Laboratório de Programação Concorrente

# Lab4 - gogo - 24.1

# Objetivo

Neste laboratório, iremos implementar as funcionalidades dos laboratórios anteriores, desta vez em golang.

Para relembrar, tivemos três funcionalidades:

1. A soma do **sum** de todos os arquivos;
2. A lista de arquivos com o mesmo **sum;**
3. A similaridade parcial entre arquivos.

Para este laboratório, já fornecemos a implementação serial das duas primeiras funcionalidades (https://github.com/thiagomanel/fpc/tree/master/2024.1/lab4/go/serial).

Então, agora, você precisa implementar duas versões concorrentes (crie diretórios para cada uma delas):

- **concurrent-0** - Na qual, as duas primeiras funcionalidades são implementadas de modo concorrente (espera-se que executem mais rapidamente);
- **serial-partial** - Na qual, a terceira funcionalidade (similaridade parcial) é implementada de modo serial
- **concurrent-partial** - Na qual, a terceira funcionalidade (similaridade parcial) é implementada de modo concorrente.

# SPEC partial similarity

Um arquivo pode apresentar algum grau de similaridade (um valor entre 0 e 1) com outro arquivo. Quanto maior a quantidade de pedaços (ou chunks) de um arquivo iguais a pedaços do outro arquivo (iguais conforme a função **sum),** mais similar os arquivos são.

Seu programa deve receber uma lista de arquivos e irá retornar para cada arquivo sua similaridade em relação aos demais.

## Detalhes da entrega

Neste laboratório, não será obrigatório o uso do VIM. Ou seja, você pode usar a IDE que preferir (vscode, eclipse, intellij, etc).

Entretanto, as máquinas do LCC estarão restritas ao modo prova, ou seja, o acesso à internet será restrito.

O código desenvolvido será entregue através de um repositório no github. Para isso, no início da aula, crie um repositório **privado** (repositórios públicos não serão considerados). Adicione os professores como colaboradores (https://github.com/thiagomanel/).

**Além disso, é muito importante que você faça commits frequentes. Qualquer mudança relevante em uma função, deve ser comitada.**

Na saída padrão deve-se indicar a similaridade em um estilo semelhante ao output abaixo:

- Similarity between …/file.1 and …/file.2: 89.73469%
- Similarity between …/file.1 and …/file.3: 89.56112%
- Similarity between …/file.2 and …/file.3: 89.578285%

Você tem total liberdade para mudar o código base da maneira que achar melhor. P.ex, criando novas funções, alterando a assinatura das funções dadas bem como usando outras estruturas de dados.

## Prazo

10/set/24 14:00

## Appendix 0

Você pode criar um tipo em golang assim:

```
type MyADT struct {
      varname1 int
      varname2 boll
}
```

E, iniciá-lo da seguinte forma:

```
xpto := MyADT{3, false}
```

## Appendix 1

https://gobyexample.com
https://go.dev
https://go.dev/tour/list