible sous Moodle :

- tp1_exo1-v1_base.c : Squelette du programme de la version 1.
- tp1_exo1-v2_base.c : Squelette du programme de la version 2.

❖ **Version 1** : Les activités parallèles sont des **processus**.

- Écrire un programme dans lequel les activités parallèles ayant le comportement décrit au-dessus sont des processus.
- Le fichier rendu aura pour nom : Votre**NomPrenom**_tp1_exo1-v1.c

Exemple d'exécution :

%./tp1_exo1-v1 5 *[On crée 5 processus fils]*

Combien d'affichages du message ?

2

Je suis le processus de rang 4, mon identificateur est 2700
Je suis le processus de rang 2, mon identificateur est 2698
Je suis le processus de rang 3, mon identificateur est 2699
Je suis le processus de rang 3, mon identificateur est 2699
Je suis le processus de rang 2, mon identificateur est 2698
Je suis le processus de rang 4, mon identificateur est 2700
Je suis le processus de rang 1, mon identificateur est 2697
Je suis le processus de rang 0, mon identificateur est 2696
Je suis le processus de rang 1, mon identificateur est 2697
Je suis le processus de rang 0, mon identificateur est 2696

❖ **Version 2** : Les activités parallèles sont des **threads**.

- Écrire un programme dans lequel les activités parallèles ayant le comportement décrit au-dessus sont des threads (compagnons).
- Le fichier rendu aura pour nom : Votre**NomPrenom**_tp1_exo1-v2.c

Exemple d'exécution :

%./tp1_exo1-v2 5 *[On crée 5 threads compagnons]*

Combien d'affichages du message ?

1

Je suis le thread de rang 4, mon identificateur est 3042315072
Je suis le thread de rang 3, mon identificateur est 3050707776
Je suis le thread de rang 2, mon identificateur est 3059100480
Je suis le thread de rang 1, mon identificateur est 3067493184
Je suis le thread de rang 0, mon identificateur est 3075885888

Exercice 2 – Threads Posix : pratique du mécanisme exit/join

Préparation [TP1] : Structure (ou algorithme) du programme demandé.
Rendu Moodle : Code associé au programme demandé.
Date limite : Fin de la 1re séance de TP.

On veut que des threads compagnons **transmettent un compte-rendu d'exécution** au thread initial/principal lorsqu'ils se terminent. Ce compte-rendu sera une valeur choisie au hasard.

Le thread principal affiche chaque compte-rendu reçu accompagné de l'identité du thread lui ayant transmis cette information. Lorsque tous les threads compagnons sont terminés, le thread principal affiche la somme des comptes rendus reçus.

- Remarque : vous pouvez repartir de la version 2 de votre exercice 1.

❖ Écrire un programme dans lequel un thread principal et N threads compagnons se comportent comme décrit ci-dessus. Le nombre N de threads compagnons pourra être fixé arbitrairement ou constituer un paramètre de l'application.

- Le fichier rendu aura pour nom : VotreNomPrenom_tp1_exo2.c

Exemple d'exécution :

%./tp1_exo2 4 [Cas où l'on passe le nombre de threads compagnons en paramètre, ici 4]

Thread compagnon de rang 3, identifié par 3050527552, je mourrai en retournant 9
Thread compagnon de rang 2, identifié par 3058920256, je mourrai en retournant 8
Thread principal 3075708672 : valeur retournée par le thread 3050527552 = 9
Thread compagnon de rang 1, identifié par 3067312960, je mourrai en retournant 9
Thread principal 3075708672 : valeur retournée par le thread 3058920256 = 8
Thread compagnon de rang 0, identifié par 3075705664, je mourrai en retournant 1
Thread principal 3075708672 : valeur retournée par le thread 3075705664 = 1
Thread principal 3075708672 : valeur retournée par le thread 3067312960 = 9
Thread principal 3075708672 : somme des valeurs reçues = 27

Exercice 3 – Threads Posix : partage d'une variable

Préparation [TP1] : Structure (ou algorithme) du programme demandé.
Rendu Moodle : Code associé au programme demandé.
Date limite : 14/10/2019 - Minuit.

On veut que des threads compagnons **incrémentent une variable (partagée)** préalablement initialisée par le thread principal. Avant de se terminer, chaque thread compagnon augmente la variable partagée d'un incrément choisi au hasard puis fait part de cet incrément au thread principal (compte-rendu d'exécution).

Le thread principal affiche chaque compte-rendu reçu accompagné de l'identité du thread lui ayant transmis cette information. Lorsque tous les threads compagnons sont terminés, le thread principal affiche la valeur de la variable partagée.

- Remarques :
  - Vous pouvez repartir de votre exercice 2 dans lequel la communication du compte-rendu d'exécution est déjà mise en place.
  - Lors de cette séance, il n'est pas nécessaire d'assurer une synchronisation des accès à la variable partagée (but des TP 2 et 3) mais on rappelle – comme cela a été vu en cours – que cet accès partagé peut poser des problèmes de cohérence .

❖ Écrire un programme dans lequel un thread principal et N threads compagnons se comportent comme décrit ci-dessus. Le nombre N de threads compagnons pourra être fixé arbitrairement ou constituer un paramètre de l'application.

- Le fichier rendu aura pour nom : VotreNomPrenom_tp1_exo3.c

Exemple d'exécution :

%./tp1_exo3 3 *[Le paramètre de l'application spécifie que l'on crée 3 threads compagnons]*

Thread principal 3075708682 : J'initialise la valeur à 10
Thread compagnon de rang 2, identifié par 3059493696, j'ai ajouté 8 à la valeur qui vaut à présent 18
Thread compagnon de rang 1, identifié par 3067886400, j'ai ajouté 7 à la valeur qui vaut à présent 25
Thread compagnon de rang 0, identifié par 3076279104, j'ai ajouté 9 à la valeur qui vaut à présent 34
Thread principal 3075708682 : valeur retournée par le thread 3076279104 = 9
Thread principal 3075708682 : valeur retournée par le thread 3067886400 = 7
Thread principal 3075708682 : valeur retournée par le thread 3059493696 = 8
Thread principal 3075708682 : La valeur vaut 34

Pour aller plus loin…

❖ **Avec les processus**

On souhaite écrire une application comptant le nombre d'occurrences d'un caractère donné dans N fichiers donnés. Les noms de fichiers et le caractère recherché seront des paramètres de l'application. Le compte des occurrences sera réalisé, en **parallèle**, par N processus créés par le processus principal qui effectuera la somme et affichera le total lorsque tous les fils auront réalisé leur tâche.

Rappel : Les primitives nécessaires à la manipulation des fichiers (open, close, read, write) ont été étudiées en L2. Des rappels sont à votre disposition sous le répertoire Moodle « Prérequis ».

❖ **Avec les threads**

Coder complètement et exécuter les exercices 1 et 2 du cours-TD sur les threads (Traitement parallèle des éléments d'une matrice, traitement parallèle du contenu de N fichiers).