# Mini EP 3: Um curto experimento com *flags* e seus impactos.

Alfredo Goldman, Elisa Silva e Luciana Marques
*MAC 0219-5742 – Programação Concorrente e Paralela 2021*

Entrega até 10 de maio de 2021

## 1. Introdução

Neste mini EP vamos entender como o uso de diferentes *flags* leva a comportamentos diferentes. A ideia é compilar um programa com e sem *flags* de otimização e medir seu tempo de execução e compilação. Para dar um gosto a mais, vamos compilar um programa que interpreta programas.

Utilizaremos o código da linguagem *Lua 5.4.3* da PUC-RIO[1] modificado para compilar sem o uso de otimizações e uma mini coleção de *benchmarks* feita em cima do *The Computer Language Benchmarks Game*.

## 2. Tarefas

Se for um usuário Windows, a recomendação é seguir as próximas instruções usando o WSL Se for um usuário de macOS, note que o comando GCC é na verdade um *alias* para o Clang, mas pode usá-lo sem problemas. Certifique-se que possui o GCC e o Make instalados em sua máquina.

Faça o *download* do arquivo "lua-5.3.4-modificado.zip" disponível no e-Disciplinas na página deste mini EP e o descomprima.

**Você deve compilar o "lua"(medindo o seu tempo de compilação) e depois medir 5 vezes o tempo que leva para o "lua" executar o programa "bench.lua" disponível na pasta. Faça isso para a versão não otimizada e a versão otimizada. Depois redija um curto parágrafo sobre os números obtidos em um arquivo ".txt".**

### 2.1 Como compilar o "lua"?

Para compilar o "lua", basta estar dentro da pasta descomprimida e rodar o comando "`make -B`" (o "`-B`" garante que o "`make`" compile todos os arquivos, mesmo aqueles já compilados). Se tudo correr bem, o programa "lua" vai estar disponível dentro da pasta "src".

Para medir o tempo de compilação, é só colocar o "`time`" na frente: "`time make -B`".

### 2.2 Como rodar o bench.lua?

Assumindo que está na pasta descomprimida, é só executar o programa gerado, passando o caminho do arquivo a ser executado como parâmetro: "`./src/lua bench.lua`". Para medir o tempo você já sabe (anotem o tempo do "user").

### 2.3 Como alterar as *flags* do GCC?

As *flags* passadas para o GCC estão no arquivo "src/Makefile" na linha que começa com "CFLAGS= -O0 -Wall..". O "-O0" é a flag que desativa as otimizações do compilador, apague-a e insira outras flags para otimizar o programa.

[1] Se você não sabe o que é Lua, já passou da hora de descobrir!

**2.4 O que é otimizar no contexto deste mini EP?**

A otimização deste mini EP está baseada apenas na mudança das *flags* do compilador. Vide a seção 5 para descobrir quais *flags* usar e a seção 2.3 para entender como usar elas.

**2.5 Estatística**

Diferente dos mini EP anteriores, não é para entregar os tempos de execução e sim sua média aritmética e a variância. Como não sabemos a média da população, iremos usar a variância da amostra usando um estimador não enviesado. Isso quer dizer que a variância é dada pela soma do quadrado da diferença das medidas pela média, tudo dividido pelo número de medidas - 1.

**2.6 O que se pode comentar sobre os números obtidos?**

O comentário é livre, mas para quem estiver querendo uma pergunta inicial para ajudar a escrever o parágrafo, segue algumas:

- Só existem pontos positivos em usar *flags* de otimização?
- Faz sentido usar *flags* de otimização durante todo o desenvolvimento de um programa?
- Como o compilador pode aumentar o desempenho de um programa sem que seja necessário alterar o código fonte?
- As *flags* impactam apenas no tempo de execução e compilação? E o tamanho do binário gerado? Como eles se relacionam?

**3. Entrega**

Envie os resultados obtidos com informações da sua máquina no seguinte formulário: https://forms.gle/hP52QuMeWnsm4mzJ8**.**

É necessário que entre com seu email USP, bem como será possível editar sua resposta depois do envio.

Envie o .txt contendo suas observações sobre os números obtidos pelo eDisciplinas da matéria.

Entrega até 10 de maio de 2021.

**4. Critério de Avaliação**

Os Mini EPs usam um critério de avaliação binária (ou 1 ou 0). Para tirar 1 envie o formulário com os resultados dos experimentos e submeta no eDisciplinas o texto.

Vale reforçar parágrafo II do artigo 23 do **Código de Ética da USP**:

> Artigo 23 - É vedado aos membros do corpo discente e demais alunos da Universidade:
> [...]
> II. lançar mão de meios e artifícios que possam fraudar a avaliação do desempenho, seu ou de outrem, em atividades acadêmicas, culturais, artísticas, desportivas e sociais, no âmbito da Universidade, e acobertar a eventual utilização desses meios.

Mini EPs plagiados receberão nota 0.

Se tiver dúvidas, envie uma mensagem no fórum do curso ou envie e-mails para elisa@silva.moe, lucianadacostamarques@gmail.com ou gold@ime.usp.br com *[miniEP3]* no assunto do e-mail. Divirta-se!

**5. Extras**

Compiladores possuem uma lista grande de *flags*. Se conhecer todas já é difícil, entender a interação entre elas é difícil o bastante para automatizarem e virar paper (ex: https://arxiv.org/pdf/1703.08228.pdf). Mas na disciplina de programação paralela é interessante entender o que significa usar uma *flag* como `-O3` ou `-fno-inline` afeta a geração e o funcionamento dos programas. Assim recomendamos acessar essa página do manual do GCC: https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html.

Recomendamos a leitura sobre otimização do GCC da *Wiki do Gentoo* (https://wiki.gentoo.org/wiki/GCC_optimization) que fala um pouco sobre as *flags* mais importantes para ganho de desempenho, tanto em execução quanto em compilação.

Se a parte de estimador não enviesado causou alguma estranheza, a página da Wikipédia sobre variância é um bom lugar para refrescar os conhecimentos estatísticos: https://pt.wikipedia.org/wiki/Vari%C3%A2ncia.