segunda-feira, 22 de março de 2021

Relatório - Editor de Texto Remoto

1. funcoes.c = funções compartilhadas
2. server.c = funções somente do servidor
3. client.c = funções somente do cliente
4. basic.c = funções auxiliares
5. main.c e funcoes.h = funções essenciais
6. ConexaoRawSocket.c

---

cd <nome_dir> – efetua a troca de diretório no servidor

- Cliente:

A partir do terminal se recebe o nome do diretório, chama-se as funções de montar pacote, enviado para o servidor e espera-se por uma resposta.

- Servidor:

Ao receber a mensagem é passado pela função de desempacotar pacote e chamado a função cd do server. Se o CHDIR der certo é enviado a resposta de ACK, se não é enviado o NACK.

BUGS: Nenhum bug conhecido.

---

lcd <nome_dir> – efetua a troca de diretório no cliente

- Cliente:

A partir do terminal se recebe o nome do diretório e chama a função de cd do cliente. (CHDIR)

BUGS: Nenhum bug conhecido.

---

ls – lista os arquivos do diretório corrente do servidor

- Cliente:

Na função de ls do cliente, chama-se as funções de montar pacote, enviado para o servidor e espera-se pela lista de dados. Quando chegar um pacote de tipo 13 (1101) é finalizado a função de ls.

- Servidor:

Ao receber a mensagem é passado pela função de desempacotar pacote e chamado a função de ls do server. Existe um while que percorre cada arquivo/pasta do diretório e é feita uma verificação: se o nome do arquivo/pasta for maior de 15 bytes é feito uma divisão do nome n vezes (n = tamanho do nome / 15)e enviado para o cliente a cada divisão; se o nome do arquivo/pasta for menor é enviado para o cliente.

A cada mensagem enviada é esperado o ACK/NACK do cliente. Quando os nomes acabarem, é enviado uma mensagem com o tipo 13 (1101).

BUGS: Se o ls não funcionar, digite novamente. Poucas vezes ocorre um segmentation fault no servidor, mas se resolve finalizando o programa nos dois terminais e abrindo de novo.

---

lls – lista os arquivos do diretório corrente do cliente

- Cliente:

Na função de ls tem um system("ls") e aparece no terminal a resposta.

BUGS: Nenhum bug conhecido.

---

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.

- Cliente:

A partir do terminal se recebe o nome do arquivo, chama-se as funções de montar pacote, enviado para o servidor e espera-se pelas linhas do arquivo. Tentei fazer que o servidor já mandasse as linhas com a numeração, mas não funcionou. Então ao receber o dado do arquivo pela mensagem é percorrido a string a procura de um /n, se encontra é impresso na tela o número da linha atual na tela e incremento a variável contador. Isso acontece até receber uma mensagem com o tipo 13 (1101) indicando que o arquivo acabou.

- Servidor:

Ao receber a mensagem é passado pela função de desempacotar pacote e chamado a função ver do server. BUGS: Nenhum bug conhecido. Usando a função de fread é lido 15 bytes por vez e enviado para o cliente. Isso acontece até o EOF.

A cada mensagem enviada é esperado o ACK/NACK do cliente. Quando os nomes acabarem, é enviado uma mensagem com o tipo 13 (1101).

BUGS: Poucas vezes ocorre um segmentation fault no servidor, mas se resolve finalizando o programa nos dois terminais e abrindo de novo.

---

linha <numero_linha> <nome_arq> - mostra a linha <numero_linha> do arquivo <nome_arq> que esta no servidor na tela do cliente.

- Cliente:

A partir do terminal se recebe o número da linha e o nome do arquivo, chama-se as funções de montar pacote para às duas informações, enviado para o servidor e espera-se um ACK/NACK para cada uma. Dentro de um while é feito a espera de toda a linha. Isso acontece até receber uma mensagem com o tipo 13 (1101) indicando que a linha acabou.

- Servidor:

Ao receber a mensagem com o número da linha é enviado um ACK/NACK e ao receber a mensagem com o nome do arquivo é verificado se o arquivo existe, se não é enviado um NACK com o número do erro no campo de dados.

Dentro de um while é verificado a cada caracter se é um \n, se sim é somado ao contador. Quando o contador for igual à linha enviada pelo cliente existem três possibilidades:

1. A linha for somente um \n e é enviada uma mensagem para o cliente com o tipo igual a 6 (110) indicando que existe nada a mostrar.
2. A linha é maior que 15 bytes e ocorre o mesmo processo de divisão do ls.
3. A linha é menor ou igual a 15 bytes e é enviada direto para o cliente.

A cada mensagem enviada é esperado o ACK/NACK do cliente. Quando as linhas acabam, é enviado uma mensagem com o tipo 13 (1101).

CONTROLE DE ERROS: Se for enviado uma linha inexistente, é enviado um NACK com o número 4 no campo de dados.

BUGS: Nenhum bug conhecido.

---

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.

- Cliente:

A partir do terminal se recebe os números das linhas e o nome do arquivo, chama-se as funções de montar pacote para às duas informações, enviado para o servidor e espera-se um ACK/NACK para cada uma. Dentro de um while é feito a

espera de todas as linhas. Isso acontece até receber uma mensagem com o tipo 13 (1101) indicando que a linha acabou.

- Servidor:

Ao receber a mensagem com os números das linhas é enviado um ACK/NACK e ao receber a mensagem com o nome do arquivo é verificado se o arquivo existe, se não é enviado um NACK com o número do erro no campo de dados.

Dentro de um for é feito a contagem da linha inicial até a final, em cada vez chamada uma função parecida com da linha: verifica os carácteres a procura de \n quando encontrar linha do cliente é enviado a ele toda a linha.

É sempre feito o ponteiro do arquivo ficar no início em cada interação.

A cada mensagem enviada é esperado o ACK/NACK do cliente. Quando as linhas acabam, é enviado uma mensagem com o tipo 13 (1101).

CONTROLE DE ERROS: Se for enviado uma linha inicial maior que a final, é enviado um NACK com o número 5 no campo de dados indicando o erro.

BUGS: Poucas vezes ocorre um segmentation fault no servidor, mas se resolve finalizando o programa nos dois terminais e abrindo de novo.

---

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.

- Cliente:

A partir do terminal se recebe o número da linha, o nome do arquivo e o texto, chama-se as funções de montar pacote para as informações, enviado para o servidor e espera-se um ACK/NACK para cada uma.

Se o texto for maior que 15 bytes, é feita uma divisão para que cada mensagem para o servidor vá com 15 bytes no máximo. Se for menor é enviado direto para o cliente. Se espera uma resposta depois de cada mensagem.

- Servidor:

Ao receber a mensagem com o número da linha é enviado um ACK/NACK e ao receber a mensagem com o nome do arquivo é verificado se o arquivo existe, se não é enviado um NACK com o número do erro no campo de dados.

É criado um novo arquivo para modificar a linha. Se cópia todas as linhas até a escolhido pelo cliente. Quando chegar, se recebe do cliente os dados até chegar uma mensagem do tipo 13 (1101). Depois de terminar a escrita do dado do cliente, finaliza a cópia para o arquivo novo.

É renomeado o arquivo novo para o nome do arquivo original e depois é removido o arquivo original.

A cada mensagem enviada é esperado o ACK/NACK do cliente. Quando as linhas acabam, é enviado uma mensagem com o tipo 13 (1101).

CONTROLE DE ERROS: Se for enviado uma linha inexistente, é enviado um NACK com o número 5 no campo de dados.

BUGS: Nenhum bug conhecido.

PROBLEMAS: Não implementei a detecção de erro pela verificação da paridade.