Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Systèmes Embarqués

TD n°2

Système embarqué & Programmation RTOS

Programmation RTOS

1 – Soit l'application suivante :

```
static int shared_var = 0;

void incTask(void *parameters) {
  int local_var;

  while (1) {
    local_var = shared_var;
    local_var++;
    vTaskDelay(random(100, 500) / portTICK_PERIOD_MS);
    shared_var = local_var;
    Serial.println(shared_var);
  }
}

void setup() {
  randomSeed(analogRead(0));
  Serial.begin(115200);
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  xTaskCreate(incTask, "Task 1", 1024, NULL, 1, NULL);
  xTaskCreate(incTask, "Task 2", 1024, NULL, 1, NULL);
  vTaskDelete(NULL);
}

void loop() {
}
```

a. Est-ce qu'elle fonctionne correctement ?
Expliquez ce qui se passe.

b. Comment la corriger ?

c. Donnez le code de la correction.

2 – Soit l'application FreeRTOS suivante :

```
#define TAILLEMAX 85
char buffer[TAILLEMAX];
volatile int rang_buffer = 0;
volatile bool a_afficher = false;

void printDirect(void *paramaters) {
  int nb_messages = 0;
  char buffer_tache[128];
  int taille_message = 0;

  while(1){
    sprintf(buffer_tache, "[Depuis 1 message %d]", ++nb_messages);
    taille_message = strlen(buffer_tache);
    if ((TAILLEMAX-rang_buffer)> taille_message)
    {
      strncpy(&buffer[rang_buffer], buffer_tache, taille_message);
      rang_buffer += taille_message;
    }
    else
    {
      strncpy(&buffer[rang_buffer], buffer_tache, TAILLEMAX-rang_buffer);
      rang_buffer = TAILLEMAX - 1;
    }
    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
```

a. À quoi sert le `volatile` ❶ ?

```
void printDirect2(void *paramaters) {
  int nb_messages = 0;
  char buffer_tache[128];
  int taille_message = 0;

  while(1){
    sprintf(buffer_tache, "[Depuis 2 message %d]", ++nb_messages);
    taille_message = strlen(buffer_tache);
    if ((TAILLEMAX-rang_buffer)> taille_message)
    {
      strncpy(&buffer[rang_buffer], buffer_tache, taille_message);
      rang_buffer += taille_message;
    }
    else
    {
      strncpy(&buffer[rang_buffer], buffer_tache, TAILLEMAX-rang_buffer);
      rang_buffer = TAILLEMAX - 1;
    }
    vTaskDelay(500 / portTICK_PERIOD_MS);
  }
}
void printBuffer(void *paramaters) {
  while (1) {
    if (rang_buffer == (TAILLEMAX - 1)) {
        buffer[rang_buffer] = '\0';
        if (rang_buffer!=0)
        {
          Serial.println(buffer);
        }
        rang_buffer = 0;
        a_afficher = false;
      }
  }
}
```

```
void setup() {
  Serial.begin(115200);
  xTaskCreate(printDirect,
              "Tache 1",
              2048,
              NULL,
              1,
              NULL);
  xTaskCreate(printDirect2,
              "Tache 2",
              2048,
              NULL,
              1,
              NULL);
  xTaskCreate(printBuffer,
              "Print Messages",
              1024,
              NULL,
              1,
              NULL);
}

void loop() {

}
```

b. Décrivez son fonctionnement.

c. Est-ce qu'il est **correct** ?

d. Est-ce qu'il est **efficace** ?

e. Proposez une **meilleure version** utilisant les outils disponibles dans FreeRTOS.