Documentação - TP1 - Redes

Gabriel Cerqueira e Silva - 2017105567

Introdução

O trabalho prático 1 teve como proposta a implementação de um sistema de transferência de arquivo simples entre um servidor e um cliente utilizando sockets em C.

Desafios

Ao realizar a implementação do trabalho , alguns desafios vieram a tona , sendo eles:

O primeiro desafio envolveu a compreensão da estrutura de sockets em C , na qual requer um entendimento de estruturas de dados, funções e chamadas do sistema relacionadas aos sockets. Foi necessário compreender os conceitos como criação de socket, associação de endereços, escuta por conexões, estabelecimento e encerramento de conexões, envio e recebimento de dados, entre outros.

Em seguida temos um outro desafio que foi o estabelecimento e desconexão de conexões, que envolveu a configuração correta dos parâmetros do socket, como família de protocolos (IPv4 ou IPv6), o socket e a porta. Além disso, foi necessário lidar com possíveis erros e tratar exceções durante o processo de conexão.

Além disso , também foi um desafio manter servidor e cliente ativos: No sistema de transferência de arquivos, foi necessário que tanto o servidor quanto o cliente estejam ativos e prontos para receber e enviar dados. Para Isso foi necessário o desenvolvimento de loops e mecanismos de espera que permitiram que o servidor fique aguardando por conexões de clientes e que o cliente estivesse pronto para enviar e receber dados do servidor. Por fim tivemos o desenvolvimento da lógica do sistema de transferência de arquivos em sí, em que envolveu a leitura de arquivos, formatação dos dados, operações de envio de arquivos pelo cliente e recebimento no lado receptor.

Dificuldades

As principais dificuldades encontradas na hora da implementação do sistema, foram a manipulação de arquivos , a compreensão da programação em sockets , e por fim a maior dificuldade encontrada foi a manipulação de strings em C , para realizar a formatação e processamento das mensagens, em ambos os lados na qual a comunicação foi estabelecida.

Solução

Infraestrutura

Com relação a infraestrutura foi definido a função “usage” , que verifica os parâmetros de entrada obedecendo o padrão :

Ipv4:

No terminal 1: ./server v4 51511
No termnal 2: ./client 127.0.0.1 51511

Ipv6:
No terminal 1: ./server v6 51511
No terminal 2: ./client ::1 51511

em seguida é mostrado exemplos de conexão.
Na parte do cliente é realizado a verificação da corretude dos parâmetros de entrada.
A estrutura “storage” armazena as informações do endereço de rede , e a função “addrparse” preenche a estrutura com base nos parâmetros de entrada.Em seguida a função “socket” cria um novo socket , sendo do tipo SOCK_STREAM,que utiliza o protocolo TCP, e o mesmo é associado ao domínio do endereço especificado em “storage.ss_family” , sendo IPv4 ou IPv6.Após isso a função “connect” estabelece a conexão com o endereço de rede recuperado.Por fim a função “run” é chamada com o objetivo de estabelecer a lógica do cliente com o servidor.

Já na parte que compreende o servidor, o armazenamento das informações de rede e a criação do socket é parecida com a lógica do cliente. A função bind associa o socket “serversock” ao endereço de rede recuperado. Em seguida a função listen é chamada para iniciar a escuta de conexões, e o loop infinito é definido para receber conexões de clientes.A função accept é chamada para aceitar uma nova conexão e em seguida é armazenado as informações do cliente conectado (client sock,client address).Por fim a função “exec” é executada , contendo a lógica da comunicação com o cliente.Quando o loop supracitado é encerrado a conexão é fechada usando a função “close” e a conexão entre ambos chega ao fim.

Para modularizar funções de suporte da infraestrutura , foi usado os arquivos (common.h e common.c) , e para modularizar funções de suporte da lógica do sistema de envio de arquivos , foi usado os arquivos (handlers.c e handlers.h), e para lidar com o envio de respostas para o cliente e o recebimento de respostas por parte do mesmo , foi utilizado respectivamente as funções “sendResponse” e “receiveResponse” que executam as funções send e recv.

Fluxo de execução

Basicamente o sistema funciona da seguinte maneira:

1 - É estabelecida a comunicação entre cliente e servidor , por meio dos comandos :

Ipv4:

No terminal 1: ./server v4 51511
No termnal 2: ./client 127.0.0.1 51511

Ipv6:
No terminal 1: ./server v6 51511
No terminal 2: ./client ::1 51511

2- O cliente seleciona um arquivo por meio do comando “select file [nomeDoArquivo]” , e então é realizado as validações se existe o arquivo com o nome citado e se o arquivo possui o formato válido (.c,.txt,.cpp,.py,.java,.tex) , se passar das validações o arquivo é selecionado.

3 - Com o arquivo selecionado o cliente pode enviar o arquivo para o servidor por meio do comando “send file”, nessa etapa é verificado se há arquivo selecionado , se sim , o arquivo é enviado e o servidor responde validando o recebimento.

4 - Em seguida o servidor trata a mensagem recebida , recuperando o nome do arquivo e o seu conteúdo, e realiza a verificação se aquele arquivo se encontra no diretório de arquivos do servidor “serverFiles”, se sim , é realizado a sobrescrita, caso contrário o arquivo é salvo normalmente.

5 - O usuário pode interromper a conexão com o servidor por meio do comando exit finalizando a conexão.