



Universidade Federal de Santa Catarina

Centro de Engenharias da Mobilidade
Curso de Engenharia Mecatrônica

Impasses

Disciplina: Sistemas Operacionais

Aluno: Maykon Hopka

Joinville – 2025

1 Exercícios

Exercício 1

Analisar e apontar quais são as regiões críticas do código sujeitas a condição de corrida.

Solução 1.1: Exercício 1

A região crítica ocorre dentro da função `ThreadAdd`, especificamente nas instruções que leem, incrementam e escrevem o valor da variável global `count`:

- `tmp = count;`
- `tmp = tmp + 1;`
- `count = tmp;`

Exercício 2

Mostrar que o resultado da execução é sujeito a condição de corrida.

Solução 1.2: Exercício 2

Executando o programa original (sem uso de mutex), é possível observar que o valor final de `count` nem sempre atinge o esperado `2 * NITER`. Um exemplo de saída:

```
$ ./sem_mutex  
BOOM! count is [1724351217], should be 2000000000
```

Isso demonstra que houve ****perda de incremento**** devido à interferência entre as duas threads que acessam `count` simultaneamente.

Exercício 3

Implementar a técnica de sincronização apropriada para resolver a condição de corrida.

```

1  #include <pthread.h>
2  #include <semaphore.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  #define NITER 1000000000
7
8  long count = 0;
9  pthread_mutex_t mutex; // Declaração do mutex
10
11 void * ThreadAdd(void * a)
12 {
13     int i;
14     long tmp;
15     for(i = 0; i < NITER; i++)
16     {
17         pthread_mutex_lock(&mutex); // Bloqueia o mutex
18
19         tmp = count; /* copy the global count locally */
20         tmp = tmp+1; /* increment the local copy */
21         count = tmp; /* store the local value into the global count */
22
23         pthread_mutex_unlock(&mutex); // Libera o mutex
24     }
25 }
26
27 int main(int argc, char * argv[])
28 {
29     pthread_t tid1, tid2;
30
31     // Inicializa o mutex
32     if (pthread_mutex_init(&mutex, NULL)) {
33         printf("\n ERROR initializing mutex");
34         exit(1);
35     }
36
37     if(pthread_create(&tid1, NULL, ThreadAdd, NULL))
38     {
39         printf("\n ERROR creating thread 1");
40         exit(1);
41     }
42
43     if(pthread_create(&tid2, NULL, ThreadAdd, NULL))
44     {
45         printf("\n ERROR creating thread 2");
46         exit(1);
47     }
48
49     if(pthread_join(tid1, NULL)) /* wait for the thread 1 to finish */
50     {
51         printf("\n ERROR joining thread");
52         exit(1);
53     }
54
55     if(pthread_join(tid2, NULL)) /* wait for the thread 2 to finish */
56     {
57         printf("\n ERROR joining thread");
58         exit(1);
59     }
60
61     // Destroi o mutex
62     pthread_mutex_destroy(&mutex);
63
64     if (count < 2 * NITER)
65         printf("\n BOOM! count is [%ld], should be %ld\n", count, (long)2*NITER);
66     else
67         printf("\n OK! count is [%ld]\n", count);
68
69     pthread_exit(NULL);
70 }

```

Figura 1: código com mutex para evitar condição de corrida