Sistemas Operacionais - 2017/2

Trabalho Prático T1

9 de outubro de 2017

1 Objetivo

O objetivo deste trabalho é implementar uma variação distribuída do Crivo de Eratóstenes (em inglês, *Sieve of Eratosthenes*), um algoritmo simples e prático para encontrar números primos no intervalo entre 2 e um dado valor limite n.

2 Descrição do Problema

Você deverá implementar um *Sieve of Eratosthenes* usando *threads*. Para tal, use um programa com um processo e várias *threads*.

- Utilize 8 *threads* de filtro (*sieve*) para testar se um número é primo ou não.
- Utilize uma *thread* para acumular os resultados da computação e imprimí-los na tela.
- A *thread* principal do seu programa fica responsável por gerar os números a serem testados até um certo limite n.

As seguintes restrições se aplicam:

- As *threads* de *sieve* ficam organizadas em anel, e a comunicação entre *sieves* é feita somente para a seguinte no anel. Assim, a *sieve* 3 se comunica somente com a *sieve* 4 e a *sieve* 7 somente com a *sieve* 0.
- Não pode haver nenhuma comunicação entre qualquer *thread* de *sieve* e a *thread* principal, exceto quando a *thread* principal insere um número para teste na rede de *sieves*.
- Toda a comunicação entre *threads* deve ser feita através de um *buffer* limitado a m números. Todos os *buffers* devem ser protegidos por seções críticas, de forma a garantir a sua consistência.
- As *threads* devem realizar o máximo de comunicação assíncrona possível, exibindo assim o máximo grau de paralelismo. Isso implica na necessidade de admissão simultânea de vários inteiros para serem testados na rede de *sieves*.

O funcionamento do crivo deve ser da seguinte forma:

- A *thread* principal cria as demais *threads* e a seguir entra em *loop*, gerando um a um os números no intervalo 2 até n.
- Ao gerar o número i, a *thread* principal o envia para a *sieve* 0, colocando i no *buffer* da *sieve*. Isso feito, dizemos que i foi inserido na rede de *sieves* para ser testado.
- Cada uma das *sieves* possui uma área de armazenamento (além do *buffer*) para guardar os primos que a *sieve* for identificando.
- Os números em teste na rede circulam da seguinte forma:

  - *Buffers* seguem uma política FIFO. Uma *sieve* retira o primeiro número i do seu *buffer* e testa i contra um dos primos já armazenados na *sieve*.
  - Se i é divisível por algum dos primos da *sieve*, ela envia essa informação para a *thread* de resultados.
  - Se i não é divisível por nenhum dos primos da *sieve*, ela propaga i para a próxima *sieve* do anel para que o número continue sendo testado. Se a rede testar i contra todos os primos anteriores à ele, o número i é declarado primo e armazenado em uma das *sieves*. A *sieve* que armazenou o número envia essa informação para a *thread* de resultados.
- O programa termina quando todos os números gerados pela *thread* principal forem testados pela rede.

O trabalho será testado da seguinte maneira:

`./trab1 n m`

onde `n` é o número máximo que deve ser testado e `m` é o tamanho do *buffers* entre as *threads*.

A *thread* de resultados deve imprimir as informações sobre cada número testado, **EM ORDEM**. Assim, se o número for primo ele deve ser indicado como tal, juntamente com a *sieve* aonde ele ficou armazenado. Se o número não for primo, essa informação também deve ser impressa. Dessa forma, por exemplo, se executarmos `./trab1 10 2`, teremos como resposta no terminal:

```
2 is prime (stored in sieve 0)
3 is prime (stored in sieve 1)
4 divided by 2 at sieve 0
5 is prime (stored in sieve 2)
6 divided by 2 at sieve 0
7 is prime (stored in sieve 3)
8 divided by 2 at sieve 0
9 divided by 3 at sieve 1
10 divided by 2 at sieve 0
```

A saída do seu programa deve ser exatamente (no mesmo formato) como no exemplo acima.

Seu programa deve ser, obrigatoriamente, compilado com o utilitário `make`. Crie um arquivo `Makefile` que gera como executável para o seu programa um arquivo de nome `trab1`.

3 Regras para Desenvolvimento e Entrega do Trabalho

- **Data da Entrega**: O trabalho deve ser entregue até às 23:55 h do dia 10/11/2017 (6a. feira). Não serão aceitos trabalhos após essa data.
- **Prazo para tirar dúvidas**: Para evitar atropelos de última hora, você deverá tirar todas as suas dúvidas sobre o trabalho até o dia 08/11/2017. A resposta para dúvidas que surgirem após essa data fica a critério do professor.
- **Grupo**: O trabalho é **individual**.
- **Linguagem de Programação e Ferramentas**: Você deverá implementar o seu trabalho em C, usando a biblioteca Pthreads.
  - Tutorial de Pthreads: `https://computing.llnl.gov/tutorials/pthreads/`.
  - Manual do Make: `http://www.gnu.org/software/make/manual/make.html`.
  - O seu trabalho será corrigido no Linux.
- **Como entregar**: Pela atividade criada no AVA. Envie um arquivo compactado com todo o seu trabalho.
- **Recomendações**: Modularize o seu código adequadamente. Crie códigos claros e organizados. Utilize um estilo de programação consistente. Comente o seu código extensivamente. Não deixe para começar o trabalho na última hora.

4 Avaliação

- O trabalho vale 2.5 pontos na média parcial do semestre.
- Trabalhos com erros de compilação receberão nota zero.
- Caso seja detectado plágio (entre alunos ou da internet), todos os envolvidos receberão nota zero.
- Serão levadas em conta, além da correção da saída do seu programa, a clareza e simplicidade de seu código.
- A critério do professor, poderão ser realizadas entrevistas com os alunos, sobre o conteúdo do trabalho entregue. Caso algum aluno seja convocado para uma entrevista, a nota do trabalho será dependente do desempenho na entrevista. (Vide item sobre plágio, acima.)

5 Curiosidade

A reportagem em `http://www.wired.com/wiredscience/2013/11/prime/all/` descreve os últimos avanços na pesquisa sobre números primos. O texto cita o uso de *sieves*, embora de um tipo diferente.