8 de Octubre de 2024

|  |  |  |  |
| --- | --- | --- | --- |
| **PLANIFICACIÓN DE PROCESOS DE TIEMPO REAL** | |  | TP3 |
| Facultad de Informática UNLP  Sistemas de Tiempo Real | |  |  |
|  | |  |  |
|  | |  |  |
|  |  |  | |
| * Melina Caciani Toniolo   melicaciani@gmail.com | |  | 02866/1 |
| * Joaquín Chanquía   joaquin.chanquia@alu.ing.unlp.edu.ar | |  | 02887/7 |
| * Mateo Emmanuel Larsen   larsenmateo.ml@gmail.com | |  | 02993/7 |
| * Gabriel Ollier   gabyollier@hotmail.com | |  | 02958/4 |
| * Franco Niderhaus   franconiderhaus@gmail.com   * Bruno Zanetti   bzanetti09@gmail.com | |  | 02976/6  02975/5 |

# EJERCICIO 1

# EJERCICIO 2

* **Objetivo**

1. Desarrolle una aplicación de planificación que contenga 3 tareas. Las mismas deben competir por imprimir en la terminal un mensaje con el nombre de cada tarea.
2. Realice pruebas variando las prioridades de cada una de las tareas, generando así diferentes escenarios en el mensaje de salida de la terminal.

* **Resolución**

1. Para desarrollar esta aplicación se utilizo el material subido por la catedra para la configuración del IDE de Arduino. Luego se programó un código en C que utiliza la librería freeRTOS para el manejo de las tres tareas que pide el enunciado.

El código es el siguiente:

// Incluimos la libreria Arduino FreeRTOS

#include <Arduino\_FreeRTOS.h>

//Declaración de las tareas

void Task1(void \*pvParameters);

void Task2(void \*pvParameters);

void Task3(void \*pvParameters);

//Código de arranque

void setup() {

  //Frecuencia de transmisión UART

  Serial.begin(9600);

  Serial.println("Comienza el programa");

  // Crear las tres tareas

  xTaskCreate(Task1, "Tarea 1", 128, NULL, 1, NULL);

  xTaskCreate(Task2, "Tarea 2", 128, NULL, 2, NULL);

  xTaskCreate(Task3, "Tarea 3", 128, NULL, 3, NULL);

  vTaskStartScheduler();

}

//Loop infinito

void loop() {

}

//Definición de las tres tareas

//Cada tarea imprime su nombre y luego cede la CPU por 1 segundo

void Task1(void \*pvParameters) {

  while(true) {

    Serial.println("Tarea 1 \n\r");

    vTaskDelay(1000 / portTICK\_PERIOD\_MS); // Espera 1 segundo

  }

}

void Task2(void \*pvParameters) {

  while(true) {

    Serial.println("Tarea 2 \n\r");

    vTaskDelay(1000 / portTICK\_PERIOD\_MS); // Espera 1 segundo

  }

}

void Task3(void \*pvParameters) {

  while(true) {

    Serial.println("Tarea 3 \n\r");

    vTaskDelay(1000 / portTICK\_PERIOD\_MS); // Espera 1 segundo

  }

}

1. Haciendo pruebas con distintas prioridades entre las tareas llegamos a las siguientes conclusiones:

**Igual prioridad entre todas las tareas:**

Si se usa la misma prioridad entre todas las tareas estas se ejecutan en el orden en el que están escritas en el programa.

Captura de pantalla de computadora

Descripción generada automáticamente

*Ejemplo con todas las tareas con prioridad 1.*

**Dos tareas con igual prioridad y una con una prioridad mayor:**

En este caso se ejecuta primero la tarea con mayor prioridad y luego las otras dos tareas se ejecutan según el orden del programa.

Captura de pantalla de computadora

Descripción generada automáticamente

*Ejemplo con la tarea 3 con prioridad 2 y las tareas 1 y 2 con prioridad 1.*

**Dos tareas con igual prioridad y una con una prioridad menor:**

Al igual que en el caso anterior las tareas con la misma prioridad se ejecutan en orden y luego se ejecuta la tarea con menor prioridad.

Captura de pantalla de computadora

Descripción generada automáticamente

*Ejemplo con la tarea 1 con prioridad 1 y las tareas 2 y 3 con prioridad 2.*

**Todas las tareas con prioridad diferente:**

En este caso las tareas se ejecutan siguiendo el orden de la prioridad que tienen.

Captura de pantalla de computadora

Descripción generada automáticamente

*Ejemplo con las prioridades: Tarea3: 3, Tarea2: 2 y Tarea1: 1.*

**EJERCICIO 3**

**EJERCICIO 4**

* **Objetivo**

Utilizando las herramientas de sincronización, implementar una aplicación multitarea que muestre constantemente en la terminal las siguientes secuencias:

1. Tarea 1 - Tarea 3 - Tarea 2
2. Tarea 2 - Tarea 2 - Tarea 3 - Tarea 1
3. Tarea 3 - Tarea 3 - Tarea 3 - Tarea 1 - Tarea 2

Nota: Todas las tareas deben tener la misma prioridad

* **Resolución**

1. **Tarea 1 - Tarea 3 - Tarea 2**

#include <Arduino\_FreeRTOS.h>

#include <semphr.h>

// Declaramos los semáforos

SemaphoreHandle\_t semaforoTarea1;

SemaphoreHandle\_t semaforoTarea2;

SemaphoreHandle\_t semaforoTarea3;

// Tarea 1

void tarea1(void \*pvParameters) {

  while (1) {

    // Esperamos el semáforo de la tarea 1

    xSemaphoreTake(semaforoTarea1, portMAX\_DELAY);

    Serial.println("Tarea 1\n\r");

    // Damos el semáforo para la tarea 3

    xSemaphoreGive(semaforoTarea3);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

// Tarea 2

void tarea2(void \*pvParameters) {

  while (1) {

    static int pased = false;

    // Esperamos el semáforo de la tarea 2

    xSemaphoreTake(semaforoTarea2, portMAX\_DELAY);

    Serial.println("Tarea 2\n\r");

    // Damos el semáforo para la tarea 1

    xSemaphoreGive(semaforoTarea1);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

// Tarea 3

void tarea3(void \*pvParameters) {

  while (1) {

    // Esperamos el semáforo de la tarea 3

    xSemaphoreTake(semaforoTarea3, portMAX\_DELAY);

    Serial.println("Tarea 3\n\r");

    // Damos el semáforo para la tarea 2

    xSemaphoreGive(semaforoTarea2);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

void setup() {

  Serial.begin(9600);

  // Creamos los semáforos

  semaforoTarea1 = xSemaphoreCreateBinary();

  semaforoTarea2 = xSemaphoreCreateBinary();

  semaforoTarea3 = xSemaphoreCreateBinary();

  // Creamos las tareas

  xTaskCreate(tarea1, "Tarea 1", 128, NULL, 1, NULL);

  xTaskCreate(tarea2, "Tarea 2", 128, NULL, 1, NULL);

  xTaskCreate(tarea3, "Tarea 3", 128, NULL, 1, NULL);

  // Inicializamos la secuencia dando el semáforo de la Tarea 1

  xSemaphoreGive(semaforoTarea1);

}

void loop() {

  // El loop se queda vacío, ya que FreeRTOS maneja las tareas

}

Captura de pantalla de computadora

Descripción generada automáticamente

1. **Tarea 2 - Tarea 2 - Tarea 3 - Tarea 1**

#include <Arduino\_FreeRTOS.h>

#include <semphr.h>

// Declaramos los semáforos

SemaphoreHandle\_t semaforoTarea1;

SemaphoreHandle\_t semaforoTarea2;

SemaphoreHandle\_t semaforoTarea3;

// Tarea 1

void tarea1(void \*pvParameters) {

  while (1) {

    // Esperamos el semáforo de la tarea 1

    xSemaphoreTake(semaforoTarea1, portMAX\_DELAY);

    Serial.println("Tarea 1\n\r");

    // Damos el semáforo para la tarea 2

    xSemaphoreGive(semaforoTarea2);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

// Tarea 2

void tarea2(void \*pvParameters) {

  while (1) {

    static int pased = 0;

    // Esperamos el semáforo de la tarea 2

    xSemaphoreTake(semaforoTarea2, portMAX\_DELAY);

    Serial.println("Tarea 2\n\r");

    if (pased){

      pased = 0;

      // Damos el semáforo para la tarea 3

      xSemaphoreGive(semaforoTarea3);

    }else{

      pased = 1;

      // Damos el semáforo para la tarea 2

      xSemaphoreGive(semaforoTarea2);

    }

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

// Tarea 3

void tarea3(void \*pvParameters) {

  while (1) {

    // Esperamos el semáforo de la tarea 3

    xSemaphoreTake(semaforoTarea3, portMAX\_DELAY);

    Serial.println("Tarea 3\n\r");

    // Damos el semáforo para la tarea 1

    xSemaphoreGive(semaforoTarea1);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

void setup() {

  Serial.begin(9600);

  // Creamos los semáforos

  semaforoTarea1 = xSemaphoreCreateBinary();

  semaforoTarea2 = xSemaphoreCreateBinary();

  semaforoTarea3 = xSemaphoreCreateBinary();

  // Creamos las tareas

  xTaskCreate(tarea1, "Tarea 1", 128, NULL, 1, NULL);

  xTaskCreate(tarea2, "Tarea 2", 128, NULL, 1, NULL);

  xTaskCreate(tarea3, "Tarea 3", 128, NULL, 1, NULL);

  // Inicializamos la secuencia dando el semáforo de la Tarea 1

  xSemaphoreGive(semaforoTarea2);

}

void loop() {

  // El loop se queda vacío, ya que FreeRTOS maneja las tareas

}

**Captura de pantalla de computadora

Descripción generada automáticamente**

1. **Tarea 3 - Tarea 3 - Tarea 3 - Tarea 1 - Tarea 2**

#include <Arduino\_FreeRTOS.h>

#include <semphr.h>

// Declaramos los semáforos

SemaphoreHandle\_t semaforoTarea1;

SemaphoreHandle\_t semaforoTarea2;

SemaphoreHandle\_t semaforoTarea3;

// Tarea 1

void tarea1(void \*pvParameters) {

  while (1) {

    // Esperamos el semáforo de la tarea 1

    xSemaphoreTake(semaforoTarea1, portMAX\_DELAY);

    Serial.println("Tarea 1\n\r");

    // Damos el semáforo para la tarea 2

    xSemaphoreGive(semaforoTarea2);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

// Tarea 2

void tarea2(void \*pvParameters) {

  while (1) {

    static int pased = false;

    // Esperamos el semáforo de la tarea 2

    xSemaphoreTake(semaforoTarea2, portMAX\_DELAY);

    Serial.println("Tarea 2\n\r");

    // Damos el semáforo para la tarea 3

    xSemaphoreGive(semaforoTarea3);

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

// Tarea 3

void tarea3(void \*pvParameters) {

  while (1) {

    static int count = 3;

    // Esperamos el semáforo de la tarea 3

    xSemaphoreTake(semaforoTarea3, portMAX\_DELAY);

    Serial.println("Tarea 3\n\r");

    count--;

    if (count){

      // Damos el semáforo para la tarea 3

      xSemaphoreGive(semaforoTarea3);

    }else{

      count = 3;

      // Damos el semáforo para la tarea 1

      xSemaphoreGive(semaforoTarea1);

    }

    vTaskDelay(500 / portTICK\_PERIOD\_MS);  // Añadimos un pequeño delay

  }

}

void setup() {

  Serial.begin(9600);

  // Creamos los semáforos

  semaforoTarea1 = xSemaphoreCreateBinary();

  semaforoTarea2 = xSemaphoreCreateBinary();

  semaforoTarea3 = xSemaphoreCreateBinary();

  // Creamos las tareas

  xTaskCreate(tarea1, "Tarea 1", 128, NULL, 1, NULL);

  xTaskCreate(tarea2, "Tarea 2", 128, NULL, 1, NULL);

  xTaskCreate(tarea3, "Tarea 3", 128, NULL, 1, NULL);

  // Inicializamos la secuencia dando el semáforo de la Tarea 1

  xSemaphoreGive(semaforoTarea3);

}

void loop() {

  // El loop se queda vacío, ya que FreeRTOS maneja las tareas

}

**Captura de pantalla de computadora

Descripción generada automáticamente**