Trabalho - Redes 1

Nicolas Dencker De Marco - GRR20171605

Março de 2021

1. Introdução

O trabalho consiste em construir uma comunicação entre um “Cliente”, que recebe todos os comandos, e um “Servidor”, que recebe boa parte dos comandos - excluindo “lcd” e “lls” - e os executa, retornando para o cliente os dados respectivos.

2. Arquivos

Os arquivos consistem em:

Um arquivo cliente.c: onde fica a leitura e tratativa dos comandos, chamada das função de envio e recebimento de mensagens, retorno dos comandos e erros;

Um arquivo servidor.c: onde fica chamada das função de envio e recebimento de mensagens, execução dos comandos recebidos e envio de erros ou retorno dos comandos executados;

Um arquivo comandos.c e comandos.h: onde fica basicamente toda a execução lógica dos comandos cd, ls, lcd, lls, ver, linha, linhas e edit.

Um arquivo functions.c e functions.h: onde fica as funções de envio, montagem e recebimento de mensagens, além das função de cálculo e checagem de paridade, função de execução do timeout, e struct do kermit_type que corresponde ao buffer enviado em cada mensagem.

O fluxo de execução de envio e recebimento do para e espera varia um pouco de comando para comando, porém eles são essencialmente iguais.

3. Fluxo das Mensagens

Primeiro o cliente envia uma mensagem com o comando, calculando sua paridade na montagem do buffer, então ele espera receber uma mensagem, assim que recebe, checa a paridade da mensagem para garantir sua integridade.

Caso demore muito para receber mensagem é retornado um timeout e encerra a execução do programa.

Caso as paridades estejam diferentes do momento do envio e do momento de recebimento da mensagem, é enviado um NACK com o intuito de receber a mensagem novamente.

Caso esteja tudo certo, o programa verifica se o tipo da mensagem é um NACK, enviando a mesma mensagem enviada antes novamente.

E basicamente fica assim até enviar o fim da transmissão ou só a mensagem por completo, e receber um ACK garantido que tudo foi recebido corretamente.

Tendo o comando recebido por completo no servidor, o servidor executa o comando desejado, usando os parâmetros passados, e envia um erro caso haja, ou envia o resultado, até receber um ACK da parte do cliente ao fim da transmissão, garantindo que o cliente tenha recebido corretamente os dados.

Após isso o cliente printa o erro ou os dados recebidos.

4. Dificuldades enfrentadas

Bom durante todo o trabalho existiram alguns pontos que sofri mais ou não fui capaz de executar de maneira ideal na minha visão, um deles foi o cálculo e checagem de paridade, não entendi muito bem como fazê-la e quando fazê-la, mas acredito que fiz de uma forma próxima da realidade.

Outro ponto que tive um pouco de dificuldade foi no tratamento de envio e recebimento das mensagens, tendo alguns problemas com duplicação de mensagem, por exemplo, ou na ordem de que tipo enviar e receber e que comando executar depois, apesar dos desenhos do protocolo serem bastante úteis, ainda sofri na implementação, não conseguindo reaproveitar muito código.

No Edit, o tratamento de erros também acabei decidindo primeiro receber todas as informações necessárias, como o nome do arquivo e número da linha para só no final enviar o código de erro caso houvesse, diferente do protocolo demonstrado pelo professor.