Laboratório 9
Concorrência com troca de mensagens em Go

Programação Concorrente (ICP-361)
Monitor: Pedro Mion
Prof. Silvana Rossetto

[1]Instituto de Computação/CCMN/UFRJ - 2024-2

Introdução

O objetivo deste Laboratório é praticar a programação concorrente com troca de mensagens, usando a linguagem Go. Para cada atividade, siga o roteiro proposto e responda às questões colocadas. Veja neste link um tutorial de apresentação da linguagem: https://gobyexample.com/

Atividade 1

Objetivo: Mostrar como criar um programa concorrente em Go usando goroutines.

Roteiro: Abra o arquivo **hello.go** leia o código do programa e verifique o que ele faz.

1. Execute o programa hello.go usando o comando `go run hello.go` e observe os resultados impressos na tela. Há mudanças na ordem de execução das goroutines?
2. Comente a chamada da função `Sleep` e avalie o resultado da execução. O que aconteceu e por que?

Atividade 2

Objetivo: Mostrar como usar *channels* para comunicação entre goroutines.

Roteiro: Abra o arquivo `channels.go`, leia o código do programa e verifique o que ele faz.

1. Execute o programa e observe o resultado impresso na tela.
2. Por que não foi necessário fazer uma chamada para a função `Sleep` no código da função `main` dessa vez?
3. Comente a linha `msg = <-str` e avalie o resultado da execução. O que aconteceu e por que?
4. Estenda o programa fazendo o fluxo principal e a goroutine implementarem o seguinte diálogo:

```
Main envia para Goroutine: 'Olá, Goroutine, bom dia!' (impresso pela Goroutine)
Goroutine envia para Main: 'Oi Main, bom dia, tudo bem?' (impresso pela Main)
Main envia para Goroutine: 'Tudo bem! Vou terminar tá?' (impresso pela Goroutine)
Goroutine envia para Main: 'Certo, entendido.' (impresso pela Main)
Goroutine imprime: finalizando
Main imprime: finalizando
```

Atividade 3

Objetivo: Mostrar exemplo de uso de *channels bufferizados* para comunicação entre goroutines e também para sincronização entre elas.

Roteiro: Abra o arquivo `channels1.go`, leia o código do programa e tente entender o que ele faz.

- Execute o programa e observe o resultado impresso na tela.
- Qual é a finalidade de cada um dos canais criados?
- O vai acontecer se comentarmos a linha 32?

Atividade 4

Objetivo: Mostrar um exemplo de uso do comando de seleção `select` para implememtar temporização de canais.

Roteiro: Abra o arquivo `select.go`, leia o código do programa e tente entender o que ele faz.

- Execute o programa e observe o resultado impresso na tela.
- Altere o código para que a sentença *result 1* seja impressa na tela.

Atividade 5

Objetivo: Usar os recursos de concorrência de Go, com **comunicação e sincronização via canais**, para resolver o problema de encontrar números primos em uma lista de entrada.

Descrição:

1. Identificar todos os **números primos** de uma sequência gerada pelo **fluxo principal** (de 1 a N) e retornar a **quantidade total de números primos encontrados**.
2. O fluxo principal deverá atribuir/enviar os números para serem testados de forma concorrente por M goroutines.
3. Use a função abaixo para verificar a primalidade de um número:

```
int ehPrimo(long long int n) {
      if (n<=1) return 0;
      if (n==2) return 1;
      if (n%2==0) return 0;
      for (int i=3; i<sqrt(n)+1; i+=2)
         if(n%i==0) return 0;
      return 1;
```

4. Não se esqueça de verificar a corretude da implementação.

Entrega: Disponibilize o código implementado nas **Atividade 2 e 5** e em um ambiente de acesso remoto (GitHub ou GitLab). **Use o formulário de entrega desse laboratório para enviar o link do repositório do código implementado e responder às questões colocadas.**