Licence Informatique

Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 09

SR2 – Partie « Systèmes » - Travaux pratiques n°3
Processus Unix – Threads Posix

Exercice 1 – Comparaison Processus / Threads

Écrire une application dans laquelle l'activité principale crée NA activités parallèles.

Chaque activité parallèle affiche un message l'identifiant NF fois avant de se terminer en retournant à son créateur son rang de création.

L'activité principale devra afficher les informations retournées au fur et à mesure que les activités parallèles se terminent.

L'application sera paramétrée par les valeurs de NA et de NF.

Attention : le nombre d'affichages sera fourni en paramètre de chaque thread (ce ne sera pas une variable partagée).

Version 1. Les activités parallèles sont des processus.

Exemples d'exécution : ./exo1v1 2 4

```
Activite rang 0 : identifiant = 139
Activite rang 0 : identifiant = 139
Activite rang 1 : identifiant = 140
Activite rang 0 : identifiant = 139
Activite rang 1 : identifiant = 140
Activite rang 0 : identifiant = 139
Activite rang 1 : identifiant = 140
Valeur retournee par le fils 139 = 0
Activite rang 1 : identifiant = 140
Valeur retournee par le fils 140 = 1
```

Version 2. Les activités parallèles sont des threads Posix.

Exemples d'exécution : ./exo1v2 2 5

```
Thread 0 : mon identificateur est 139689981839104
Thread 0 : mon identificateur est 139689981839104
Thread 0 : mon identificateur est 139689981839104
Thread 0 : mon identificateur est 139689981839104
Thread 0 : mon identificateur est 139689981839104
Thread 1 : mon identificateur est 139689973384960
Thread 1 : mon identificateur est 139689973384960
Thread 1 : mon identificateur est 139689973384960
Thread 1 : mon identificateur est 139689973384960
Thread 1 : mon identificateur est 139689973384960
Valeur retournee par le thread 139689981839104 = 0
Valeur retournee par le thread 139689973384960 = 1
```

Exercice 2 – Threads – Partage d'une variable

Écrire une application dans laquelle NT threads s'exécutent en parallèle et créditent ou débitent (d'un montant aléatoire) le solde d'un compte bancaire commun. Lors d'une opération de débit, si le solde est insuffisant, cela sera signalé mais l'opération de débit sera quand même réalisée.

L'application sera paramétrée par la valeur de NT, le solde initial du compte et le nombre de débits/crédits.

À l'exécution, faites varier le nombre de threads et de débits/crédits, voire temporisez un peu les opérations, que constatez-vous ?

Exemples d'exécution : ./exo2 2 0 2

```
Credit 140656714581760 : credit (9) => solde = 9
Credit 140656714581760 : credit (10) => solde = 12
Debit 140656706127616 : debit (7) => solde = 2
Debit 140656706127616 : debit (3) => solde = 9
Solde = 9
```

Exercice 3 – Threads – Partage d'un tableau géré circulairement

Écrire une application dans laquelle NP + NC threads s'exécutent en parallèle et partagent un tableau contenant des messages composés d'au plus 20 caractères et d'une valeur entière.

Les NP threads sont des « producteurs » de messages déposés dans le tableau partagé.

Les NC threads sont des « consommateurs » des messages qui se trouvent dans le tableau partagé.

Les dépôts se font dans l'ordre croissant des indices, de manière circulaire.

Les consommations se font dans l'ordre des dépôts.

L'application sera paramétrée le nombre NP de « producteurs, » celui NC de « consommateurs », le nombre de « dépôts » et de « consommations » et la taille du tableau partagé.

À l'exécution, faites varier les paramètres, voire temporisez un peu les threads, que constatez-vous ?

Exemples d'exécution : ./exo3 2 2 2 1 1

➔ 2 producteurs et 2 consommateurs déposant ou retirant chacun 2 messages, tableau de taille 1

```
        Prod 0 (140144848799488) : Message depose = Bonjour 1 de prod 0
                Conso 1 (140144823437056) : Message retire = Bonjour 1 de prod 0
        Prod 0 (140144848799488) : Message depose = Bonjour 2 de prod 0
                Conso 0 (140144831891200) : Message retire = Bonjour 2 de prod 0
        Prod 1 (140144840345344) : Message depose = Bonjour 1 de prod 1
        Prod 1 (140144840345344) : Message depose = Bonjour 2 de prod 1

Fin de l'execution du main
```