Trabalho
Agenda Telefônica

CI215 – Sistemas Operacionais
Primeiro Semestre de 2012
Prof. Bona

1. Descrição do Problema

O Prof. Giacomo Casanova tem problemas com a agenda telefônica de seu celular. Devido ao altíssimo número de pessoas armazenadas em sua lista e ao grande número de buscas e inserções realizadas diariamente, Casanova passa boa parte de seu dia aguardando pelo processamento de seu Android.

Você deve ajudá-lo. O seu objetivo neste trabalho é implementar um tabela *hash multi-threaded*, que armazene e busque informações telefônicas de forma paralela e eficiente. Cada linha de uma tabela *hash* consiste em uma *chave* e um *valor*. À chave, ou o nome do contato, é aplicada uma função *hash* que a transforma no índice da tabela. O valor, ou número do telefone, é então armazenado na linha da tabela adequada.

2. Dados de Entrada

As operações a serem realizadas sobre a tabela são descritas em um arquivo de entrada. Uma linha especifica uma operação. Um exemplo de um arquivo de entrada pode ser visto abaixo:

```
PUT José 33333636
PUT João 33443434
PUT Joaquim 32222222
GET José
GET Joaquim
DELETE João
PRINT
PUT Josiney 33545454
```

Cada linha do arquivo de entrada tem o seguinte formato:

OP CHAVE [*VALOR*]

Onde:

- OP descreve o tipo da operação. Pode assumir os seguintes valores:
  - PUT - Insere um valor na agenda
  - GET - Retorna o valor associado a uma chave
  - DELETE - Remove um contato da agenda
  - PRINT – Sincroniza os operações realizadas até então, e imprime o estado atual da agenda.
- CHAVE é o nome do contato
- O campo VALOR é opcional e existe apenas nas operações do tipo PUT, especificando o valor associado à chave (telefone do contato).

3. Saída do Programa

A cada execução de uma operação GET, o nome do contato e seu respectivo telefone devem ser impressos no arquivo de saída. Além disso, a cada operação PRINT o programa deve fornecer uma listagem com todos os telefones inseridos na agenda, precedidos pela string “Início agenda” e terminados pela string “Fim agenda”. A saída do exemplo acima seria:

```
José 33333636
Joaquim 32222222
Início agenda
33333636
32222222
Fim agenda
```

Observações:

1. A saída do comando PRINT deve ser ordenada pelo *hash* do nome do contato.
2. Na implementação *multi-threaded*, preserve a ordem das operações.
3. Sua saída deve ser exatamente como a especificada pois a saída de seu programa será validada com o resultado esperado utilizando o comando **diff** .

4. Sobre a implementação

Você deve considerar que para resolver este problema você tem a disposição uma máquina com múltiplos núcleos que deve ser programada utilizando **Linguagem C** e a **biblioteca Pthreads** (**libpthreads**). Não devem ser usadas bibliotecas mais abstratas que escondam a libpthreads ou mesmo as estruturas de dados utilizadas.

Você deve tomar cuidado para garantir a correta sincronização das operações; caso contrário, os resultados apresentados podem acabar contendo mais ou menos telefones e/ou alterando a ordem esperada da saída do programa.

Será fornecida uma implementação sequencial do programa utilizando estruturas de dados simples, assim como a função de *hash* a ser utilizada. Seu programa deve estar preparado e otimizado para aceitar grandes arquivos de entrada, contendo um grande número de operações.

5. Relatório

Você deve apresentar um relatório (em Latex) contendo:

1. Introdução
2. Arquitetura do Sistema
3. Experimentos
4. Conclusão

Na arquitetura você deve descrever a arquitetura do seu software, apresentando os diferentes grupos de threads, as principais estruturas de dados, bem como as técnicas de sincronização utilizadas. Figuras e diagramas são úteis nesta parte. Mas não deixe de usar texto para justificar e defender a arquitetura proposta.

Os experimentos devem apresentar execuções do programa com diferentes parâmetros para o número de threads dos pools e outros parâmetros considerados relevantes. Também é desejável resultados de experimentos realizados com versões iniciais do seu programa, supostamente mais lentas. Experimentos com diferentes cargas (conjunto distintos de arquivos a serem copiados) também são desejáveis. A ideia é que os experimentem mostrem como as decisões de projeto melhoraram o desempenho do algoritmo. Você deve tentar demonstrar que teu programa é realmente eficiente.

Importante: Não menospreze o relatório, ele é parte da nota e importante. Como ele inclui experimentação você deve terminar a implementação antes da data limite para poder trabalhar no relatório final.

6. Apresentação

Além do relatório, no dia da apresentação será realizado um teste de funcionamento e desempenho do sistema. Todos os trabalhos serão executados utilizando o mesmo arquivo de entrada em uma máquina com 8 núcleos (sujeito a disponibilidade do HW). A não execução ou execução com erros, pode, potencialmente, gerar problemas na agenda telefônica e constrangimentos ao Prof. Casanova, sendo sua nota AUTOMATICAMENTE (0) ZERO. Um tempo de execução máximo (baseado na versão sequencial do programa) também será adotado. Caso o tempo de execução seja maior que este tempo máximo, AUTOMATICAMENTE implicará em nota (0) ZERO.

Lembre-se, quanto menor o tempo de execução, maior será sua nota, não desconsiderando a corretude e a qualidade da apresentação e do relatório.

7. Instruções para Entrega

Enviar para o e-mail bona@inf.ufpr.br até o deadline um tar.gz contendo:

\src (com o codigo fonte e makefile)
\relatorio (com o relatorio em pdf)
\equipe.txt (nome da equipe e e-mails)
\instrucoes.txt (explicando como o programa deve ser executado, em especial qual os ajustes para rodar em uma maquina de 8 núcleos)