Redes de Computadores
Trabalho Prático 1

Luís Felipe Ramos Ferreira

lframos_ferreira@outlook.com

1 Introdução

O Trabalho Prático 1 da disciplina de Redes de Computadores teve como proposta o desenvolvimento de um *Campo Minado* que permite a interação entre um cliente e um servidor usando sockets em linguagem C.

O repositório onde está armazenado o código utilizado durante o desenvolvimento desse projeto pode ser encontrado neste endereço.

2 Implementação

Conforme especificado no enunciado, o projeto foi todo desenvolvido na linguagem de programação C em um ambiente *Linux*, e o manuseio de sockets por meio da interface POSIX disponibilizada para a linguagem. Para manter uma maior organização do código, além dos arquivos *server.c* e *client.c*, os quais possuem respectivamente as implementações do servidor e do cliente, foi criado um arquivo auxiliar *common.c* e seu arquivo de cabeçalho *common.h*, os quais possuem as especificações e implementações de funções auxiliares que podem ser utilizadas por ambas as partes do projeto.

3 Desafios, dificuldades e imprevistos

A primeira dificuldade imposta pelo trabalho prático foi a familiarização com a interface POSIX de programação em sockets. Embora a linguagem C seja considerada de alto nível, em muitos momentos suas funcionalidades podem ser confusas, o que tornou difícil um primeiro contato com a criação dos sockets e da conexão entre eles. No entanto, esse empecilho não durou muito, uma vez que existe uma infinidade de conteúdos sobre o assunto disponibilizados na *internet*, além, é claro, das páginas de manual disponibilizadas nas distribuições *Linux*. Outra fonte extremamente útil para lidar com dificuldades do tipo foi a *playlist* do professr Ítalo Cunha, disponibilizada na especificação do trabalho.

Outro desafio encontrado durante o desenvolvimento foi a de manter a flexibilidade da escolha entre *IPv4* e *IPv6* na comunicação entre o cliente e o servidor. Em particular, compreender como identificar, por parte do cliente, qual tipo de endereço estava sendo utilizado, tomou bastante tempo. No fim, graças mais uma vez à *playlist* do professor Ítalo o problema foi resolvido. A função *addrparse()* desenvolvida pelo professor, que utiliza a função *inet_pton()* para converter da *string* que representa o endereço para o formato binário correto facilitou a resolução desse problema.

A lógica de conexão e saída de diferentes clientes com o servidor também foi um desafio no desenvolvimento do projeto. A princípio, na primeira implementação, assumi que após um cliente enviar o comando *exit* o servidor deveria ser desligado também. No entanto, essa lógica esta incorreta, uma vez que o comportamento esperado é de que, após um cliente se desconectar, o servidor se mantenha rodando para que novos clientes possam se conectar e jogar uma partida de campo minado. Para resolver esse problema, dois laços foram utilizados por parte do servidor. No primeiro laço, utiliza-se a função *accept()* para aceitar a conexão com algum cliente que esteja tentando se conectar com o servidor. Assim que uma conexão for feita, um novo laço é feito, pelo qual a comunicação entre cliente e servidor é feita. Agora, quando o cliente atual envia uma mensagem de *exit*, a conexão entre o servidor e este cliente é fechada, mas o servidor continua funcionando, e executa a ação bloqueante do *accept()* enquanto aguarda por um novo cliente para se conectar. Desse modo, o servidor só irá

parar de funcionar quando acontecer um erro durante a execução ou quando o usuário com controle de acesso sobre o servidor forçadamente parar sua execução.

Por fim, um dos outros grandes desafios encontrados durante o desenvolvimento do trabalho foi o do envio da estrutura de *action* nas mensagens trocadas entre o cliente e o servidor. Em um primeiro momento, a estrutura foi passado diretamente nas mensagens, utilizando-se a função *send()* com o o *buffer* de mensagem sendo a própria estrutura de *action*. Após discussões no fórum da disciplina, foi levantada a questão de se utilizar serialização para envio da estrutura. Fiz uma implementação inicial dessa serialização com base em recomendações de conteúdos disponibilizados na *internet*. Apesar de ter funcionado, pelas respostas obtidas pela monitora no fórum, ficou-se entendido que também seria correto não utilizar serialização, o que tornaria o código mais simples e compatível com o dos outros colegas que não utilizaram da mesma forma de implementação da serialização.

4 Conclusão

Em suma o projeto permitiu grandes aprendizados tanto na parte teórica como na parte prática do desenvolvimento de sistemas de redes. As implementações tornaram possível compreender melhor como funciona o protocolo de comunicação TCP, como deve ser feita e mantida a comunicação entre um servidor e um cliente, etc. Na parte prática, foi permitido obter uma maior familiaridade com a interface POSIX de programação em Sockets, assim como em programação na linguagem C de maneira geral.

5 Referências

- Livros:
  - Tanenbaum, A. S. & Wetherall, D. (2011), Computer Networks, Prentice Hall, Boston.
  - TCP/IP Sockets in C: Practical Guide for Programmers, Second Edition
- Web:
  - https://www.tutorialkart.com/c-programming/c-read-text-file/#gsc.tab=0
  - https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html
  - https://riptutorial.com/c/example/30858/using-gnu-getopt-tools
  - https://www.ibm.com/docs/en/zos/2.3.0?topic=sockets-using-sendto-recvfrom-calls
  - https://www.educative.io/answers/how-to-implement-tcp-sockets-in-c
  - https://www.geeksforgeeks.org/regular-expressions-in-c/
- Youtube:
  - Jacob Sorber
  - Think and Learn sockets playlist
  - Playlist do professor Ítalo Cunha