# Rede de Computadores

Daniel Augusto Costa de Sá - 2019041515
daniel.augusto191@gmail.com
UFMG - 2023

## Introdução

O trabalho visa implementar uma aplicação de tranferencia de arquivos por socket tanto para IPV4 quanto para IPV6, entre um servidor e um cliente ultilizando a linguagem C.

O Cliente tem as funções de selecionar um arquivo, enviar um arquivo e encerrar a conexão. Cada função deve comunicar com o servidor e imprimir o resultado.

## Implementação e execução

Para implementar tanto servidor quando cliente, fiz uso das bibliotecas padrao de C, (stdio, stdlib), a biblioteca string.h para manipulação dos arquivos a serem enviados, e unistd, types.h, socket.h, inet.h, para a implementação da comunicação cliente-servidor.

No mais, a implementação segue um script simples, lê do teclado no servidor o comando a ser executado, faz uma separação entre as partes do comando e guarda em um vetor ("AstrSplit") onde separo cada execução.

Em caso de select, uso a biblioteca string.h para separa o nome do arquivo da extenção, e verifico se é valida entre as opções do trabalho, inclusive tratei o caso com pontos no nome do arquivo (i.e: exemplo.a.cpp, é um cpp). Se é valida, imprimo na tela a confirmação de seleção de arquivo, se não imprimo não é valido. E caso o arquivo não exista antes de verificar a extenção imprimo a falta do arquivo. (Imagino que está seja a ordem correta de impressão pela especificação do trabalho).

Em caso de send, verifico de o ponteiro pro arquivo é nulo (arquivo não foi selecionado) se for imprimo o erro, se não crio um cabeçalho com o nome do arquivo e uso um caracter '|' (pipe, ascii: 124), para separar o cabeçalho da mensagem. Pelo que entendi da documentação apenas de dizer sem símbolos, foi dito no forum que caracteres ascii eram valido, apenas não podia UTF-8, acentos e etc..., logo usei esse ASCII como delimitador. O motivo disso, que inclusive vi no forum de duvidas é que na estrutura {cabeçalho}{conteudo} tem alguns corner cases, que podem gerar erro na hora de separar a mensagem no servidor, por exemplo o arquivo .c que começo com {pp ....} pode gerar ambiguidade se é arquivo .c que começa com "pp", ou um arquivo "cpp", portanto usei "|" para separação. Depois de enviado o servidor recebe, verifica, e retorna a mensagem de verificação para ser impressa no cliente.

Em caso exit, envio um "exit\end" para o servidor e trato lá dizendo que é para fechar a coneção, o servidor confirma, manda a mensagem "connection closed" como especificado, e encerra a conexão e termina, o cliente imprime e termina também.

Além disso, por garantia usei um buffer size do dobro da especificação(1024) para garantir, que mensagem cabeçalho etc... funcionem.

## Notas e observações

A implementação da parte de conexão foi bem tranquila, a playlist e o livro disponibilizados ajudaram muito, foi preciso apenas ajustar para receber multiplos comandos, o que foi bem rápido de fazer, pelo incrivel que parece a parte do tratamento de string foi um pouco mais complicado, erros de overflow por caso do caracter \0 foram um pouco chatos de verificar e corrigir, mas é só questão de experiencia mesmo, conforma o trabalho foi se desenvolvendo fui entendendo melhor como ocorriam. No geral, ultilizei só uma sobrepocição indexada, entao onde erá \0, mudei para '\' e deixei o tamanho da string com uma folga, para setar mais 3 caracteres "end".

A Especificação do trabalho também me deixou com algumas duvidas, estava bem explicado mas algumas coisas me pareciam ambiguas, ou não seguiam o que eu imagino que um servidor real faria, então fui implementando como interpretei e deixo aqui a especificação de tudo no geral:

- Como mencionado mais cedo, usei o "|" para separar cabeçalho de mensagem, apenas de estar especificado sem simbolos, houve discursão no forum, além do problema de extensões que apontei antes, e cheguei a conclusão que adicionar um delimitador não seria problema.

- Imprimir mensagem são apenas no cliente, toda a especificação disse sobre respostas do servidor, e como as mensagem são "arquivo recebido" "arquivo sobreescrito" etc... imagino que sejá a implementação do que o servidor responde ao cliente, e o cliente vê como seguiu a execução. Logo não faria sentido printar algo no servidor. Espero que esteja sobre isso, aparentemente outros colegas que pareciam ter chegado a outra conclusão logo não tenho certeza.

- Com forme foi discutido no forum, quando há um comando errado, o cliente é desconectado e o servidor continua rodando esperando outra conexão, acredito que é o que faça sentido, e aparentemente outros interpretaram assim também.

**Referencias**


- linux.die.net/man
- man linux
- Cap. 2,3 do livro programação de sockets, sugerido na especificação.
- Playlist sugerida na especificação