Caio Teles Cunha 2020006434

1. Introdução

A proposta do Trabalho Prático 1, da disciplina de Redes, foi implementar um servidor e um cliente, utilizando sockets, que pudessem se comunicar e executar um jogo simples de campo minado.

O servidor ficou com a responsabilidade de armazenar o estado atual do jogo, lidar com os comandos enviados pelo cliente e sinalizar o final do jogo. Já o cliente foi responsável pelo tratamento de erros de input e pela impressão no terminal do grid do jogo de forma mais amigável ao usuário, utilizando char ao invés de int como ficou armazenado no servidor.

2. Implementação

O programa foi desenvolvido em um ambiente operacional linux, na linguagem C.

3. Estrutura de dados

Apenas uma estrutura de dados bem simples foi usada a struct action. Essa estrutura contém apenas três atributos: um inteiro ”type”, que armazena a ação enviada pelo cliente ou pelo servidor, um vetor de inteiros de duas posições “coordinates” para armazenar as coordenadas do grid em que alguma ação será executada e uma matriz de inteiros 4x4 para armazenar o tabuleiro do jogo. Essa struct foi definida pela documentação de forma a manter os servidores e clientes compatíveis entre os alunos, já que a comunicação entre o servidor e o cliente é feita através do envio dessa struct.

4. Discussão dos desafios

O primeiro desafio encontrado foi tornar o código compatível com os protocolos ipv6 e ipv4. O protocolo ipv4 tem um endereço de 32 bits, já o protocolo ipv6 possui um endereço de 128 bits. A biblioteca usada possui estruturas diferentes para lidar com cada endereço, então foi necessário criar uma função que, sabendo qual o protocolo, inicializa as estruturas necessárias. É necessário utilizar o struct sockaddr como uma interface para as as structs sockaddr_in (ipv4) e sockaddr_in6 (ipv6). O struct sockaddr_storage também foi usado devido à sua funcionalidade de armazenar qualquer um dos endereços e depois escolhemos o tipo do endereço específico. As manipulações dessas structs para que o programa funcionassem de maneira correta foram complexas de entender e aplicar. Outra coisa que foi desenvolvida foi o entendimento de host byte order e network byte order. A porta, por exemplo, deve estar sempre em network byte order (big-endian) ao passar para a struct sockaddr_storage.

Outro desafio encontrado foi entender a lógica por trás da criação do servidor. Primeiro é necessário inicializar o socket, depois passamos para a fase de abertura passiva. Nela primeiramente usamos a função bind que fala para o sistema operacional onde o servidor vai esperar uma conexão passando um endereço de rede e uma porta. A função listen é usada para manter o servidor aguardando uma conexão. Após isso vem a função accept que aceita uma conexão de um dispositivo remoto, então ela pode demorar um período arbitrário para retornar. Entender essas

partes foi um processo demorado, mas ao desenvolver o programa ficou claro sua utilização bem como o funcionamento geral.

Depois de entender a lógica dos sockets, a implementação do jogo em si foi mais simples. Entender a lógica de manter o servidor conectado e do loop para o recebimento das jogadas foi a parte mais complicada.

4. Conclusão

Em resumo, o trabalho foi uma ótima forma de praticar e solidificar os conhecimentos recebidos em aula sobre a matéria de redes. Também foi de suma importância para o aprendizado da programação em sockets e como eles funcionam. Acredito que o trabalho foi essencial para me ajudar a compreender a matéria e me ajudar no estudo da mesma.