Feuille de TD3 : Threads & verrous

```
typedef struct
      {
        int variable;
      }
      var_struc;

void *ma_fonction(void *arg){
       var_struc *temp=(var_struc*)arg;
       printf("je suis un thread %d \n",temp->variable);
       temp->variable++;
       pthread_exit(NULL);
      }

int main(){
       var_struc t;
       t.variable=10;
      pthread_t my_thread1;
       int ret =pthread_create(&my_thread1, NULL, ma_fonction, (void*) &t);
      pthread_join(my_thread1,NULL);
}
```

Exercice 1 :

Ecrire un programme qui crée un thread prenant en paramètre une structure contenant un tableau d'entiers et l'affiche dans le terminal.

Exercice 2 :

Ecrire un programme qui utilise une structure contenant un entier initialisé à 10. Ce programme va lancer 4 threads les uns après les autres (sans attendre la fin du ou des précédents). Le premier thread ajoute 4 à la variable globale, le second divise par 2 (arrondit à l'inférieur), le troisième multiplie par 4, le dernier retranche 5. Lancer plusieurs fois ce programme. Vous devriez obtenir des valeurs différentes. (N'oubliez pas d'afficher les valeurs intermédiaires dans les threads)

Exercice 3 :

Ecrire un programme qui crée une structure contenant un tableau d'entiers, le nombre d'éléments du tableau et un entier *x*. Initialiser le tableau avec des 0. Votre programme lancera le premier thread qui changera les valeurs du tableau par des valeurs aléatoires comprises entre 1 et 5, puis votre programme lancera un thread pour afficher le contenu du tableau. Les deux threads doivent s'exécuter en parallèle.

Modifier votre code pour que l'affectation des cases du tableau attende une seconde entre chaque case affectée. Le second thread affiche le tableau toutes les 2 secondes.

```
pthread_mutex_t  mutex;//le verrou variable globale

pthread_mutex_init(&mutex,NULL); //initialisation du mutex dans le main

pthread_mutex_lock (& mutex);
pthread_mutex_unlock (& mutex);
```

Exercice 4 : Protection du tableau

Modifier l'exercice 3 afin de vous assurer que chaque thread accède seul à une case du tableau.

Exercice 5 : protection d'accès

Proposer un programme qui remplit un tableau (dans une structure) avec des valeurs aléatoires et l'affiche. Votre programme exécute *n* threads, chaque thread doit avoir un numéro unique. Toutes les secondes chaque thread affiche son numéro, une position aléatoire dans le tableau et le contenu de la case.

Exercice 6 : Tri fusion en parallèle

Proposer un algorithme permettant de trier votre tableau en parallèle. Demander un entier *k,* diviser le tableau en *k* tableaux de même taille (attention à ne pas dupliquer le tableau, utiliser les indices du tableau), lancer un thread par tableau afin de les trier en parallèle puis fusionner les tableaux triés.