**DAOUADJI ABDERRAHMANE G3**

**EXO 1 TP2 partie 1:**

**Éxécution:**

**Thread 0 appelle sem\_wait(p)**

**Thread 0: i = 0**

**Thread 1 appelle sem\_wait(p)**

**Thread 0 appelle sem\_post(v)**

**Thread 0 appelle sem\_wait(p)**

**Thread 0: i = 1**

**Thread 0 appelle sem\_post(v)**

**Thread 0 appelle sem\_wait(p)**

**Thread 0: i = 2**

**Thread 0 appelle sem\_post(v)**

**Thread 0 appelle sem\_wait(p)**

**Thread 0: i = 3**

**Thread 0 appelle sem\_post(v)**

**Thread 0 appelle sem\_wait(p)**

**Thread 0: i = 4**

**Thread 0 appelle sem\_post(v)**

**Thread 1: i = 0**

**Thread 1 appelle sem\_post(v)**

**Thread 1 appelle sem\_wait(p)**

**Thread 1: i = 1**

**Thread 1 appelle sem\_post(v)**

**Thread 1 appelle sem\_wait(p)**

**Thread 1: i = 2**

**Thread 1 appelle sem\_post(v)**

**Thread 1 appelle sem\_wait(p)**

**Thread 1: i = 3**

**Thread 1 appelle sem\_post(v)**

**Thread 1 appelle sem\_wait(p)**

**Thread 1: i = 4**

**Thread 1 appelle sem\_post(v)**

**\_ Si on supprime sleep(1) dans la section critique :** **Les affichages des threads sont toujours bien ordonnés, car les sémaphores sont actifs (sem\_wait / sem\_post).**

**EXO 1 TP 2 partie 2:**

**1)**

**int main() {**

**pthread\_t threads[nb\_membre];**

**int ids[nb\_membre];**

**sem\_init(&mutex, 0, 1);**

**for (int i = 0; i < nb\_membre; i++) {**

**id[i] = i;**

**pthread\_create(&threads[i], NULL, membre, &id[i]);**

**}**

**for (int i = 0; i < nb\_membre; i++) {**

**pthread\_join(threads[i], NULL);**

**}**

**sem\_destroy(&mutex);**

**printf("Solde final = %d DA", compte);**

**return 0;**

**}**

**2) protection de compte:**

**le thread demande l’accès avec sem\_wait(&mutex);Il effectue son opération puis il libère l’accès avec sem\_post(&mutex);**

**3)**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <pthread.h>**

**#include <semaphore.h>**

**#include <unistd.h>**

**int nb\_membre = 8;**

**int nb\_mois = 100;**

**int compte = 0;**

**int priorité = 0;**

**sem\_t mutex;**

**sem\_t consulté;**

**void\* membre(void\* arg) {**

**int id = \*(int\*)arg;**

**for (int mois = 1; mois <= nb\_mois; mois++) {**

**int operation = rand() % 3;**

**if (operation == 0 || operation == 1) {**

**sem\_wait(&mutex);**

**priorité++;**

**sem\_post(&mutex);**

**sem\_wait(&consulté);**

**sem\_wait(&mutex);**

**if (operation == 0) {**

**int montant = (rand() % 100) + 1;**

**compte += montant;**

**printf("[membre %d][mois %d] Dépôt de %d DA | Solde = %d ", id, mois, montant, compte);**

**} else {**

**int montant = (rand() % 50) + 1;**

**if (compte >= montant) {**

**compte -= montant;**

**printf("[membre %d][mois %d] Retrait de %d DA | Solde = %d ", id, mois, montant, compte);**

**} else {**

**printf("[membre %d][mois %d] Retrait refusé (solde insuffisant: %d DA ", id, mois, compte);**

**}**

**}**

**sem\_post(&mutex);**

**sem\_wait(&mutex);**

**priorité--;**

**if (priorité == 0)**

**sem\_post(&consulté);**

**else**

**sem\_post(&consulté);**

**sem\_post(&mutex);**

**} else {**

**sem\_wait(&consulté);**

**sem\_post(&consulté);**

**sem\_wait(&mutex);**

**printf("[membre %d][mois %d] Consultation : Solde = %d ", id, mois, compte);**

**sem\_post(&mutex);**

**}**

**usleep(1000);**

**}**

**pthread\_exit(NULL);**

**}**

**int main() {**

**pthread\_t threads[nb\_membre];**

**int id[nb\_membre];**

**sem\_init(&mutex, 0, 1);**

**sem\_init(&consulté, 0, 1);**

**for (int i = 0; i < nb\_membre; i++) {**

**id[i] = i;**

**pthread\_create(&threads[i], NULL, membre, &ids[i]);**

**}**

**for (int i = 0; i < nb\_membre; i++) {**

**pthread\_join(threads[i], NULL);**

**}**

**sem\_destroy(&mutex);**

**sem\_destroy(&consulté);**

**printf("Solde final = %d ", compte);**

**return 0;}**