EMB5632 – Sistemas Operacionais
Prof. Dr. Giovani Gracioli
2020/01

Projeto de um Sistema Operacional em Nível de Usuário
Trabalho 6: Preempção baseada em tempo

1. Descrição

Este trabalho tem os seguintes objetivos:

1. Usar uma interrupção de um temporizador para executar uma rotina de reescalonamento (preempção) baseada em tempo.

2. Criar uma classe Timer para abstrair a criação e tratamento de sinais do Linux que irão simular as interrupções de um timer.

3. Implementar o método "static void reschedule()" na Thread que irá realizar o reescalonamento do sistema.

Os seguintes métodos devem ser implementados na classe Timer:

- **Timer(const unsigned int period, const Function * handler);**

  Construtor que recebe o período (em us) para realizar as interrupções do Timer e um ponteiro de função que irá ser chamada para tratar a interrupção. Function é definida na própria classe Timer.

- **void reset();**

  Este método deve resetar a contagem do Timer.

A classe Timer irá utilizar o mecanismo de sinais do Unix/Linux para simular as interruções de um timer de hardware. Mais especificamente, será utilizado o sinal SIGALRM. Para isso, a classe Timer possui dois atributos, s**truct sigaction action** e **struct itimerval timer**. Os links abaixos apresentam uma explicação sobre sinais no Linux e um exemplo de como usar as duas estruturas para configurar um alarme no Linux:

- https://towardsdatascience.com/signals-in-linux-b34cea8c5791
- https://www.gnu.org/software/libc/manual/html_node/Setting-an-Alarm.html

A classe Thread deve ter um novo atributo estático, static Timer * _timer, que irá criar um novo Timer quando a preempção por tempo estiver habilitada no sistema (no Traits) e também um novo método, static void reschedule(), que deve ser chamado a cada interrupção de tempo do alarme.

As seguintes instruções resumem o que deve ser implementado/modificado neste trabalho:

- Métodos da classe Timer conforme descritos acima;
- A preempção por tempo ou não deve ser configurável pelo Traits do sistema. Quando habilitada, a preempção por tempo torna a thread dispatcher desnecessária. Quando desabilitada, o sistema deve funcionar da mesma forma que o trabalho anterior (usando dispatcher).
- O QUANTUM ou período entre duas interrupções de tempo deve ser definido no Traits;
- A inicialização do Timer deve ser feita pela classe Thread apenas quando a preempção por tempo estiver habilitada no sistema;
- Implementar o método reeschedule() da Thread para escolher outra Thread a ser executada quando o atual QUANTUM acabar.
- Quando o Timer deve ser resetado?

Os alunos têm a liberdade para adicionar métodos e atributos necessários para a implementação dos métodos/funcionalidades descritas acima, inclusive em outras classes.

2. Arquivos Disponibilizados

Os seguintes arquivos foram disponibilizados neste trabalho:

- timer.h: contém a declaração dos métodos que devem ser implementados e dos atributos a serem utilizados. Deve-se ainda criar um arquivo timer.cc para conter a implementação dos métodos.
- main_class.h e main_class.cc implementação de uma aplicação exemplo.

A saída esperada do programa exemplo é similar a abaixo:

```
main: inicio
main: esperando Pang...
  Pang: inicio
  Pang: 0
     Peng: inicio
     Peng: 0
        Ping: inicio
        Ping: 0
           Pong: inicio
           Pong: 0
              Pung: inicio
              Pung: 0
  Pang: 1
     Peng: 1
        Ping: 1
           Pong: 1
              Pung: 1
  Pang: 2
     Peng: 2
        Ping: 2
           Pong: 2
              Pung: 2
  Pang: 3
     Peng: 3
```

```
        Ping: 3
          Pong: 3
            Pung: 3
        Ping: 4
            Pung: 4
  Pang: 4
    Peng: 4
          Pong: 4
  Pang: 5
    Peng: 5
      Ping: 5
        Pong: 5
          Pung: 5
    Peng: 6
      Ping: 6
        Pong: 6
          Pung: 6
  Pang: 6
  Pang: 7
    Peng: 7
      Ping: 7
        Pong: 7
          Pung: 7
    Peng: 8
      Ping: 8
          Pung: 8
  Pang: 8
          Pong: 8
  Pang: 9
    Peng: 9
      Ping: 9
        Pong: 9
          Pung: 9
  Pang: fim
main: Pang acabou com exit code 0
main: esperando Peng...
    Peng: fim
main: Peng acabou com exit code 1
main: esperando Ping...
        Ping: fim
main: Ping acabou com exit code 2
main: esperando Pong...
          Pong: fim
main: Pong acabou com exit code 3
main: esperando Pung...
            Pung: fim
main: Pung acabou com exit code 4
main: fim
```

3. Formato de Entrega

Todos os arquivos utilizados na implementação do trabalho devem ser entregues em um único arquivo .zip ou .tar.gz na atividade do moodle. Deve ser anexado um arquivo Makefile para compilar o código.

4. Data de Entrega

Data e horário da entrega estipulados na tarefa do moodle.

5. Avaliação

A avaliação se dará em 3 fases

1. Avaliação de compilação: compilar o código enviado. Caso haja erros de compilação, a nota do trabalho será automaticamente zerada.

2. Avaliação de execução: para validar que a solução executa corretamente sem falhas de segmentação. Caso haja falhas de segmentação, a nota é zerada. Será também avaliado o uso de variáveis globais (-5 pontos) e vazamentos de memória (-20%).

3. Avaliação da organização do código: busca-se nesta fase avaliar a organização do código orientado a objetos. Deve-se usar classes e objetos e não estilo de programação baseado em procedimentos (como na linguagem C).

Este trabalho precisará ser apresentado ao professor em horário a ser agendado. Plágio não será tolerado em nenhuma hipótese ao longo dos trabalhos, acarretando em nota 0 a todos os envolvidos.

6. Tempo Estimado

Estima-se um tempo total necessário para a conclusão desta atividade de 8 horas.