Universidade Federal
de São João del-Rei

Implementação de um Jogo Multiusuário por turnos

Redes

Professor: Rafael Sachetto Oliveira

Matheus Tavares

Vítor Augusto Niess Soares Fonseca

1. Introdução

Este trabalho tem como objetivo melhorar o entendimento em programação de protocolos na camada de aplicação e transporte em ambientes Unix/Linux. O foco principal é a implementação de um jogo multiusuário baseado em turnos, que utiliza o modelo cliente-servidor, onde a comunicação entre o servidor e dois clientes ocorre de forma precisa e eficiente. Através deste projeto, busca-se o desenvolvimento de um protocolo de comunicação robusto e funcional.

O jogo em questão é uma versão digital, desenvolvida na linguagem de programação C, do popular jogo "Pedra, Papel e Tesoura", no qual dois jogadores competem para alcançar a vitória, acumulando pontos, processo que será explicado com mais detalhes nas seções seguintes.

Além do aspecto lúdico do jogo, o trabalho se concentra em estabelecer uma comunicação eficaz entre o servidor e os clientes. Essa comunicação é essencial para garantir que as ações dos jogadores sejam refletidas com precisão no jogo e que os resultados sejam transmitidos de maneira adequada.

2. Descrição do Jogo

O jogo implementado neste trabalho é um jogo de “Pedra, Papel e Tesoura”, no qual cada jogador, durante seu turno, escolhe uma de três opções, pedra, papel ou tesoura, para que, então, o servidor determine o vencedor de acordo com as regras do jogo. O jogo é baseado em rodadas, onde os jogadores fazem suas jogadas simultaneamente até que um dos jogadores atinja cinco pontos, tornando-se o vencedor.

3. Metodologia

O programa desenvolvido foi dividido em três arquivos principais, sendo “server.c”, responsável pela implementação do servidor, “client.c” e “client1.c”, ambos sendo responsáveis por definir os clientes que se conectarão ao servidor.

3.1. Explicação Detalhada

3.1.1. Servidor

No arquivo do servidor, o “server.c”, a inicialização se da pela função “main()”, que começa criando um *socket* do servidor para aceitar conexões de clientes. Ela especifica um endereço IP ("127.0.0.1") e uma porta (5566) para aguardar conexões. Após isso, é verificado se o *socket* foi criado com sucesso e imprime uma mensagem de erro, caso ocorra o contrário.

O servidor, agora ativo, configura o *socket* para aceitar conexões, entrando em um estado de escuta utilizando a função “listen”, em que ele aguarda a conexão de dois clientes, representados pelos *sockets* “client_socket1” e “client_socket2”.

Quando ambos os clientes estiverem conectados, o jogo se inicia, com o servidor entrando em um estado de *loop*, em que ele recebe as escolhas dos jogadores usando a função “recv”. Após as escolhas terem sido feitas, o vencedor é determinado com base nas regras do jogo e, de acordo com quem venceu a rodada, o placar com os pontos de cada jogador é atualizado. Para finalizar a rodada ativa, o servidor envia mensagens de resultado para ambos os jogadores, informando quem venceu ou se houve empate e o placar atualizado.

O jogo continua até que um dos jogadores atinja cinco pontos ou até que algum dele se desconecte. Quando ocorre do jogo terminar normalmente, o servidor envia uma mensagem informando o vencedor para ambos os jogadores e encerra o jogo. Da mesma forma, caso algum dos jogadores se desconecte, o jogo apenas termina. Como descrito, nos dois casos o jogo é finalizado, o que é feito, juntamente com o encerramento das conexões, utilizando a função "close".

3.1.2. Clientes

Nos arquivos dos clientes, "client.c" e "client1.c", que são iguais, a função main() inicia o cliente, criando um *socket* para se conectar ao servidor, especificando o mesmo endereço IP ("127.0.0.1") e a mesma porta (5566) usados pelo servidor.

O cliente, então, entra em um *loop* que permite que jogador faça escolhas e as envia para o servidor usando a função "send". Após o envio das informações e do processamento por parte do servidor, os clientes recebem as mensagens de resultado pela função "recv" e imprime na tela o resultado do jogo.

O jogo continua até que um dos jogadores atinja cinco pontos ou até que algum deles se desconecte. Caso algum jogador vença, o servidor envia uma mensagem informando o vencedor, o cliente a imprime e encerra o jogo. Para o caso de algum dos jogadores se desconectar, o jogo apenas se encerra. Em ambos os casos, assim como o servidor, os clientes fecham a conexão com o servidor usando a função "close".

3.2. Listagem de Rotinas

3.2.1. Server

As principais funções do arquivo "server.c" serão listadas a seguir.

main():

- Declaração de variáveis:
  char *ip para armazenar o endereço IP.
  int port para armazenar o número da porta.
  int server_socket, client_socket1, client_socket2, n para sockets do servidor e clientes, e uma variável temporária.
  struct sockaddr_in server_addr, client1, client2 para estruturas de endereço.
  socklen_t addr_size para o tamanho do endereço.
  char buffer[1024] para um buffer de comunicação.
  char player1_move[10], player2_move[10], result[200], winner[200] para armazenar as escolhas dos jogadores, o resultado e o vencedor.
  int player1_points, player2_points para armazenar os pontos dos jogadores.
- Criação do socket do servidor e manipulação de erros, como a verificação de erros após a chamada socket().
- Configuração da estrutura server_addr com informações de IP e porta.
- Realização do bind do socket do servidor com o endereço configurado e verificação de erros.

- Início da escuta de conexões com listen.
- Aceitação de conexões dos dois jogadores com accept.
- Início de um loop infinito para gerenciar o jogo:
  Recebimento das escolhas dos jogadores com recv e exibição das escolhas.
  Avaliação das escolhas para determinar o vencedor ou um empate.
  Verificação se algum jogador atingiu 5 pontos, encerrando o jogo se necessário.
  Envio dos resultados para ambos os jogadores.
- Fechamento dos sockets dos jogadores com close.
- Fim do programa com return 0.

3.2.2. Client e Client1

As principais funções dos arquivos “client.c” e “client1.c” serão listadas a seguir.

main():

- Declaração de variáveis:
  char *ip para armazenar o endereço IP.
  int port para armazenar o número da porta.
  int sock para o socket do cliente.
  struct sockaddr_in addr para a estrutura de endereço do servidor.
  socklen_t addr_size para o tamanho da estrutura de endereço.
  char buffer[1024] para um buffer de comunicação.
  int n para uma variável temporária.
  char move[10], result[200] para armazenar a escolha do jogador e o resultado do jogo.
- Criação do socket do cliente e tratamento de erros, incluindo a verificação de erros após a chamada socket().
- Configuração da estrutura addr com informações de IP e porta.
- Conexão ao servidor com connect.
- Início de um loop infinito para gerenciar o jogo do cliente:
  Solicitação para o jogador fazer uma escolha ("pedra", "papel" ou "tesoura") e leitura da escolha com scanf.
  Verificação da escolha para garantir que seja válida e, em caso contrário, exibe uma mensagem de erro e continua o loop.
  Envio da escolha do jogador para o servidor com send.
  Recebimento do resultado do servidor com recv.
  Verificação do resultado para determinar se o jogo terminou e exibição apropriada de mensagens de encerramento.
  Se o jogo não terminou, exibe o resultado atual do jogo.
  Fechamento do socket do cliente com close quando o jogo termina.
- Exibição de uma mensagem indicando que o cliente foi desconectado do servidor.
- Fim do programa com return 0.

3.3. Protocolo Desenvolvido

O protocolo desenvolvido para a comunicação entre o servidor e os clientes é simples e baseado em texto. Os jogadores enviam suas escolhas como strings para o servidor, que determina o

vencedor e envia mensagens de resultado de volta para os clientes, utilizando as funções “send” e “recv” para a troca de informações.

O protocolo inclui mensagens de vitória quando um jogador atinge cinco pontos e a mensagem de empate quando não há vencedor. As mensagens são interpretadas pelo servidor para determinar o resultado do jogo, que envia as mensagens de encerramento do jogo quando necessário.

4. Conclusão

Ao final do desenvolvimento do projeto, foi possível perceber a importância da simplicidade e clareza na forma de o servidor interpretar as informações proveninetes dos clientes, para que não hajam problemas e para que a comunicação seja feita de forma eficiente. Além disso, também foi evidenciado a crucialidade do papel dos clientes na interatividade do jogo, que, novamente com o objetivo de evitar complicações, devem fornecer informações precisas sobre as ações ocorridas, além de necessitarem de uma interface amigável para os participantes.