#include <stdio.h>

#include <stdlib.h>

#include <semaphore.h>

#include <pthread.h>

*sem\_t* mutex1;//CRIACAO DA VARIAVEL 1 DO TIPO SEMAFORO

*sem\_t* mutex2;//CRIACAO DA VARIAVEL 2 DO TIPO SEMAFORO

void \*thread\_1(void \**arg*){

while(1){

printf("ID semaforo = %lu, ID thread = %ld\n", (*size\_t*)&mutex1, pthread\_self());//IMPRIME INFORMACOES DO SEMAFORO E DA THREAD EM EXECUCAO

sem\_wait(&mutex1);// FUNCAO PRINCIPAL DE VERIFICACAO DA OCORRENCIA DE CICLO --> DEADLOCK

printf("Execucao Thread 1\n");

sem\_wait(&mutex2); // FUNCAO PRINCIPAL DE VERIFICACAO DA OCORRENCIA DE CICLO --> DEADLOCK

sem\_post(&mutex1); // FUNCAO QUE INCREMENTA (DESBLOQUEIA) O SEMAFORO mutex1

sem\_post(&mutex2); // FUNCAO QUE INCREMENTA (DESBLOQUEIA) O SEMAFORO mutex2

}

pthread\_exit(NULL);// FUNCAO QUE TERMINA O THREAD DE CHAMADA

}

void \*thread\_2(void \**arg*){

while(1){

printf("ID semaforo = %lu, ID thread = %ld\n", (*size\_t*)&mutex2, pthread\_self());//IMPRIME INFORMACOES DO SEMAFORO E DA THREAD EM EXECUCAO

sem\_wait(&mutex2);// FUNCAO PRINCIPAL DE VERIFICACAO DA OCORRENCIA DE CICLO --> DEADLOCK

printf("Execucao Thread 2\n");

sem\_wait(&mutex1);// FUNCAO PRINCIPAL DE VERIFICACAO DA OCORRENCIA DE CICLO --> DEADLOCK

sem\_post(&mutex1);// FUNCAO QUE INCREMENTA (DESBLOQUEIA) O SEMAFORO mutex1

sem\_post(&mutex2);// FUNCAO QUE INCREMENTA (DESBLOQUEIA) O SEMAFORO mutex2

}

pthread\_exit(NULL);// FUNCAO QUE TERMINA O THREAD DE CHAMADA

}

int main(void) {

*pthread\_t* threads[2];// CRIA VETOR DE THREADS, COM DUAS POSICOES

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("/\*\*\*UNIFESP - Instituto de Ciência e Tecnologia\*\*\*/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2020-1ºSEMESTRE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*Sistemas Operacionais\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("/\*\*\*\*\*\*Daiana Kathrin Santana Santos 120.357\*\*\*\*\*\*/\n");

printf("/\*\*\*\*\*\*Isadora Rosa de Freitas Muniz 120.431\*\*\*\*\*\*/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n\n\n");

sem\_init(&mutex1, 0, 1);// FUNCAO QUE INICIALIZA O SEMAFORO mutex1, COM VALOR INICIAL 1.O VALOR 0 DO SEGUNDO ARGUMENTO DA FUNCAO PERMITE O COMPARTILHAMENTO ENTRE OS THREADS DE UM PROCESSO, E DEVEM ESTAR LOCALIZADOS EM ALGUM ENDEREÇO QUE SEJA VISIVEL PARA TODOS OS TOPICOS (VARIAVEL GLOBAL OU UMA VARIAVEL ALOCADA DINAMICAMENTE NA PILHA)

sem\_init(&mutex2, 0, 1);// FUNCAO QUE INICIALIZA O SEMAFORO mutex2, COM VALOR INICIAL 1.

pthread\_create(&(threads[0]), NULL, thread\_1, NULL);//CRIA A THREAD NA POSICAO 0 DO VETOR DE THREADS

pthread\_create(&(threads[1]), NULL, thread\_2, NULL);//CRIA A THREAD NA POSICAO 1 DO VETOR DE THREADS

pthread\_join(threads[0], NULL);// ESPERA PELO THREAD NA POSICAO 0 TERMINAR

pthread\_join(threads[1], NULL);// ESPERA PELO THREAD NA POSICAO 1 TERMINAR

return 0;

}