Rapport de TP2 de SEPT

Matthieu Jolimaitre & Quentin Harro

Sémaphore

Une sémaphore est un objet partagé entre les fonctions de plusieurs fils d'exécutions :

```
static SemaphoreHandle_t semaphore;
```

Une tâche peut libérer le sémaphore de cette manière :

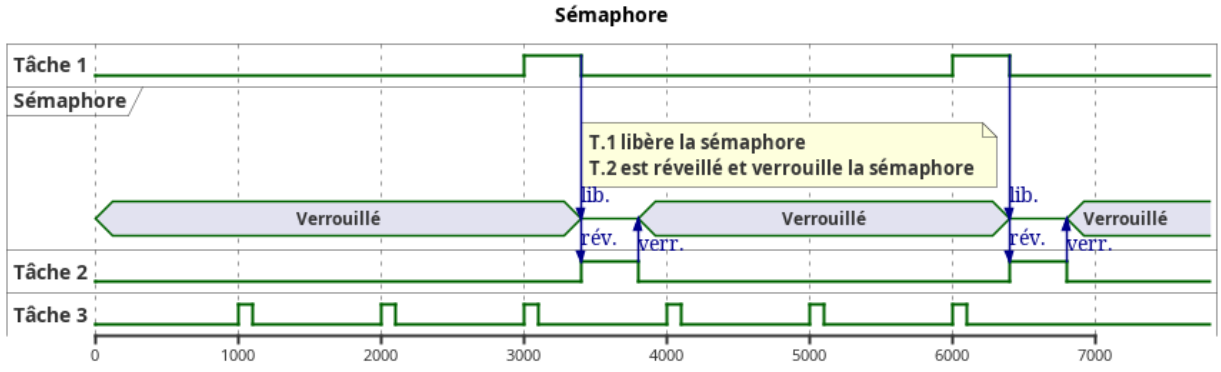
```
void task_1(void* params) {
    printf("[Task 1] start\n");
    while (1) {
        xSemaphoreGive(semaphore);
        printf("[Task 1] gave semaphore\n");
        vTaskDelay(_3_SECONDS);
    }
}
```

note : Ici, la sémaphore est libérée périodiquement toute les 4 secondes.

Une autre tâche peut attendre la libération de cette sémaphore de cette manière :

```
void task_2(void* params) {
    printf("[Task 2] start\n");
    while (1) {
        printf("[Task 2] waiting semaphore\n");
        xSemaphoreTake(semaphore, portMAX_DELAY);
        printf("[Task 2] took semaphore\n");
        printf("[Task 2] Message from task 2\n");
    }
}
```

Nous sommes donc assurés que la tâche 2 exécutera systématiquement sa section critique après celle de la tâche 1.



semaphore

note : Ici, la tâche attend périodiquement la libération de la sémaphore.

Queue

Une queue est un objet partagé entre les fonctions de plusieurs fils d'exécutions :

```
static QueueHandle_t queue;
```

Une tâche peut enfiler une valeur dans la queue de cette manière :

```
void task_1(void* params) {
    printf("[Task 1] start\n");
    while (1) {
        auto ticks = xTaskGetTickCount();
        printf("[Task 1] sending %lu to queue\n", ticks);
        xQueueSend(queue, &ticks, portMAX_DELAY);
        vTaskDelay(_1_SECOND);
    }
}
```

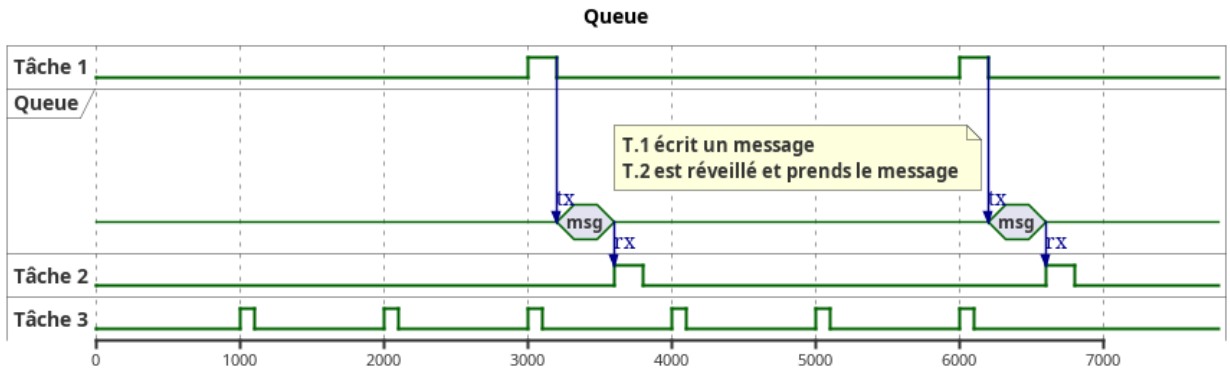
note : Ici, la tâche enfile périodiquement une empreinte temporelle dans la queue.

Une tâche peut ensuite itérer de manière bloquante sur les valeurs enfilés dans la queue :

```
void task_2(void* params) {
    printf("[Task 2] start\n");
    while (1) {
        TickType_t result;
        xQueueReceive(queue, &result, portMAX_DELAY);
        printf("[Task 2] Received %lu from queue\n", result);
    }
}
```

note : Ici, la tâche itère sur les empreintes temporelles que la tâche précédente enfile.

Nous pouvons donc transférer un contexte d'un fil d'exécution à un autre sans risque de corruption.



queue