Exercícios S05 Sistemas Embarcados

Francielle de Lima Brozoski

Exercício 3

1. Abra o projeto “tarefas” a partir da área de trabalho “EK-TM4C1294_RTOS_IAR8”.
2. Modifique o código-fonte no arquivo `tarefas.c` para garantir exclusão mútua (evitando condições de corrida) no acesso aos LEDs pelas tarefas `thread1` e `thread2`.
3. Aproveite para criar atributos com nomes para cada tarefa (ver documentação do RTX)

```
#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
```

Exercício 4

1. Abra o projeto “tarefas” a partir da área de trabalho “EK-TM4C1294_RTOS_IAR8”.
2. Modifique o código-fonte no arquivo `tarefas.c` para que as duas tarefas periódicas sejam temporizadas com `osDelayUntil()`.
3. Crie dois mutexes diferentes para e utilize-os nas tarefas da forma descrita a seguir.

```
#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
```

6. Verifique o comportamento do programa quanto ao acionamento dos LEDs D1 e D2 novamente.
7. Quais as conclusões que se pode tirar em cada caso estudado?
   - Utilize as ferramentas de depuração do ambiente de desenvolvimento para investigar o caso!

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.

Exercício 5

- As duas tarefas (produtora e consumidora) no projeto "prodcons" são periódicas com períodos de ativação Tp e Tc, respectivamente.

1. Verifique qual é a frequência de acionamento dos LEDs do kit de desenvolvimento quando:
   - Tp = 500ms, Tc = 1s
   - Tp = 2s, Tc = 250ms
   - Tp = 500ms, Tc = 500ms

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.

Exercício 5

2. Qual é o fator que determina a frequência de acionamento dos LEDs nesse sistema?
3. Qual é o impacto do tamanho do buffer quando ambas as tarefas são periódicas?
4. Desenhe um diagrama de objetos para a arquitetura desse sistema.
5. Desenhe um diagrama de atividades ou um diagrama de máquina de estados para cada tarefa desse sistema.

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.