L:EIC / SO2122:
Pthreads e Concorrência
(usando a API do POSIX)

Q1. Considere o seguinte programa que cria dois threads a partir do thread principal de um processo e os usa para incrementar um contador. Compile-o e execute-o. Leia o código com atenção.

```
#include <stdio.h>
#include <pthread.h>

int count=0;

void * inc(void * arg)
{
  for (int i=0; i<1000000; i++) count++;
  return NULL;
}

int main()
{
  printf("Start: %d\n",count);

  pthread_t tid1, tid2;
  pthread_create(&tid1,NULL,inc,NULL);
  pthread_create(&tid2,NULL,inc,NULL);
  pthread_join(tid1,NULL);
  pthread_join(tid2,NULL);


  printf("End: %d\n",count);
}
```

Pense por que motivo o resultado pode variar e não ser o esperado.

Q2. O seguinte exemplo mostra como obter exclusão mútua nos incrementos usados no programa anterior e garantir a obtenção de um resultado correcto. Contudo cada thread acaba por executar todo o ciclo em bloco. Altere o programa de modo a fazer uma exclusão mútua mais fina. Compare os tempos de execução de ambas abordagens.

```
#include <stdio.h>
#include <pthread.h>

int count=0;
pthread_mutex_t lock;

void * inc(void * arg)
{
  pthread_mutex_lock(&lock);
  for (int i=0; i<1000000; i++) count++;
  pthread_mutex_unlock(&lock);
  return NULL;
}

int main()
{
  printf("Start: %d\n",count);

  pthread_mutex_init(&lock,NULL);

  pthread_t tid1, tid2;
  pthread_create(&tid1,NULL,inc,NULL);
  pthread_create(&tid2,NULL,inc,NULL);
  pthread_join(tid1,NULL);
  pthread_join(tid2,NULL);

  pthread_mutex_destroy(&lock);

  printf("End: %d\n",count);
}
```

Modifique novamente o programa de modo a um thread incrementar e outro decrementar o contador. O resultado final deve ser 0 se forem feitos tantos incrementos como decrementos. Acrescente código para imprimir um '.' sempre que o contador esteja negativo.

Q3. O seguinte programa impede que ocorram valores negativos no contador. Modifique-o de modo a este continuar a funcionar correctamente mas serem feitos menos signals na variável de condição.

```
#include <unistd.h>
#include <stdio.h>
#include <pthread.h>

int count=0;
pthread_mutex_t lock;
pthread_cond_t not_zero;

void * inc(void * arg)
{
  for (int i=0; i<1000000; i++)
  {
    pthread_mutex_lock(&lock);
    count++;
    pthread_cond_signal(&not_zero);
    pthread_mutex_unlock(&lock);
  }
  return NULL;
}

void * dec(void * arg)
{
  for (int i=0; i<1000000; i++)
  {
    pthread_mutex_lock(&lock);
    while (count == 0)
      pthread_cond_wait(&not_zero, &lock);
    count--;
    if (count<0) write(1,".",1);
    pthread_mutex_unlock(&lock);
  }
  return NULL;
}


int main()
{
  printf("Start: %d\n",count);

  pthread_mutex_init(&lock,NULL);
  pthread_cond_init(&not_zero,NULL);

  pthread_t tid1, tid2;
```

```
  pthread_create(&tid1,NULL,inc,NULL);
  pthread_create(&tid2,NULL,dec,NULL);
  pthread_join(tid1,NULL);
  pthread_join(tid2,NULL);

  pthread_mutex_destroy(&lock);

  printf("End: %d\n",count);
}
```