Universidade Federal de Minas Gerais
DCC023: Redes de Computadores
Trabalho Prático (TP0)

Introdução

Este é um trabalho simples, introdutório, apenas para permitir um primeiro contato com a programação em sockets.

Neste trabalho iremos desenvolver duas aplicações. Uma aplicação cliente que se conecta a um servidor e modifica um contador global, e uma aplicação servidora que controla o contador global. Qual aplicação você irá desenvolver depende do seu número de matrícula (ver entrega e avaliação).

Trabalho **individual**
Valor: 5 pontos
Data de entrega: 27 de março às 13:00 – não serão aceitos trabalhos atrasados!
Linguagem: C ou Python (todos devem ser capazes de comunicar com todos).

Objetivos

- Introduzir a interface de programação de soquetes POSIX.
- Introduzir os conceitos de aplicação cliente e aplicação servidor.

Execução

- O trabalho é individual e vale 5 pontos.
- A data de entrega está disponível no Moodle ou no Plano de Curso.

Especificação

Programa Cliente

O programa cliente irá se conectar ao servidor [socket, bind, connect]. Após estabelecimento da conexão, o cliente irá enviar um (1) byte contendo os caracteres “+” ou “-”, requisitando o incremento ou decremento do contador global (módulo 1000, de forma que o contador só possa assumir valores entre 0 e 999) [send]. Após o envio da requisição, o cliente irá esperar do servidor um inteiro de 4 bytes codificado em network byte order [htons/ntohs] contendo o valor do contador global [recv]. Após recebimento do contador global, o cliente deverá reenviar o valor recebido para o servidor em três caracteres numéricos (em formato ASCII) [sprintf], confirmando o recebimento e processamento da mensagem [send]. Após o envio da comunicação, o cliente de fechar a conexão com o servidor [close].

Programa Servidor

O programa servidor irá esperar a conexão de clientes [socket, bind, listen, accept]. Após o estabelecimento de uma conexão, o servidor deverá receber um (1) byte de requisição [recv]. Após o recebimento da requisição, o servidor deverá processá-la e enviar o valor próximo valor do contador global para o cliente [send]; o valor do contador deve ser enviado como um inteiro de 4 bytes codificado em network byte order [htons/ntohs]. Após o envio do próximo valor do contador, o servidor deverá esperar três bytes contendo a confirmação do valor do contador pelo cliente [recv]. Após o recebimento da confirmação (ou disparo do temporizador), o servidor deve fechar a conexão com o cliente [close]. Se o valor confirmado pelo cliente for o próximo valor do contador [atoi], o servidor atualiza o contador global.

Detalhes de Implementação

- O servidor deverá escutar no endereço IP 127.0.0.1 (representando a máquina local em que o programa executa) e no porto 51515.
- Os caracteres '+' e '-' devem ser transmitidos em 1 byte usando codificação ASCII.
- Clientes e servidores devem configurar um temporizador (*timeout*) para detectar falhas de comunicação ao chamar a função [recv]. A configuração de temporizador é feita chamando-se a função [setsockopt]. No Linux, em C, as opções disponíveis para uso na função [setsockopt] são descritas na seção [socket] do manual 7 (acesse usando [man 7 socket]); procure por [SO_RCVTIMEO]. Em Python, o socket tem uma opção [settimeout].

Entrega e Avaliação

Você deve entregar o código fonte do seu programa. Se o seu número de matrícula for ímpar, você deve entregar o código do programa cliente; se seu número de matrícula for par, você deverá entregar o código do programa servidor.

Seu programa será testado semi-automaticamente com as implementações de referência do cliente e servidor implementados pelo professor. Por isso, envie *apenas* o código fonte (arquivo .c) do seu programa. Para testar o correto funcionamento do seu cliente ou servidor, teste seu programa com o programa complementar de outros colegas.