Especificação Trabalho Redes 1

Vocês devem implementar um editor de textos remoto simplificado. Vocês devem implementar um programa como servidor, o qual não recebe comandos pelo teclado ou tem qualquer iteração com o usuário. Vocês também devem implementar um programa cliente, que vai cuidar das iterações com o usuário. O cliente e o servidor devem se comunicar através da RAWSockets, utilizando a função exemplo que esta no site www.inf.ufpr.br/albini. Observação, RAWSocket só pode ser utilizada pelo ROOT da máquina, então tomem cuidado. O servidor e o cliente devem rodar na mesma máquina e devem se conectar utilizando o endereço de rede 127.0.0.1 ou a interface de rede de Loop Back (lo), caso especifiquem o número de porta, podem usar qualquer número entre 1024 e 65000.

Os programas devem ser implementados em C ou C++.

O trabalho é individual. Cópias terão nota igual a 0.

O trabalho deverá ser apresentado através de videoconferência no final de setembro, em datas e horários agendados com o professor.

Os seguintes comandos são obrigatórios:

1. **cd** *<nome_dir>* – efetua a troca de diretório no servidor
2. **lcd** *<nome-dir>* – efetua a troca de diretório no cliente
3. **ls** – lista os arquivos do diretório corrente do servidor
4. **lls** – lista os arquivos do diretório corrente do cliente
5. **ver** *<nome_arq>* - mostra o conteúdo do arquivo texto do servidor na tela do cliente. As linhas do arquivo devem ser numeradas para mostrar na tela.
6. **linha** *<numero_linha> <nome_arq>* - mostra a linha *<numero_linha>* do arquivo *<nome_arq>* que esta no servidor na tela do cliente.
7. **linhas** *<numero_linha_inicial> <numero_linha_final> <nome_arq>* - mostra as linhas entre a *<numero_linha_inicial>* e *<numero_linha_final>* do arquivo <nome_arq> que esta no servidor na tela do cliente.
8. **edit** *<numero_linha> <nome_arq> "<NOVO_TEXTO>"* – troca a linha *<numero_linha>* do arquivo *<nome_arq>* que esta no servidor pelo texto <NOVO_TEXTO> que deve ser digitado entre aspas. As aspas são delimitadores do texto e não devem ser escritas no arquivo.

O trabalho deve ser implementado seguindo o protocolo especificado em anexo, baseado no Kermit simplificado.

Protocolo:

- Mensagens

Marcador Início	Tamanho	Sequência	Tipo	Dados	Paridade

- Enquadramento: Marcador de início – 01111110
  Campo Tamanho – 4 bits – indica o tamanho da área de dados em bytes.
- Dados: pode viajar vazia (0 bytes) até o limite que pode ser representado pelo campo tamanho (15 bytes). Se a mensagem tiver mais de 15 bytes devem ser enviadas mensagens subsequentes até que toda a mensagem seja enviada.
- Sequencialização: 8 bits
- Detecção de erros: 8 bits - Paridade Vertical de 8 bits – Deve ser calculada sobre os campos: Tamanho / Sequência / Tipo / Dados. Pode ser feita com um XOR bit a bit de todos estes campos.
- Controle de Fluxo: Para-e-Espera
- Timeout é obrigatório. Existem diversas formas de implementar o Timeout, vocês podem escolher qual quiserem.
- Tipo:
  - 0000 – cd – nome do diretório viaja na área de dados
  - 0001 – ls
  - 0010 – ver – nome arquivo viaja na área de dados
  - 0011 – linha – nome arquivo viaja na área de dados
  - 0100 – linhas – nome arquivo viaja na área de dados
  - 0101 – edit – nome arquivo viaja na área de dados
  - 0110
  - 0111
  - 1000 – ACK
  - 1001 – NACK
  - 1010 – linha inicial e final – área da dados carrega um ou dois inteiros
  - 1011 – Conteúdo ls
  - 1100 – Conteúdo Arquivo
  - 1101 – Fim transmissão – área da dados sempre vazia
  - 1110
  - 1111 – Erro – campo de dados contém código do erro
- Códigos de erro
  - Números inteiros
  - 1 – acesso proibido / sem permissão
  - 2 – diretório inexistente
  - 3 – arquivo inexistente
  - 4 – linha inexistente