Utilisation de Verrou en C

Le type abstrait est implémenté en C par le type `pthread_mutex_t` de la bibliothèque `pthread`. Ainsi pour l'utiliser on veillera à indiquer `include <pthread.h>` en tête du fichier `main.c`, et à le compiler avec l'option `-pthread` : `gcc -pthread main.c -o main`.

Une fois un verrou `v` déclaré, on garantit l'exclusion mutuelle entre les sections de codes délimitées par un appel à `pthread_mutex_lock(&v);` et un appel à `pthread_mutex_unlock(&v)`. Cependant, le verrou `v` doit être initialisé dans la fonction `main` par un appel à `pthread_mutex_init(&v, NULL);` avant les appels à `pthread_mutex_lock` et `pthread_mutex_unlock`. Ainsi pour utiliser des verrous en C on réalise les étapes suivantes.

- Déclarer autant de verrous que nécessaire.
- Définir des structures pour passer aux tâches leurs arguments.
- Définir les tâches à faire, en protégeant leur sections critiques avec les verrous.
- Dans la fonction `main` :
  - initialiser chaque verrou par un appel à `pthread_mutex_init(&v, NULL);` ;
  - initialiser les variables qui seront partagées par les fils ;
  - initialiser autant de structures que besoin ;
  - déclarer puis lancer les fils en attribuant à chacun sa tâche et ses arguments ;
  - attendre la fin de tous les fils pour récupérer le résultat.

Exemple 1. On donne ci-dessous une exemple avec trois fils d'exécution devant réaliser la même tâche, à savoir incrémenter un compteur partagé un nombre donné de fois. On définit donc une seule tâche (la fonction `add_one`, et une seule structure pour passer à cette fonction ses arguments (la structure `args`). Afin de mettre en exclusion mutuelle les incrémentations du compteur faites par chaque fil réalisant `add_one`, on protège la ligne 19 grâce à un verrou déclaré au préalable (ligne 5).

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4
5  pthread_mutex_t verrou;
6
7  struct args_s {
8    int* cpt; //adr. du compteur partagé
9    int nbt; //nb de tours souhaités
10 };
11 typedef struct args_s args;
12
13 void* add_one(void* arg) {
14   //hyp : arg est de type args*
15   //incr. arg->nbt fois *(arg->cpt)
16   args* a = (args*) arg;
17   for (int i = 0; i < a->nbt ; i++) {
18     pthread_mutex_lock(&verrou);
19     *(a->cpt)= *(a->cpt) + 1;
20     pthread_mutex_unlock(&verrou);
21   }
22   return NULL;
23 }
```

```
30 int main(int argc, char* argv[]) {
31
32   int cpt = 0; // variable partagée
33   args a1 = {.cpt = &cpt, .nbt = 10000};
34
35   pthread_mutex_init(&verrou, NULL);
36
37   pthread_t p1, p2, p3;
38   pthread_create(&p1, NULL, add_one, &a1);
39   pthread_create(&p2, NULL, add_one, &a1);
40   pthread_create(&p3, NULL, add_one, &a1);
41
42   pthread_join(p1, NULL);
43   pthread_join(p2, NULL);
44   pthread_join(p3, NULL);
45
46   printf(" [cpt: %d]\n [obj: %d]\n",
     ↪  *(a1.cpt), 3*a1.nbt);
47
48   return 0;
49 }
```