Sistemas Embarcados - Trabalho 1

Objetivo

O objetivo geral do trabalho é adicionar ao kernel do UCX/OS o suporte para o escalonamento de tarefas periódicas utilizando a política LRLF (*Least Runtime Laxity First*). A implementação do novo escalonador deverá complementar o modelo de execução de tarefas já implementado, de tal forma que o kernel suporte tarefas de tempo real e execute tarefas de melhor esforço (com prioridades) no tempo livre de CPU.

Descrição

O kernel do UCX/OS suporta um modelo de execução de tarefas misto, onde são suportados o escalonamento de tempo real e o escalonamento de melhor esforço. Tarefas de melhor esforço são escalonadas por meio da política Round-Robin com prioridades relativas, que é o mecanismo padrão para a execução de tarefas. A forma como o escalonador funciona baseia-se em dois aspectos: estado da tarefa (TASK_STOPPED, TASK_READY, TASK_RUNNING, TASK_BLOCKED, TASK_SUSPENDED) e prioridade relativa. Internamente, o kernel mantém uma lista encadeada de descritores de tarefas, implementando uma estrutura de controle (TCB, *task control block*). Na função *krnl_schedule()*, a decisão sobre qual tarefa deve ser executada depende do escalonador utilizado (tarefa de tempo real ou não), de seu estado (apenas tarefas no estado TASK_READY estão aptas a executar) e de sua prioridade (para tarefas de melhor esforço).

A estrutura de dados KCB (*Kernel Control Block*), encontrada no arquivo *ucx-os/include/kernel/kernel.h* referencia uma possível implementação de um escalonador definido pelo usuário no ponteiro para função *rt_sched*:

```
/* kernel control block */
struct kcb_s {
    struct list_s *tasks;
    struct node_s *task_current;
    jmp_buf context;
    int32_t (*rt_sched)(void);
    struct list_s *timer_lst;
    volatile uint32_t ticks;
    uint16_t id_next;
    char preemptive;
};

extern struct kcb_s *kcb;
```

Um escalonador definido pelo usuário deve escolher a próxima tarefa a executar iterando circularmente sobre a lista de tarefas, além de gerenciar o estado da tarefa atual (preemptada) de acordo com alguns passos:

- Verificar o estado da tarefa atual e colocá-la no estado TASK_READY;
- Tentar escalonar uma tarefa de tempo real, de acordo com a política implementada;
- Caso escalonada, atualizar o ponteiro $kcb \rightarrow task\_current$ para o elemento da lista de tarefas e atualizar o estado da tarefa para $TASK\_RUNNING$;
- Retornar o *id* da tarefa escalonada ou -1 caso não seja.

Caso não seja escalonada uma tarefa de tempo real, a referência $kcb \rightarrow task\_current$ não deve ser atualizada e o valor -1 deve ser retornado pelo escalonador. Dessa forma, o kernel sabe que deve escolher uma tarefa de melhor esforço para execução. As tarefas adicionadas ao sistema são referenciadas em uma lista por meio de um ponteiro na estrutura de dados KCB. Essa lista é composta por nodos que implementam uma estrutura de dados por tarefa, chamada TCB (*Task Control Block*):

```
/* task states */
enum task_states {TASK_STOPPED, TASK_READY, TASK_RUNNING,
    TASK_BLOCKED, TASK_SUSPENDED};

/* task control block node */
struct tcb_s {
    void (*task)(void);
    jmp_buf context;
    size_t *stack;
    size_t stack_sz;
    void *rt_prio;
    uint16_t id;
    uint16_t delay;
    uint16_t priority;
    uint8_t state;
};
```

Caso uma determinada tarefa seja de tempo real, a entrada *rt_prio* em sua TCB deve referenciar uma estrutura de dados definida pelo usuário, que contém os parâmetros necessários para atribuir e manter informações referentes à prioridade da tarefa.

O escalonador e verificação de funcionamento

Os seguintes componentes deverão fazer parte da implementação do seu escalonador de tempo real:

- Estrutura de dados utilizada para definir prioridades das tarefas;
- O escalonador de tempo real, utilizando a política LRLF.

Em uma aplicação, deve-se definir uma estrutura de dados chamada *priorities*, que irá manter os valores das prioridades das tarefas bem como a inicialização de valores para a política implementada. Além disso, o seu escalonador deverá ser associado à KCB e as tarefas de tempo real deverão ter sua prioridade atualizada por meio da função *ucx_task_rt_priority()*. Verifique o exemplo contido em *ucx-os/app/rtsched.c* para entender como pode ser implementado um escalonador de tempo real em uma aplicação. O seu

escalonador, no entanto, deverá ser um componente à parte, separado da aplicação.

A verificação do correto escalonamento das tarefas deverá ser validada na ferramenta Cheddar. Deverão ser definidos pelo menos 5 cenários de teste contendo conjuntos de tarefas diversos (pelo menos um conjuto com mais de 5 tarefas). Esses conjuntos deverão conter diferentes características, como por exemplo baixa, média e alta utilização de CPU, harmonia variada entre períodos e um caso não factível para o LRLF (que deverá gerar perdas de deadline). Não esqueça de incluir pelo menos uma tarefa de melhor esforço no conjunto de tarefas (pode ser uma *idle task*, que executa um laço infinito) para que o escalonador de melhor esforço não venha a falhar por falta de tarefas.

Apresentação, entrega e critérios avaliativos

Este trabalho deverá ser realizado individualmente ou em duplas e apresentado no dia 19/05 (apresentação em torno de 10 minutos). Para a entrega, é esperado seja enviado pelo Moodle um arquivo *.tar.gz*, contendo o código fonte da implementação, aplicações de teste e verificação do funcionamento usando o Cheddar. Os seguintes critérios serão considerados para avaliação:

- Definição das estruturas de dados, adição das tarefas e configuração do escalonador (2 pontos);
- Implementação do escalonador de tarefas de acordo com a política LRLF (3 pontos);
- Validação da implementação, definição de conjuntos de tarefas para testes e comparação do funcionamento com a ferramenta Cheddar (5 pontos);