INE5412 – Sistemas Operacionais I
Profs. Giovani Gracioli e Márcio Bastos Castro
2023/01

Projeto de um Sistema Operacional em Nível de Usuário
Trabalho 5: Semáforos com Dormir e Acordar

1. Descrição

Este trabalho tem os seguintes objetivos:

1. Criação de uma classe **Semaphore** que irá implementar as operações p (sleep) e v (wakeup) de semáforos.

2. Implementar **novos métodos** na classe Thread para suportar as operações da classe **Semaphore**.

3. Implementar **métodos para incrementar e decrementar um inteiro de forma atômica**, também utilizados pelo semáforo.

Os seguintes métodos devem ser implementados na classe Semaphore:

- **void p();**

  Este método deve implementar a operação p (ou sleep) de um semáforo. Deve-se decrementar o inteiro do semáforo de forma atômica (utilizando fdec descrita abaixo) e colocar a Thread para dormir caso a mesma não conseguir acessar o semáforo (já existir em uso por outra Thread).

- **void v();**

  Este método deve implementar a operação v (ou wakeup) de um semáforo. Deve-se incrementar o inteiro do semáforo de forma atômica (utilizando finc descrita abaixo) e acordar uma Thread que estiver dormindo no semáforo.

- **int finc(volatile int & number); e int fdec(volatile int & number);**

  Implementa as operações de incremento e decremento no inteiro recebido como parâmetro de forma atômica. Para isso, deve-se utilizar linguagem assembly, mais especificamente a instrução xadd disponíveis nos processodores x86 (https://www.felixcloutier.com/x86/xadd) juntamente com a instrução lock para garantir a execução de xadd de forma atômica (https://www.felixcloutier.com/x86/lock).

  Para utilização de assembly dentro de um trecho de código C++, deve-se utilizar a macro ASM, conforme exemplificado em https://www.geeksforgeeks.org/c-asm-declaration/.

  Algo parecido com o exemplo abaixo:

ASM("lock xadd %0, %2"....);

- **void sleep(), void wakeup() e void wakeup_all();**

  O método sleep() deve colocar a Thread que não conseguir acessar o semáforo para dormir e mudar seu estado para WAITING (note que WAITING é diferente de SUSPENDED do trabalho trabalho anterior). O método wakeup() deve acordar uma Thread que estava dormindo no semáforo. O método wakeup_all() deve acordar todas as Thread que estavam dormindo no semáforo.

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

Perguntas para guiar o desenvolvimento da solução:

- Como controlar o "dormir" e "acordar" das Threads dentro da classe Semaphore?
- Como a classe Semaphore vai se relacionar com a classe Thread, já que todas as atividades relacionadas com Threads devem ser obrigatoriamente implementadas dentro da classe Thread? Dica: possivelmente deve-se adicionar novos métodos na classe Thread para suportar sleep(), wakeup() e wakeup_all().
- Quando wakeup_all() do Semaphore é chamado?
- Em qual classe deve-se implementar as instruções finc() e fdec() em assembly, já que as mesmas são dependentes do processador?

2. Arquivo Disponibilizado

Os seguintes arquivos foram disponibilizados neste trabalho:

- semaphore.h: contém a declaração dos métodos que devem ser implementados. A declaração dos atributos faz parte deste trabalho. Deve-se ainda criar um arquivo semaphore.cc para conter a implementação dos métodos.
- main_class.h e main_class.cc implementação de uma aplicação exemplo, usando Semaphore.

A saída esperada do programa exemplo é:

```
main: inicio
main: 0
main: 1
main: esperando Pang...
  Pang: inicio
  Pang: 0
     Peng: inicio
        Ping: inicio
           Pong: inicio
              Pung: inicio
  Pang: 1
  Pang: 2
  Pang: 3
  Pang: 4
  Pang: 5
  Pang: 6
```

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

Pung: 4
Pung: 5
Pung: 6
Pung: 7
Pung: 8
Pung: 9
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 4 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).

4. A quarta fase consiste na apresentação do trabalho em dia e horário agendado pelo professor. Durante as apresentações, o professor irá avaliar o **conhecimento individual dos alunos sobre os conteúdos teóricos e práticos vistos em aula e sobre a solução adotada no trabalho**. A nota atribuída à cada aluno *i* no trabalho ($NotaTrabalho_i$) será calculada da seguinte forma, onde A_i é a nota referente à apresentação do aluno *i* e *S* é a nota atribuída à solução do trabalho:

$$NotaTrabalho_i = \frac{A_i x S}{10}$$

Plágio não será tolerado em nenhuma hipótese ao longo dos trabalhos, acarretando em nota 0 a todos os envolvidos.