DCC031 – Redes de computadores

Trabalho prático 0 – Calc Server

Professor: Daniel Fernandes Macedo

Data de entrega: 25/04/2016

Trabalho em grupos de dois alunos.

Valor do trabalho: 5 pontos

1. Introdução

O objetivo deste trabalho é praticar o desenvolvimento de programas que se comunicam através de redes de computadores com a utilização de datagramas do tipo UDP (sem conexão).

A aplicação a ser desenvolvida é uma calculadora via rede que deve oferecer as seguintes operações:

- soma, subtração, multiplicação, divisão, resto, exponencial, raiz quadrada e fatorial.

2. Programa a ser desenvolvido

Dois programas deverão ser desenvolvidos:

Cliente

A aplicação cliente funcionará como uma calculadora normal do ponto de vista do usuário, mas haverá uma diferença fundamental em sua implementação – todas as operações deverão ser executadas em um servidor.

Alguns aspectos importantes deverão ser levados em conta durante o desenvolvimento do cliente:

- Entrada dos dados pelo usuário: Deve-se criar uma uma sequência de instruções que facilite a utilização do programa (Ex: menu com operações disponíveis). Use a criatividade para permitir uma forma fácil de entrada de dados para todos os tipos de operações. A operação de soma, por exemplo, deverá receber ser realizada para dois números digitados pelo usuário enquanto a operação fatorial receberá apenas um inteiro. Seu programa deve suportar todas as operações atendendo as características específicas de cada uma.

Servidor

O servidor deverá processar as operações requisitadas por clientes e retornar a resposta. Além disso, o servidor deverá ter uma opção para fornecer um relatório de quantas operações de cada tipo foram efetuadas no total. O relatório deverá ser simples contendo em cada linha o tipo de operação e a quantidade total de operações

daquele tipo já realizadas. Note que não é preciso contar a quantidade de operações por clientes.

Execução dos programas

Ambos vão receber dois parâmetros de entrada, como mostrado a seguir:

cliente <ip/nome> <porta>

servidor <porta>

O primeiro programa, o cliente, irá conectar no servidor definido pelo endereço IP (ou nome, por exemplo *login.dcc.ufmg.br*) e porta passados por parâmetro.

Já o servidor irá executar um serviço, que necessariamente irá tratar várias solicitações e só sair depois que o usuário apertar control-c. A porta a ser escutada é dada pelo parâmetro único do programa.

3. Detalhes da comunicação

A comunicação deverá ser feita utilizando-se o protocolo UDP em nível de transporte e os dados trocados entre cliente e servidor deverão enviados no formato ASCII. Uma operação de adição, por exemplo, poderia ser enviada como um texto na forma 15+55. Os operandos poderão, eventualmente, conterem valores negativos. Note que os dados trocados na comunicação podem ter um formato definido pelos integrantes do grupo, desde que contenham apenas caracteres ASCII.

4. Entrega do código

O código e a documentação devem ser entregues em um arquivo Zip (não pode ser RAR nem .tgz nem .tar.gz) no Moodle, contendo um arquivo **readme.txt** com o nome dos integrantes, e um arquivo PDF da documentação. Incluam todos os arquivos (.c, .h, makefile, não incluam executáveis ou arquivos objeto) EM UM ÚNICO DIRETÓRIO. Um makefile deve ser fornecido para a compilação do código. Parte desse trabalho envolve o aprendizado de como construir um makefile e utilizar a ferramenta Make [3]. Este makefile, quando executado sem parâmetros, irá gerar os dois programas, *calc_cliente* e *calc_server*, EXATAMENTE com esses nomes. Submissões onde os programas não seguem as especificações de parâmetros e nomes, makefiles não funcionando ou arquivos necessários faltando não serão corrigidos. Programas que não compilarem também não serão corrigidos. O julgamento do trabalho será feito em um computador rodando o SO Linux.

Os programas desenvolvidos deverão rodar em um computador **Linux**. Sugiro fortemente a execução em um computador Linux mesmo para o TP0 (os trabalhos subsequentes DEVERÃO executar em Linux, e ocorrem diferenças em plataformas diferentes quando executamos programas de rede).

5. Documentação

A documentação, além de ser entregue com o Zip junto ao código, deve ser entregue impressa ao professor até o dia seguinte à entrega, na sala de aula. Caso não tenhamos aula no dia seguinte ao prazo para envio, iremos combinar antecipadamente qual é a

forma mais conveniente de entrega da documentação. Trabalhos que forem entregues no Moodle mas sem a documentação impressa **não serão** corrigidos.

O texto da documentação deve ser breve, de forma que o corretor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado
- Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, e procedimentos e as decisões de implementação
- Decisões de implementação que porventura estejam omissos na especificação
- Testes, mostrando que o programa está funcionando de acordo com a especificação, seguidos da sua análise
- Conclusão e referências bibliográficas

Para este trabalho, a documentação deve conter exemplos de execução dos dois programas, mostrando a linha de comando a ser executada, um arquivo de entrada simples e a sua saída.

6. Avaliação

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc).
- Se o código executa as funções esperadas e da forma definida na especificação do trabalho.
- Execução correta do código em entradas de testes, a serem definidas no momento da avaliação. As entradas de teste irão exercitar a funcionalidade completa do código e testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto.
- Conteúdo da documentação, que deve conter os itens mencionados anteriormente.
- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua, qualidade textual e facilidade de compreensão).

Como mencionado anteriormente, trabalhos fora da especificação (por exemplo, sem makefile, que não gerem programas com os nomes especificados, que não compilem ou não possuam os parâmetros esperados) não serão corrigidos. Trabalhos fora da especificação tomam muito trabalho do corretor, que deve entender como compilar e

rodar **cada programa avaliado**, tempo esse que deveria ser empregado para avaliar a execução e corretude do código.

7. Desconto de nota por atraso

Os trabalhos poderão ser entregues até a meia-noite do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema Moodle, ou seja, a partir de 00:01 do dia seguinte à entrega no relógio do Moodle, os trabalhos já estarão sujeitos a penalidades.

A fórmula para desconto por atraso na entrega do trabalho prático ou resenha é:

Desconto = 2^{d-1}/0.32 %

onde *d* é o atraso em dias úteis. Note que após 5 dias úteis, o trabalho não pode ser mais entregue.

8. Referências

Programação em C:

- http://www.dcc.ufla.br/~giacomin/Textos/tutorialc.pdf
- ftp://ftp.unicamp.br/pub/apoio/treinamentos/linguagens/**c**.pdf
- http://www.cs.cf.ac.uk/Dave/C/CE.html
- http://www2.its.strath.ac.uk/courses/c/
- http://www.lysator.liu.se/c/bwk-tutor.html

Uso do programa make e escrita de Makefiles:

- http://comp.ist.utl.pt/ec-aed/PDFs/make.pdf
- http://haxent.com.br/people/ruda/make.html
- http://informatica.hsw.uol.com.br/programacao-em-c16.htm
- http://www.gnu.org/software/make/manual/make.html
- http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/