UFSC – Universidade Federal de Santa Catarina
Centro Tecnológico de Joinville
EMB5633 – Sistemas em Tempo Real

Dados do Aluno

Matrícula	Turma	Aluno
22101934	06605	Kamylo Serafim Porto

PROJETO 2 - Escalonador de tarefas periódicas

O escalonamento periódico implementado foi o EDF.

Foi criada uma struct para cada tarefa periódica, onde são armazenadas informações sobre deadline e período (absolutos e dinâmicos). Essa struct foi implementada no Thread Control Block, pois, nesta implementação, cada thread representa uma tarefa periódica. As informações absolutas são aquelas que não são alteradas durante o escalonamento das tarefas, sendo intrínsecas a cada tarefa. Já as informações dinâmicas são atualizadas a cada tick.

De maneira geral, uma tarefa pode estar pronta para ser escalonada ou esperando o próximo período de ativação.

A função *OS_calculate_next_periodic_task()* tem como objetivo calcular a tarefa não concluída com o deadline mais próximo. Ela é chamada sempre antes de uma tarefa ser escalonada pela *OS_sched()*, ou seja, é invocada tanto na *OS_run()* (devido ao primeiro escalonamento), quando uma tarefa é finalizada (quando a *OS_wait_next_period()* é chamada) e na *SysTick_Handler()* (quando uma nova tarefa deve ser escalonada).

A função *OS_wait_next_period()* é chamada quando uma tarefa finaliza seu processamento, ou seja, quando o processador fica livre para executar uma próxima tarefa. A sua função é colocar aquela task em espera e escalonar outra.

Além dessas duas funções, foram realizadas alterações na *OS_tick()* e na *OS_sched()*. Na *OS_tick()*, foi adicionado o compromisso de atualizar as variáveis dinâmicas das tarefas. Na *OS_sched()*, as tarefas passaram a ser escalonadas pelo índice da tarefa com o deadline mais próximo, conforme calculado pela função

OS_calculate_next_periodic_task().

Em relação às tarefas que foram escalonadas, foi simulado o custo da tarefa através de um contador. Para calcular esses custos, foram realizados diversos testes a fim de medir o tempo médio que o OS demora, em ticks, para realizar o loop. A figura abaixo exemplifica como foi realizado esse cálculo.

```
void task1(){
    while(1){
        uint32_t start_tick = HAL_GetTick();

        actual_task = 1;

        cont_task_1 = 0;

        while(cont_task_1 < 0xFFFF){
            cont_task_1++;
        }

        time_task1 = HAL_GetTick() - start_tick;

        OS_wait_next_period();
    }
}
```

Com isso, foi possível definir três valores de contador para as três tarefas utilizadas. Esses valores, o custo, deadline e período são mostrados na tabela abaixo.

Tarefa	Contador	Custo (ticks)	Deadline (ticks)	Período (ticks)
T_1	0xFFFF	13	50	50
T_2	0X1FFFF	29	100	100
T_3	0X2FFFF	44	200	200

Com esses valores, temos uma utilização de, aproximadamente, 77%, logo, é escalonável por EDF.