Laboratório de Programação Concorrente

Lab1 - Warmup - 24.2

Objetivo

Neste laboratório, vocês precisarão melhorar o desempenho de um decifrador (que considera a cifra https://pt.wikipedia.org/wiki/ROT13). A cifra rot13 é um algoritmo de substituição simples que substitui cada caractere do texto original pelo caractere 13 posições à frente no alfabeto. Imagine por exemplo que você quer manter um arquivo na sua máquina com as senhas dos serviços online que você usa. Manter essas senhas anotadas em texto puro é uma péssima ideia, talvez você queira usar uma cifra simples para manter esses dados minimamente protegidos.

Temos duas implementações básicas (python e java) sequenciais deste algoritmo. Estas implementações básicas processam os arquivos sequencialmente, realizando o rot13 em cada um dos 100 arquivos de senhas fornecidos. O código atual realiza a transformação de forma sequencial para cada arquivo passado como parâmetro pelo usuário. O objetivo é usar múltiplas threads para processar os arquivos de forma concorrente, aplicando o rot13 a todos os arquivos e reduzindo o tempo total de execução.

A entrega, detalhada nas seções seguintes, envolverá o código fonte. Iremos avaliar tanto as possibilidades de plágio entre os alunos quanto a geração automática de código.

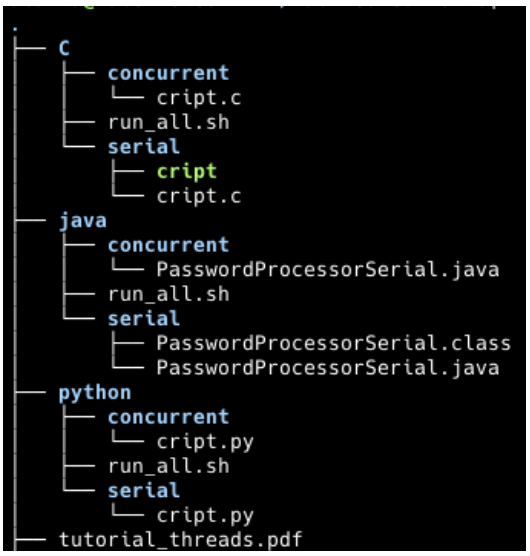
Prazo

28/11/24 às 18h

Visão geral do código base

https://github.com/thiagomanel/fpc/tree/master/2024.2/Lab1

O código deverá estar organizado na seguinte hierarquia:



Temos três diretórios principais: **lab1/python**, **lab1/java** e **lab1/c**. Dentro de cada um deles, há um subdiretório para a implementação serial, e possui um subdiretório concurrent que atualmente não possui código.

O objetivo é copiar os arquivos que necessários do diretório serial e criar um a implementação em concorrente no diretório concurrent. Lembre-se de seguir a estrutura dos diretórios e de garantir que o código esteja funcionando corretamente nas versões implementadas.

O subdiretório serial em cada linguagem já contém as implementações sem concorrência, enquanto o subdiretório concurrent já foi criado, mas ainda não possui as implementações. Durante o desenvolvimento do laboratório, você deverá implementar as versões concorrentes do código e armazená-las nos subdiretórios correspondentes: lab1/python/concurrent, lab1/java/concurrent e lab1/c/concurrent.

As implementações básicas (serial) e concorrentes são acompanhadas de um script Bash chamado run_all.sh, responsável por executar o código tanto serial quanto concorrente. Esse script está presente nos diretórios principais de cada linguagem (lab1/python, lab1/java, lab1/c) e pode ser usado para rodar tanto a versão serial quanto a versão concorrente do código da mesma linguagem.

Você deve garantir que as implementações serial e concorrente estejam funcionando corretamente e que o script run_all.sh execute ambas as versões conforme esperado.

Entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script submit-answer. Uma vez que você tenha concluído sua resposta, seguem as instruções:

1) Crie um arquivo *lab1_matr1_matr2.tar.gz* com todo o código produzido. Para isso, supondo que o diretório raiz de seu repositório privado chama-se lab1_pc, você deve executar:
   tar -cvzf lab1_matr1_matr2.tar.gz lab1_pc/
2) Submeta o arquivo lab1_matr1_matr2.tar.gz usando o script submit-answer (https://github.com/thiagomanel/fpc/blob/master/submit-answer/submit-answer.sh):
   *bash submit-answer.sh lab1 path/to/*lab1_matr1_matr2.tar.gz

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Alterações no código realizadas após o prazo de entrega não serão analisadas.