Programmation des threads POSIX : Exercice n°1

Points de matière concernés :
- Lancement d'un thread avec et sans paramètre (pthread_create)
- Attente de la fin d'un thread avec récupération de sa valeur de retour (pthread_join)
- Observation du travail du scheduler

Avant de commencer, munissez-vous de 4 gros fichiers texte (entre 1500 et 2000 lignes).

Etape 1

A partir du thread principal, lancez un thread (sans paramètre) dont la tâche est de compter le nombre d'occurrences d'un mot (par exemple « printf » ou « cout ») dans un fichier texte qu'il ouvrira dans sa fonction. Le thread principal attendra la fin du thread et récupérera le résultat qu'il affichera.

Dans le but de mettre en évidence le parallélisme (à partir de l'étape 2) et le rôle du scheduler, on va délibérément ralentir l'exécution du thread. Pour cela, le thread, dans une boucle (i étant initialisé à 0),

1. **ouvre le fichier**,
2. se positionne sur le caractère de position i dans le fichier,
3. lit autant de caractères que nécessaire (ex : 6 si mot cible = « printf ») (attention à la fin de fichier !),
4. **ferme le fichier**,
5. teste si la chaîne lue correspond au mot cible et incrémente éventuellement un compteur.
6. incrémente i de **un**, remonte dans la boucle et cela jusqu'à la fin du fichier.

De plus, à chaque fois qu'il ouvrira le fichier le thread affichera « * » sur le terminal.
Consigne : on demande de travailler avec les fichiers non bufferisés.

Etape 2

Idem mais vous devez lancer 4 threads (sans paramètre) en parallèle sur 4 fonctions différentes. Les différences sont les fichiers à analyser, les mots recherchés et le nombre de tabulations (voir juste après). Dans cette étape, les noms de fichier et les mots recherchés sont « hard-codés » dans la fonction. A chaque ouverture d'un fichier, le thread 1 affichera une « * », le thread 2 affichera une tabulation puis « * », le thread 3 affichera deux tabulations puis « * », et le thread 4 affichera trois tabulations puis « * ». On obtiendra donc un résultat du type suivant

```
*
*
*
        *
        *
*
*
                *
```

```
                        *
                        *
                                    *
                                    *
                                    *
            *
            *
*
*
*
                        *
                        *
…
et ainsi de suite
```

A nouveau, le thread principal attend la fin des 4 threads secondaires avant d'afficher les résultats de recherche.

Etape 3

Idem mais il n'y a plus qu'une seule fonction pour les 4 threads secondaires. A leur lancement, le thread principal leur passe en paramètre l'adresse d'une structure contenant le nom du fichier à analyser, le mot recherché et le nombre de tabulations à afficher avant « * ».