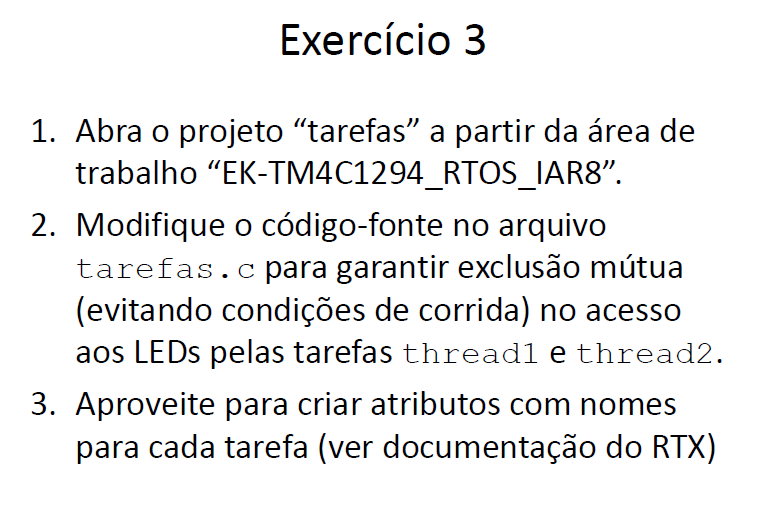
Exercícios S05 Sistemas Embarcados

Francielle de Lima Brozoski



#include "system\_tm4c1294.h" // CMSIS-Core

#include "driverleds.h" // device drivers

#include "cmsis\_os2.h" // CMSIS-RTOS

osThreadId\_t thread1\_id, thread2\_id;

osMutexId\_t mutex1\_id;

osMutexId\_t mutex2\_id;

void thread1(void \*arg){

uint8\_t state = 0;

uint32\_t tick;

while(1){

osMutexAcquire(mutex1\_id, osWaitForever);

tick = osKernelGetTickCount();

state ^= LED1;

osMutexAcquire(mutex2\_id, osWaitForever);

LEDWrite(LED1, state);

osMutexRelease(mutex2\_id);

osDelayUntil(tick + 100);

osMutexRelease(mutex1\_id);

} // while

} // thread1

void thread2(void \*arg){

uint8\_t state = 0;

uint32\_t tick;

while(1){

osMutexAcquire(mutex2\_id, osWaitForever);

tick = osKernelGetTickCount();

state ^= LED2;

osMutexAcquire(mutex1\_id, osWaitForever);

LEDWrite(LED2, state);

osMutexRelease(mutex1\_id);

osDelayUntil(tick + 101);

osMutexRelease(mutex2\_id);

} // while

} // thread2

void main(void){

LEDInit(LED2 | LED1);

osKernelInitialize();

thread1\_id = osThreadNew(thread1, NULL, NULL);

thread2\_id = osThreadNew(thread2, NULL, NULL);

mutex1\_id = osMutexNew(NULL);

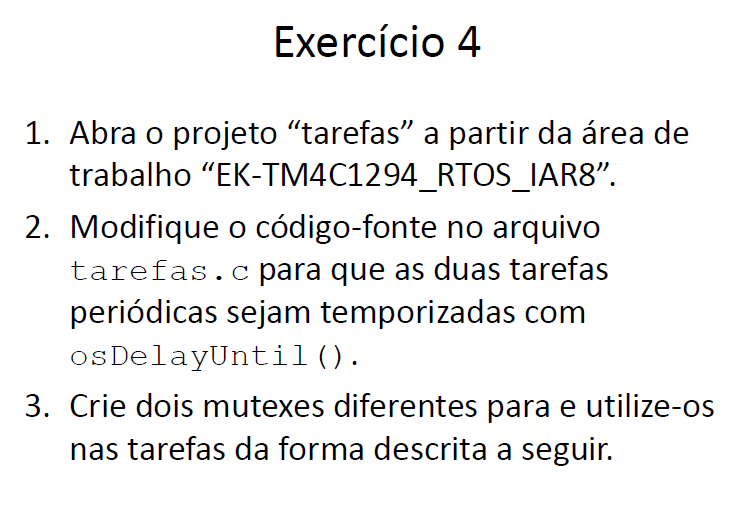
mutex2\_id = osMutexNew(NULL);

if(osKernelGetState() == osKernelReady)

osKernelStart();

while(1);

} // main



#include "system\_tm4c1294.h" // CMSIS-Core

#include "driverleds.h" // device drivers

#include "cmsis\_os2.h" // CMSIS-RTOS

osThreadId\_t thread1\_id, thread2\_id;

osMutexId\_t mutex1\_id;

osMutexId\_t mutex2\_id;

void thread1(void \*arg){

uint8\_t state = 0;

uint32\_t tick;

while(1){

osMutexAcquire(mutex1\_id, osWaitForever);

tick = osKernelGetTickCount();

state ^= LED1;

osMutexAcquire(mutex2\_id, osWaitForever);

LEDWrite(LED1, state);

osMutexRelease(mutex2\_id);

osDelayUntil(tick + 100);

osMutexRelease(mutex1\_id);

} // while

} // thread1

void thread2(void \*arg){

uint8\_t state = 0;

uint32\_t tick;

while(1){

osMutexAcquire(mutex2\_id, osWaitForever);

tick = osKernelGetTickCount();

state ^= LED2;

osMutexAcquire(mutex1\_id, osWaitForever);

LEDWrite(LED2, state);

osMutexRelease(mutex1\_id);

osDelayUntil(tick + 101);

osMutexRelease(mutex2\_id);

} // while

} // thread2

void main(void){

LEDInit(LED2 | LED1);

osKernelInitialize();

thread1\_id = osThreadNew(thread1, NULL, NULL);

thread2\_id = osThreadNew(thread2, NULL, NULL);

mutex1\_id = osMutexNew(NULL);

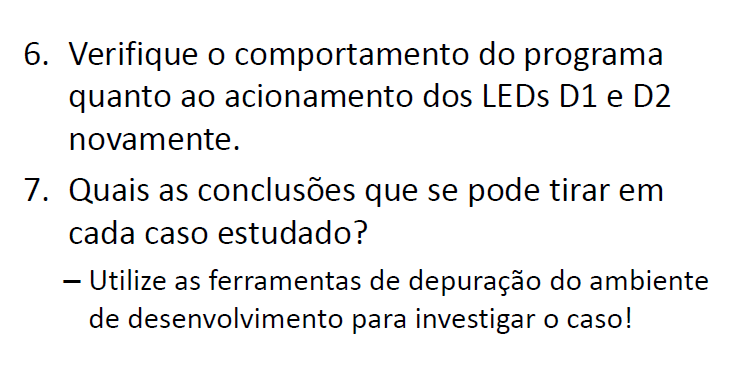
mutex2\_id = osMutexNew(NULL);

if(osKernelGetState() == osKernelReady)

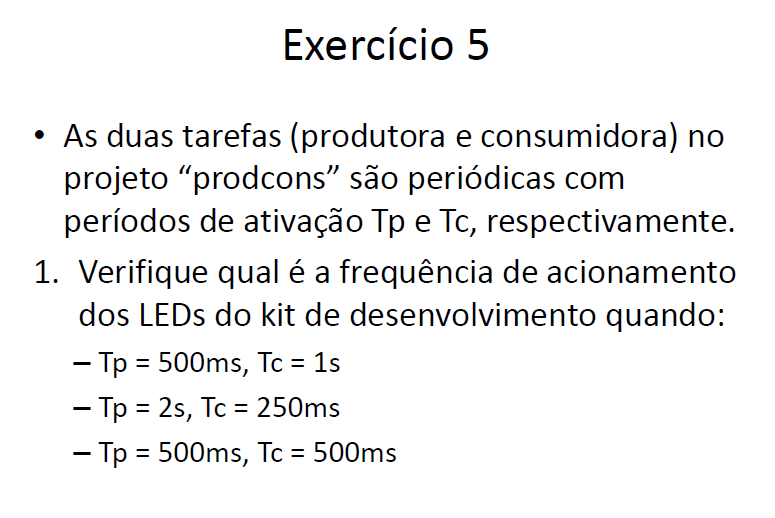
osKernelStart();

while(1);

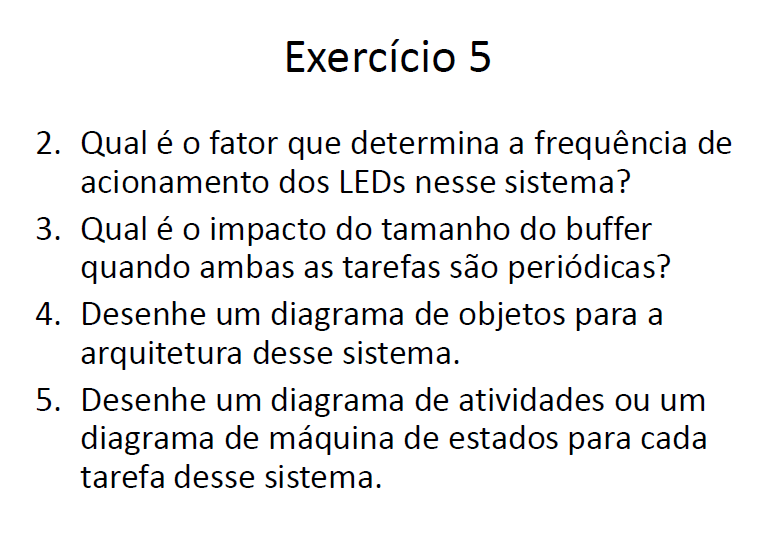
} // main



Sem a exclusão mútua, os leds piscam em conjunto. Com a garantia de exclusão mútua, aplicando dois mutexes, os leds não piscam concorrentemente e sim de forma alternada.



O que foi possível observar é que com o uso de semáforos, o tempo do produtor sempre será respeitado e o consumidor ficará aguardando cada “produto” ser disponibilizado de acordo com o tempo do produtor. Ambos estão com a contagem no buffer sincronizada, produz um, consome um e assim por diante. Caso não utilizássemos semáforos, o consumidor quando tivesse tempo menor que o do produtor, consumiria mais rápido do que o produtor pode produzir e aconteceria um esvaziamento do buffer. Já se o tempo do produtor fosse menor que o tempo do consumidor aconteceria um fila no buffer sem ser consumida.



O fator que determina a frequência de acionamento dos LEDS é o tempo do produtor.

Quanto ao buffer, ao passo que um coloca o produto no buffer o outro tira e assim sucessivamente, ou seja o buffer nunca ficará cheio.