TD 3 : Collecte et stockage de données

Spécifications du problème

Dans le cadre d'une application multithread développée sous Unix, on souhaite enregistrer dans un fichier les données mesurées par deux capteurs.

Une tâche sera dédiée à la lecture de chaque capteur : tâche capteur *i*, i ∈ [1;2].

Une tâche d'enregistrement est chargée de les écrire dans un fichier.

On ne s'occupera pas de la terminaison « propre » de l'application (elle sera à faire au TD suivant).

Tâches capteur

Pour pouvoir tester l'application indépendamment du matériel, nous remplacerons la lecture des capteurs par la lecture de fichiers de valeurs.

Les deux tâches capteurs écrivent les valeurs lues sous la forme de chaînes de caractères de longueur fixe dans une file d'attente.

Chaque tâche capteur effectuera en boucle les traitements suivants :

- lecture d'une valeur dans le fichier **`donnees`*`i`***
- écriture dans la file du message : « **`Capteur i : valeur`** »
- attente de 100 microsecondes

- ***`i`*** vaudra 1 ou 2 selon le capteur.
- Les fichiers **`donnees1`** et **`donnees2`** sont à récupérer sur e-campus.

(cf. doc page 2 pour la construction de la chaîne, l'attente et les accès fichiers)

Tâche d'enregistrement

La tâche d'enregistrement tourne en boucle infinie.

Elle crée un fichier au démarrage et y inscrit une à une les données lues dans la file d'attente.

Travail demandé

1 - Ecrivez l'algorithme de chaque tâche.

Méthode :

- faites d'abord un schéma des tâches et des zones de communication,
- écrivez votre algorithme sous forme de commentaires dans le fichier source, il sera facile d'insérer ensuite le code correspondant.

2 – Implémentez votre algorithme en C en utilisant les threads Posix.

Documentation

Construction d'une chaîne de caractère

Fonction `sprintf()` : cf doc TD1

Attente de 100 microsecondes

Fonction `usleep()` (déclarée dans `unistd.h`)

La fonction `usle`**`ep()`** permet d'attendre le nombre de microsecondes passées en paramètre.

Exemple

```
usleep(100); // Mise en attente de la tâche pendant 100 µs
```

Lecture et écriture dans un fichier

Dans la mesure où les informations manipulées sont des chaînes de caractères, il est pertinent d'utiliser les fonctions **`fopen(), fputs(), fgets(), fclose().`**
On notera que **`fgets()`** renvoie la valeur **`NULL`** quand la fin du fichier est atteinte.

Exemple lecture

```
FILE*fic ;
char buf[100]
// ouverture en lecture
if ((fic = fopen("toto", "r")) == NULL){
    perror("ouverture fichier toto");
    exit(-1);
  }
// lecture d'une ligne du fichier
fgets(buf, sizeof(buf), fic) ;

fclose(fic) ;
```

Exemple écriture

```
FILE*fic ;
char *buf = "Bonjour !";
// ouverture en ecriture
if ((fic = fopen("titi", "w")) == NULL){
    perror("ouverture fichier titi");
    exit(-1);
  }
// ecriture dans le fichier
fputs(buf, fic) ;
fclose(fic) ;
```