Relatório trabalho Redes 1

Esse trabalho teve como objetivo implementar um editor de texto remoto. Foi implementado um sistema de Cliente - Servidor, ambos conversando via Raw Socket.

Descrição do trabalho

O "Cliente" implementa as funcionalidades de listar diretório e alterar diretório
O "Servidor" implementa as funcionalidades de listar diretório, alterar diretório, mostrar o conteúdo de um arquivo especifíco, mostrar a linha de um arquivo específico, mostrar um intervalo de linhas de um arquivo especifíco e editar um arquivo.

O trabalho possui o seguinte fluxo:

- O "Cliente" recebe os comandos via terminal
- O "Cliente", caso o comando seja um comando de cliente
    - Processa o comando
    - Mostra na tela o resultado
- Caso seja um comando de "Servidor"
    - Envia para o programa "Servidor" o comando recebido
    - O "Servidor" processa o comando
    - Devolve a resposta para o "Cliente"
    - O "Cliente" mostra na tela caso tenha alguma mensagem

Detalhes de implementação

Nesse trabalho, a comunicação entre Cliente - Servidor, é utilizado o protocolo Kermit Simplificado. O protocolo possui os campos de marcador de início, endereço de destino, endereço de origem, tamanho da mensagem, número da mensagem (sequência), o tipo da mensagem, os dados (de máximo de 15 em 15 bytes o envio da mensagem) e o campo de paridade.
Foi utilizado "para e espera" para envio de mensagens.
O Timeout mata o programa se o cliente ou o servidor ficarem no minimo 5 segundos sem receber alguma resposta.
Nesse trabalho não consegui implementar 100% o "para e espera" para o envio de mensagens, ao invés de enviar um ACK após receber cada mensagem, é enviado somente a cada comando. Eu notei que deveria enviar um ACK a cada mensagem um pouco antes da data final de entrega do trabalho, eu teria que mudar grande parte do envio e recebimento de mensagens para suportar essa mudança, assim eu não consegui fazer essa alteração.
Apesar desse incidente, consegui implementar todos os comandos, verificar a paridade do protocolo, implementar o timeout, tratar os códigos de erros e o reenvio da mensagem caso NACK ocorra.

O trabalho possui um make para compilar. Basta compilar e rodar

```
make
```

```
sudo ./cliente
```

```
sudo ./servidor
```

No cliente utilizar como por exemplo:

```
/home/lucas/Materias/redes_2/redes (cliente): lls
```

```
/home/lucas/Materias/redes_2/redes (cliente): lcd ..
```

```
/home/lucas/Materias/redes_2/redes (cliente): ls
```

```
/home/lucas/Materias/redes_2/redes (cliente): ver teste.txt
```

```
/home/lucas/Materias/redes_2/redes (cliente): linha 1 teste.txt
```

```
/home/lucas/Materias/redes_2/redes (cliente): linhas 1 5 teste.txt
```

```
/home/lucas/Materias/redes_2/redes (cliente): edit 5 teste.txt texto
bonitinho aqui...
```

Problemas ao decorrer

Tive algumas dificuldades durante o desenvolvimento do trabalho, entre eles:

- O envio e recebimento de mensagens estavam sempre duplicados
  Eu corrigi esse problema lendo sempre duas mensagens e descartando uma delas
- Cliente estava lendo as mensagens que ele mesmo enviada.
  Para a correção desse, notei que eu não verificava os campos de endereço de destino e origem, após eu começar a fazer a verificação desses campos eu não tive mais esse tipo de problema.
- Servidor lendo mensagens de servidor
  Mesmo caso do anterior.

- Problema de como enviar um ack/nack para cada mensagem enviada.
  Esse problema eu não consegui resolver, consegui enviar somente um ACK para cada comando, acredito que esse problema esteja ligado com o problema seguinte, em não conseguir estruturar direito como ficaria a comunicação entre os 2.
- Problema em como estruturar a mensagem entre cliente e servidor
  Esse foi o meu maior problema, após um grande tempo sem programar em C tive dificuldades em como estrutura o projeto e a comunicação entre os 2 processos.