EMB5633 - Sistemas de Tempo Real
Implementação e Análise de Thread Periódica no Linux

Prof. Giovani Gracioli

2020/02

1 Descrição

O trabalho consiste na implementação de threads ou tarefas periódicas no Linux. Será utilizado o *Timer* padrão POSIX e o sinal enviado por ele para implementar o comportamento periódico de uma thread.

Usualmente uma thread ou tarefa periódica tem um comportamento que se repete a cada período, como exemplificado no trecho de código abaixo:

```
while(wait for a new period) {
    execute the periodic work
}
```

Para a implementação deste comportamento periódico no Linux, será necessário utilizar no mínimo as seguintes funções:

- timer_create (clockid = CLOCK REALTIME)
- timer_settime
- sigaction (signum = SIGALRM)
- sigwait (signum = SIGALRM)

A atividade do trabalho consiste em imprimir uma mensagem de aviso caso o *deadline* (que deve ser igual ao período da tarefa) for perdido, além de uma análise para entender a relação entre o escalonador e o tempo de resposta de tarefas concorrentes.

Para isso, deve-se usar o comportamento da biblioteca de *Timer* do Linux: se um sinal for gerado pelo *Timer* e o processo/thread não está bloqueado na função `sigwait`, o tratador (*handler*) registrado para o sinal é usado em seu lugar.

O programa deve receber pela linha de comando os seguintes argumentos:

- período (em milisegundos, de 1 ms a 999 ms);
- prioridade da tarefa/thread;
- fator de carga da CPU;
- política de escalonamento (SCHED_FIFO ou SCHED_RR).

A cada período, a tarefa, depois de receber o sinal do *Timer*, deve executar uma carga de trabalho genérica/simulada da seguinte forma:

```
 for ( int i = 0; i < load * 1000; i++) {
/* do nothing , keep counting */
}
```

Onde *load* representa o fator de carga da CPU recebido pela linha de comando.

Deve também ser impresso o tempo de resposta da tarefa, i.e., o período de tempo entre o recebimento do sinal do *Timer* e o término da computação da tarefa. Dica: a função `timer_gettime` retorna a duração relativa até o *Timer* expirar.

Para fins de análise de interferência entre duas tarefas, você deverá executar duas instâncias do programa no mesmo processador (utilizando o comando taskset antes de executar o programa para alocá-lo em um processador/core) com os parâmetros definidos da Tabela 1:

Para a análise, primeiramente observe o tempo de resposta de cada tarefa executada em isolamento (sem o segundo programa em execução). Após, execute ambas tarefas em paralelo com as seguintes configurações:

Table 1: Parâmetros de teste das tarefas.

	Período	Fator de Carga
Tarefa 1	500 ms	100000
Tarefa 2	500 ms	100000

- Ambas tarefas usando a política de escalonamento SCHED_RR com a mesma prioridade;
- Ambas tarefas usando a política de escalonamento SCHED_RR e a tarefa 1 com maior prioridade.

Compare os tempos de resposta observados e explique as diferenças no relatório e apresentação. O tempo de resposta da tarefa de mais baixa prioridade é sempre o mesmo? Porquê? O que acontece se a carga de trabalho das tarefas é diminuída ou aumentada?

2 Compilação e Funções da Biblioteca

Utilize as flags -lrt para incluir a biblioteca de tempo real POSIX e -Wall para mostrar todos os avisos de compilação.

As funções `sched_getparam, sched_setscheduler, sched_getscheduler e sched_rr_get_interval` da biblioteca *sched.h* (`https://pubs.opengroup.org/onlinepubs/7908799/xsh/sched.h.html`)podem ser usadas para as configurações da política de escalonamento.

A seguinte página contém uma descrição das principais funções associados ao *Timer* e tratamento do sinal gerado por ele: `https://www3.physnet.uni-hamburg.de/physnet/Tru64-Unix/HTML/APS33DTE/DOCU_007.HTM`.

Embora as funções utilizadas sejam da biblioteca C, na solução do trabalho deve-se utilizar a linguagem C++ e, portanto, seu código deve ter classes abstraindo o uso do *Timer* e thread periódica.

3 Entrega do Trabalho

Este trabalho deve ser entregue até a data limite estipulada pela tarefa do Moodle. Cada dupla deverá apresentar o trabalho em horário previamente agendado.

Em um único arquivo .zip, envie o código implementado incluindo um arquivo README explicando como executar o código e também um relatório explicando o código implementando e discutindo os resultados alcançados.