**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

**Bacharelado em Ciência da Computação**

**Laboratório de Sistemas Operacionais**

**Professor Gustavo Maciel Dias Vieira**

***Campus* Sorocaba**

**Projeto 4**

**Programação Concorrente**

Rafael Paschoal Giordano - 408298

Thales Gonçalves Chagas - 408557

**Sorocaba**

**2014**

**1 Introdução**

O projeto 4 têm como objetivo utilizar programação concorrente para diminuir o tempo de execução de um programa que quebra senhas por meio de busca exaustiva.

**2 Descrição**

O projeto consiste em 4 tarefas:

1. Baixar, compilar e executar o programa exemplo fornecido. Estudar o seu funcionamento.

2. Medir o tempo que este programa leva para quebrar o hash “aaRCVPtrkrWUY”.

3. Alterar o programa, adicionando suporte para múltiplas threads.

4. Medir o tempo que o programa multithreaded leva para quebrar o hash “aaRCVPtrkrWUY”.

Explicações abaixo:

1. A partir da execução do programa fornecido, foi notado que este utiliza força bruta para quebrar uma senha de quatro caracteres. Sendo a execução não multithreaded de desempenho baixo, levando um tempo considerável para terminar a execução.
2. Os teste foram realizados em um computador com processador Intel® Core™ i7-2670QM CPU @ 2.2GHZ no sistema operacional fedora 20 de 64 bits. O programa fornecido demorou 0m48.894s para quebrar o hash “aaRCVPtrkrWUY” correspondente a senha “raio”.
3. Para a alterar o programa para suportar múltiplas threads, foi utilizada a organização produtor/consumidor fornecida adaptada, assim o produtor gera as senhas e o consumidor testa as senhas geradas. Foram criadas variáveis globais para que a comunicação entre as threads ocorra de forma facilitada, foi utilizado um vetor de 50 posições que armazena as senhas produzidas a serem consumidas também um ponteiro para armazenar a hash que deve ser procurada, além dos controladores das threads como passado no exemplo de produtor/consumidor.

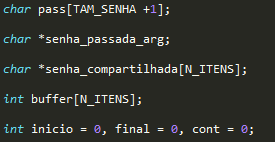


Figura - Variáveis globais utilizadas.

Foi utilizado na execução uma thread para produção e quatro threads para consumir, pois a produção é bem mais rápida do que o consumidor e esta dá conta de suprir para as quatro threads consumidoras.

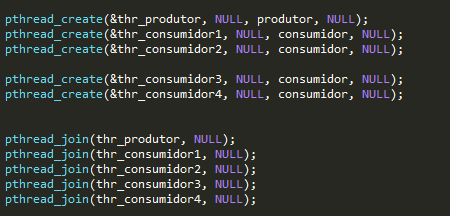


Figura – Threads criadas e utilizadas.

1. O tempo obtido ao executar novamente o programa com as novas threads rodando foi de 0m13.753s, para quebrar o hash “aaRCVPtrkrWUY” correspondente a senha “raio”. Executado no mesmo processador mencionado anteriormente.

**3 Dificuldades**

A maior dificuldade encontrada no projeto foi controlar a execução das threads geradas, para garantir que estas executem concorrentemente e sem erros(uma thread tentar consumir mais do que havia sido produzido). A criação e utilização das threads foi de simples entendimento.

**4 Conclusão**

Programas multithreaded são mais ágeis por terem partes de seu código executando de maneira concorrente, aumentando a agilidade do processamento dos dados.