Ressource R5.B04

Gestion des threads sous Linux

Objectif :

Mise en œuvre de la bibliothèque de gestion des threads du système **Linux**.

Travail demandé et configuration :

Posséder une machine virtuelle hébergeant un système UNIX/Linux. Un éditeur de texte qui, de préférence, supporte la coloration syntaxique et la numérotation des lignes. Un compilateur C
Vous serez amenés au cours de ses TD/TP à effectuer beaucoup de recherches.

Exercice 1 :

Écrire un programme qui réalise le traitement suivant :

- Un processus (thread principal) crée 1 thread avec les attributs par défaut.
- Le thread créé affiche ses attributs.

Exercice 2 :

Écrire un programme qui réalise le traitement suivant :

- Un processus (**thread principal**) crée successivement **2 threads** avec les attributs par **défaut.**
- Le **premier thread** reçoit en paramètre **un nombre**.
- Le **second thread** reçoit en paramètre **une chaîne de caractères.**
- Le **premier thread** affichera sur la sortie standard son nom (« je suis le premier thread ») et son **TID** puis **le nombre** qui lui a été transmis en paramètre. En se terminant il retournera la valeur de ce nombre arrondie à l'entier le plus proche.
- Le **second thread** affichera sur la sortie standard son nom et son **TID** puis **la chaîne de caractères** qui lui a été transmise en paramètre. En se terminant il retournera cette chaîne convertie en majuscules**.**
- Après la création des 2 threads**,** le thread principal se mettra en attente de leur terminaison. Chaque fois qu'un thread se terminera, il affichera sur la sortie standard le **TID** et la **valeur retournée** par ce thread.

Exercice 3 :

Écrire un programme qui réalise le traitement suivant :

- Un processus (**thread principal**) présente un menu qui propose à l'utilisateur d'effectuer une opération arithmétique (+, -, *) appliquée à 2 opérandes.entiers qui auront été saisis. Ensuite le thread principal crée successivement **2 threads** avec les attributs par **défaut.**
- Le **premier thread** reçoit en paramètre **les 2 opérandes, un pointeur sur le résultat, un pointeur sur la fonction à exécuter.** .

- Le **second thread** reçoit en paramètre **les 2 opérandes et un pointeur sur la fonction à exécuter.**

- Après la création des 2 threads**,** le thread principal se mettra en attente de leur terminaison. Chaque fois qu'un thread se terminera, il affichera sur la sortie standard le **TID** du thread et la valeur de la variable modifiée dans le cas du thread1 ou la **valeur retournée** dans le cas du thread2.