ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



BÁO CÁO HỆ THỐNG NHÚNG

Bài thực hành số 3

**Lương Hữu Phú Lợi - 1911545**

**Cao Thanh Lương - 1914076**

**Nguyễn Văn Việt - 1912436**

**Huỳnh Ngọc Bảo Trân - 1912269**

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **MSSV** | **Đóng góp** |
| Lương Hữu Phú Lợi | 1911545 | 100% |
| Cao Thanh Lương | 1914076 | 100% |
| Nguyễn Văn Việt | 1912436 | 100% |
| Huỳnh Ngọc Bảo Trân | 1912269 | 100% |

**GitHub:** [PhuLoi-1911545/school-LAB-embedded-ESP-IDF (github.com)](https://github.com/PhuLoi-1911545/school-LAB-embedded-ESP-IDF)

1. Prioritized Pre-emptive Scheduling with Time Slicing

Ta có đoạn chương trình như sau:

static void task\_1(void\* *args*)

{

    while(1)

    {

        printf("\n L01 - Nhom 3 - TASK 1 +\n");

        vTaskDelay(pdMS\_TO\_TICKS(100));

    }

}

static void task\_2(void\* *args*)

{

    while(1)

    {

        printf("\n L01 - Nhom 3 - TASK 2\n");

        vTaskDelay(pdMS\_TO\_TICKS(100));

    }

}

static void task\_3(void\* *args*)

{

    while(1)

    {

        printf("\n L01 - Nhom 3 - TASK 3\n");

        vTaskDelay(pdMS\_TO\_TICKS(1000));

    }

}

void app\_main(void) {

    vTaskDelay(pdMS\_TO\_TICKS(100));

xTaskCreatePinnedToCore(task\_3, "task 3", 1024, NULL, 0, NULL, 0);

    xTaskCreatePinnedToCore(task\_2, "task 2", 1024, NULL, 0, NULL, 0);

    vTaskDelay(pdMS\_TO\_TICKS(5000));

    xTaskCreatePinnedToCore(task\_1, "task 1", 1024, NULL, 1, NULL, 0);

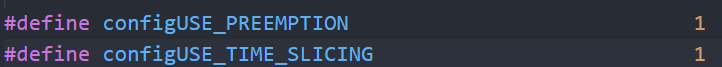
}

Ta vào đường dẫn:

… esp-idf\components\freertos\include\esp\_additions\freertos

rồi mở file FreeRTOSConfig.h

Tìm và chỉnh sửa 2 dòng sau:



Sau đó nạp code xuống ESP32 và được kết quả như hình bên dưới. Ta thấy task\_2 và task\_3 luân phiên nhau chạy (task\_2 bị delay 100ms và task\_3 bị delay 1000ms). Cho đến khi task\_1 có priority cao hơn (priority 1) nhảy vào pre-empt chiếm quyền chạy.

Graphical user interface, text

Description automatically generated

A picture containing text

Description automatically generated

1. Prioritized Pre-emptive Scheduling (without Time Slicing)

Ta có đoạn chương trình sau:

static void task\_1(void\* *arg*)

{

    while(1)

    {

for (int j=0; j<5; j++)

{

printf("L01 - Nhom 3 - TASK 1 \n");

}

        vTaskDelay(pdMS\_TO\_TICKS(1000));

    }

}

static void task\_2(void\* *arg*)

{

    while(1)

    {

        printf("\n L01 - Nhom 3 - TASK 2\n");

        vTaskDelay(pdMS\_TO\_TICKS(100));

    }

}

static void task\_3(void\* *arg*)

{

    while(1)

    {

        printf("\n L01 - Nhom 3 - TASK 3\n");

        vTaskDelay(pdMS\_TO\_TICKS(100));

    }

}

void app\_main(void) {

    vTaskDelay(pdMS\_TO\_TICKS(100));

xTaskCreatePinnedToCore(task\_3, "task 3", 1024, NULL, 0, NULL, 0);

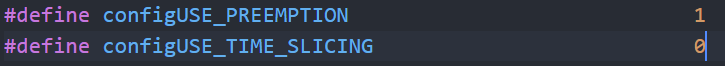
    xTaskCreatePinnedToCore(task\_2, "task 2", 1024, NULL, 0, NULL, 0);

    vTaskDelay(pdMS\_TO\_TICKS(5000));

    xTaskCreatePinnedToCore(task\_1, "task 1", 1024, NULL, 1, NULL, 0);

}

Ta tiến hành đổi giá trị của configUSE\_TIME\_SLICING thành 0



Sau đó nạp code xuống ESP32 và được kết quả như hình bên dưới. Ta thấy vì không sử dụng time slicing cho nên task\_3 sẽ được chạy liên tục, không cho task\_2 nhảy vào pre-empt. Sau đó task\_1 nhảy vào pre-empt task\_3 (bởi vì task\_1 có priority là 1 cao hơn priority 0 của task\_3). Sau khi task\_1 thực thi xong, lúc này task\_2 sẽ được lấy ra khỏi hàng đợi trước và được thực thi.

A screenshot of a computer

Description automatically generated with medium confidence

1. Co-operative Scheduling

Ta có đoạn chương trình sau:

#include <stdio.h>

#include <stdlib.h>

#include "sdkconfig.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "esp\_system.h"

#include "esp\_spi\_flash.h"

#include "driver/gpio.h"

#include "freertos/semphr.h"

#include "esp\_err.h"

static void task\_1(void\* *arg*)

{

    while (1) {

        printf("Task 1 \n");

        vTaskDelay(pdMS\_TO\_TICKS(1000));

    }

}

static void task\_2(void\* *arg*)

{

    while (1) {

        printf("Task 2 \n");

        vTaskDelay(pdMS\_TO\_TICKS(4000));

        taskYIELD();

    }

}

static void task\_3(void\* *arg*)

{

    while (1) {

        printf("Task 3 \n");

        vTaskDelay(pdMS\_TO\_TICKS(1000));

        taskYIELD();

    }

}

void app\_main(void)

{

    vTaskDelay(pdMS\_TO\_TICKS(100));

    xTaskCreatePinnedToCore(task\_3, "task 3", 1024, NULL, 0, NULL, 0);

*// vTaskDelay(pdMS\_TO\_TICKS(4000));*

    xTaskCreatePinnedToCore(task\_2, "task 2", 1024, NULL, 0, NULL, 0);

    vTaskDelay(pdMS\_TO\_TICKS(5000));

    xTaskCreatePinnedToCore(task\_1, "task 1", 1024, NULL, 1, NULL, 0);

}

Ta tiếp tục đổi giá trị của configUSE\_PREEMPTION thành 0



Sau đó nạp code xuống ESP32 và được kết quả như hình bên dưới. Ta thấy task\_3 được khởi tạo và chạy trước, sau đó sẽ đến task\_2 (task\_3 bị delay 1000ms còn task\_2 bị delay 4000ms) sau khi task 3 gọi taskYield. Sau đó task\_1 nhảy vào và được thực thi sau khi task\_3 hoặc task\_2 (task được thực thi trước đó) gọi taskYield. Sau khi task\_1 gọi delay thì task kế tiếp (task\_2 hoặc task\_3) sẽ được thực thi.

Text

Description automatically generated