

# MAC 0219/5742 – Introdução à Computação Concorrente, Paralela e Distribuída

Prof. Dr. Alfredo Goldman  
EP1 - Parallel Grep versão 1.0

Monitores: Giuliano Belinassi e Matheus Tavares

## 1 Introdução

O `grep` é uma ferramenta para encontrar linhas em arquivos de texto que estejam dentro de uma linguagem gerada por uma expressão regular. Segundo Kernighan<sup>1</sup>, o `grep` foi desenvolvido em assembler de PDP-11 durante uma madrugada por Ken Thompson, utilizando o código fonte do editor de texto `ed` como base. Este programa possibilitou que Lee McMahon fizesse as análises de texto de sua pesquisa.

Neste exercício programa, seu objetivo é reimplementar o `grep` com um adendo: ele deve ser capaz de aproveitar o paralelismo dos processadores modernos.<sup>2</sup>

## 2 Requisitos

Seu EP deve ser capaz de atender os seguintes requisitos:

1. Ler vários arquivos e testar correspondência com a expressão regular fornecida em cada linha dos arquivos, utilizando paralelismo. **Você deverá implementar o paralelismo particionando o conjunto de arquivos entre as *threads*.**
2. Não misturar as saídas de arquivos diferentes, isto é, não pode haver linhas de um arquivo *B* entre as linhas de um arquivo *A*, como no `grep` original.
3. Aceitar uma expressão regular<sup>3</sup> POSIX para pesquisa.
4. Aceitar o caminho para um diretório e recursivamente filtrar os arquivos contidos no diretório fornecido e subdiretórios. (Como a opção `-r` do `grep` faz.)
5. Aceitar uma opção que diz quantas *threads* o programa poderá utilizar, no máximo. Você tem liberdade de escolher um número mínimo de *threads* constante e indicar um erro se o número fornecido for menor que o mínimo que você especificou.
6. Conter um `Makefile` que gera o binário `pgrep`. Este binário deve aceitar como argumento a seguinte linha de comando:

```
pgrep MAX_THREADS REGEX_PESQUISA CAMINHO_DO_DIRETÓRIO
```

---

<sup>1</sup><https://www.youtube.com/watch?v=NTfOnGZUZDk>

<sup>2</sup>A boa notícia é que você tem muito mais tempo do que o Ken Thompson teve ☺

<sup>3</sup>[https://pt.wikipedia.org/wiki/Express%C3%A3o\\_regular](https://pt.wikipedia.org/wiki/Express%C3%A3o_regular)

Você não precisa sanitizar os parâmetros de entrada, i.e., é garantido que `MAX_THREADS`  $\in \{1, 2, \dots, 128\}$ , `REGEX_PESQUISA` é uma expressão regular POSIX válida e `CAMINHO_DO_DIRETÓRIO` é uma string para um diretório existente e com permissão de leitura.

7. Todo o paralelismo deve ser feito usando `Pthread`.
8. O programa deverá imprimir na saída padrão (`stdout`) uma linha para cada linha dos arquivos lidos em que houver correspondência com a expressão regular fornecida. Estas linhas impressas devem ser no formato

`nome_do_arquivo: numero_da_linha_correspondida`

A contagem de linhas deve começar no 0. Exemplo de saída:

```
xpto.c: 42
xpto.c: 84
foo.s: 1337
bar.hs: 0
```

Por fim, seu EP não precisa gerar uma saída colorida.

### 3 Dicas

1. Simule as coisas no papel antes de codificar. Isso ajuda a identificar problemas de concorrência que podem vir a acontecer, e assim fica mais fácil projetar o seu EP com isto em mente.
2. Para iterar sobre os conteúdos de um diretório você pode achar as funções `opendir`, `readdir` e `closedir` interessantes.
3. Para criar e testar correspondência de expressões regulares, você pode usar as funções do cabeçalho `regex.h`.
4. Como você vai dividir o trabalho entre as *threads*?
5. Como você vai matar as *threads* quando não houver mais trabalho para elas?
6. Como você vai garantir que as linhas não sejam impressas de maneira misturada?
7. Tenha certeza que você domina os seguintes conceitos: *Mutex*, *Semáforos*, *Barreiras*, *Filas*, e o *Problema do Produtor-Consumidor*.
8. Deu `SEGFAULT`? Use o `gcc` com a opção `-g` e use o `gdb` para ver onde está o erro!
9. A ferramenta `valgrind` pode te ajudar a encontrar problemas de concorrência.

## 4 Entrega

Deverá ser entregue um pacote no sistema PACA com uma pasta com o nome e o sobrenome do estudante que o submeteu no seguinte formato: `nome_sobrenome.zip`. Se o EP for feito em trio, o formato deve ser:

`nome1_sobrenome1.nome2_sobrenome2.nome3_sobrenome3.zip`

Somente um estudante do time submeterá a tarefa. Essa pasta deve ser comprimida em formato ZIP e deve conter dois itens:

- Os códigos fonte do programa, em conjunto com um `Makefile` que o compila, gerando o executável especificado.
- Um arquivo `.txt` ou `.pdf` com o nome dos integrantes, e uma breve explicação sobre a sua solução e desafios encontrados. Se você projetou seu código para funcionar com um número mínimo de *threads*, informe também neste arquivo, com breve justificativa. Imagens também podem ser inseridas no arquivo. Relatórios em `.doc`, `.docx` ou `odt` **não** serão aceitos.

Em caso de dúvidas, use o fórum de discussão do Paca. A data de entrega deste Exercício Programa é até às **16:00h do dia 11 de Abril**. Não se esqueça que o professor irá sortear um aluno para explicar sua implementação do EP neste mesmo dia. Não é necessário preparar slides, apenas falar brevemente sobre o código e as decisões de projeto tomadas.

*Boa Sorte!*