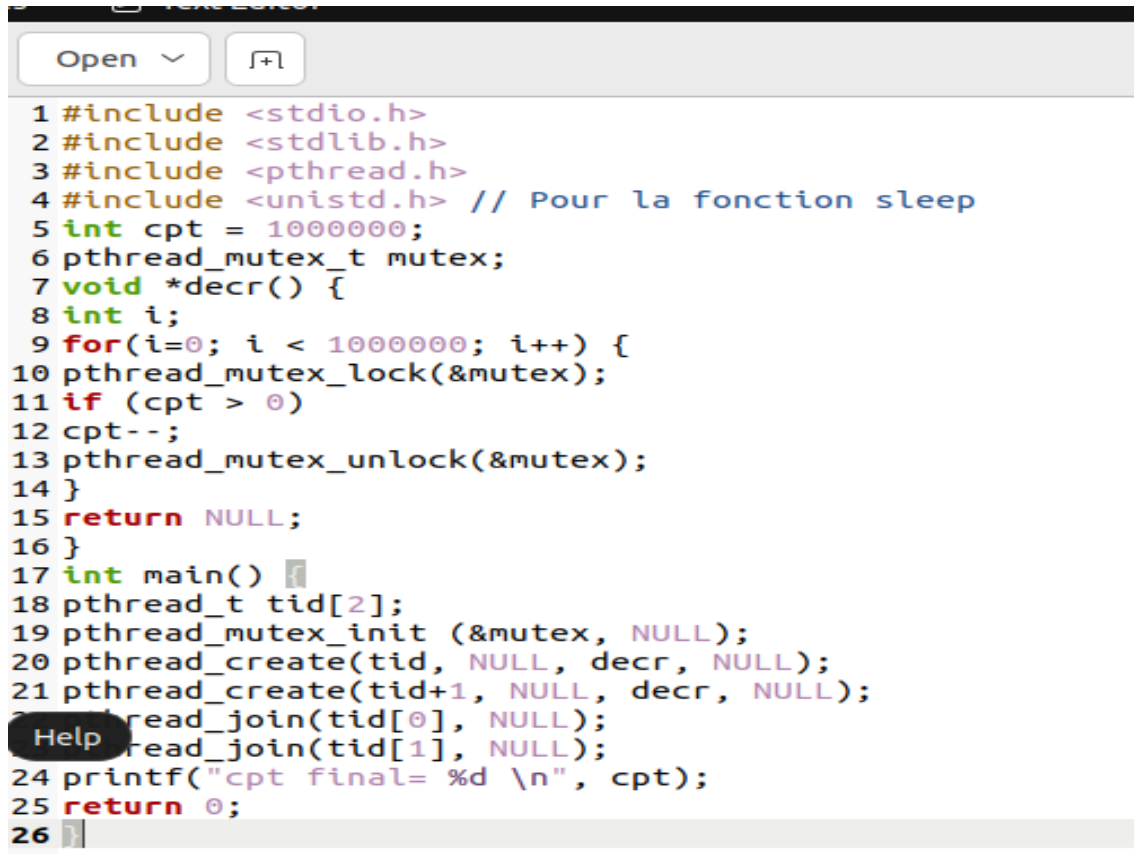
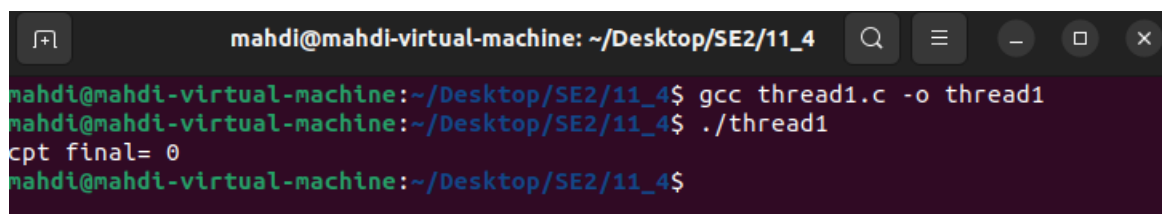


# Rendu Séance 11/04

1)



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h> // Pour la fonction sleep
5 int cpt = 1000000;
6 pthread_mutex_t mutex;
7 void *decr() {
8     int i;
9     for(i=0; i < 1000000; i++) {
10 pthread_mutex_lock(&mutex);
11 if (cpt > 0)
12 cpt--;
13 pthread_mutex_unlock(&mutex);
14 }
15 return NULL;
16 }
17 int main() {
18 pthread_t tid[2];
19 pthread_mutex_init (&mutex, NULL);
20 pthread_create(tid, NULL, decr, NULL);
21 pthread_create(tid+1, NULL, decr, NULL);
22 pthread_join(tid[0], NULL);
23 pthread_join(tid[1], NULL);
24 printf("cpt final= %d \n", cpt);
25 return 0;
26 }
```



```
mahdi@mahdi-virtual-machine: ~/Desktop/SE2/11_4
mahdi@mahdi-virtual-machine:~/Desktop/SE2/11_4$ gcc thread1.c -o thread1
mahdi@mahdi-virtual-machine:~/Desktop/SE2/11_4$ ./thread1
cpt final= 0
mahdi@mahdi-virtual-machine:~/Desktop/SE2/11_4$
```

# Application

```
thread1.c

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <unistd.h> // Pour la fonction sleep
5 static pthread_mutex_t my_mutex; // Déclaration du mutex
6 static int tab[5]; // Tableau partagé
7 // Fonction pour lire le tableau
8 void *read_tab_process(void *arg) {
9     pthread_mutex_lock(&my_mutex); // Verrouiller le mutex
10    for (int i = 0; i < 5; i++) {
11        printf("read_process, tab[%d] vaut %d\n", i, tab[i]);
12        sleep(2);
13    }
14    pthread_mutex_unlock(&my_mutex); // Déverrouiller le mutex
15    pthread_exit(NULL); // Terminer le thread
16 }
17 // Fonction pour écrire dans le tableau
18 void *write_tab_process(void *arg) {
19    pthread_mutex_lock(&my_mutex); // Verrouiller le mutex
20    for (int i = 0; i < 5; i++) {
21        tab[i] = 2 * i; // Remplir le tableau avec des valeurs
22        printf("write_process, tab[%d] vaut %d\n", i, tab[i]);
23        sleep(1); // Relentit le thread d'écriture
24    }
25    pthread_mutex_unlock(&my_mutex); // Déverrouiller le mutex
26    pthread_exit(NULL); // Terminer le thread
27 }
28 int main(int ac, char **av) { // Ajout de int au prototype de main
29    pthread_t th1, th2; // Déclaration des threads
30    void *ret;
31    // Initialiser le mutex
32    if (pthread_mutex_init(&my_mutex, NULL) != 0) {
33        fprintf(stderr, "Erreur d'initialisation du mutex\n");
34        exit(1);
35    }
36    // Créer le thread d'écriture
37    if (pthread_create(&th1, NULL, write_tab_process, NULL) != 0) {
38        fprintf(stderr, "Erreur lors de la création du thread 1\n");
39        exit(1);
40    }
41    // Créer le thread de lecture
42    if (pthread_create(&th2, NULL, read_tab_process, NULL) != 0) {
43        fprintf(stderr, "Erreur lors de la création du thread 2\n");
44        exit(1);
45    }
46    // Attendre la fin des threads
47    pthread_join(th1, &ret);
48    pthread_join(th2, &ret);
49    // Détruire le mutex après utilisation
50    pthread_mutex_destroy(&my_mutex);
51    return 0; // Retourner 0 pour indiquer le succès
52 }
```

```
mahdi@mahdi-virtual-machine:~/Desktop/SE2/11_4$ gcc thread2.c -o thread2
mahdi@mahdi-virtual-machine:~/Desktop/SE2/11_4$ ./thread2
write_process, tab[0] vaut 0
write_process, tab[1] vaut 2
write_process, tab[2] vaut 4
write_process, tab[3] vaut 6
write_process, tab[4] vaut 8
read_process, tab[0] vaut 0
read_process, tab[1] vaut 2
read_process, tab[2] vaut 4
read_process, tab[3] vaut 6
read_process, tab[4] vaut 8
mahdi@mahdi-virtual-machine:~/Desktop/SE2/11_4$
```