DCC

DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Trabalho Prático Individual 02

Disciplina: Redes de Computadores
Objetivo: Este trabalho prático proporcionará aos alunos uma experiência prática no desenvolvimento de sistemas distribuídos simples, além de oferecer insights sobre os desafios e considerações envolvidos na criação de aplicativos cliente-servidor.
Linguagem: C
Tipo de conexão: UDP
Valor: 25 pontos
Data de entrega: xx/xx/2024

Descrição das atividades:

Neste trabalho prático, os alunos serão desafiados a desenvolver uma aplicação console que simule o funcionamento básico de aplicativos de streaming, com um servidor capaz de lidar com múltiplas conexões de clientes utilizando multithreading. Além disso, a comunicação entre o cliente e o servidor será realizada por meio de uma conexão UDP. O servidor enviará frases aleatórias para cada cliente conectado a cada 3 segundos, que serão exibidas no console do cliente.

Requisitos:

- Cliente:
    - Implementar um programa cliente em formato de aplicação console.
    - O cliente deve ter um menu para escolha de 3 filmes. (Figura 1)

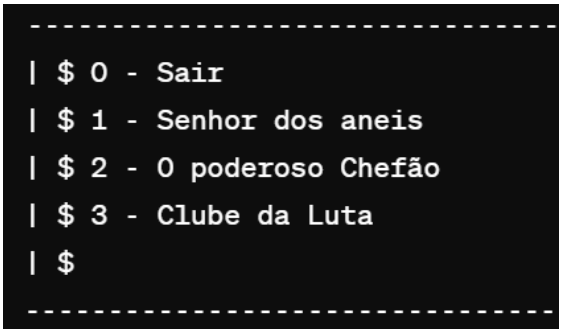

Figura 1 - Menu do Cliente

- Após selecionar uma opção, o cliente deve se conectar ao servidor utilizando UDP, informando a opção selecionada (1, 2 ou 3).
- O cliente deve exibir as frases recebidas do servidor no console. (Figura 2)

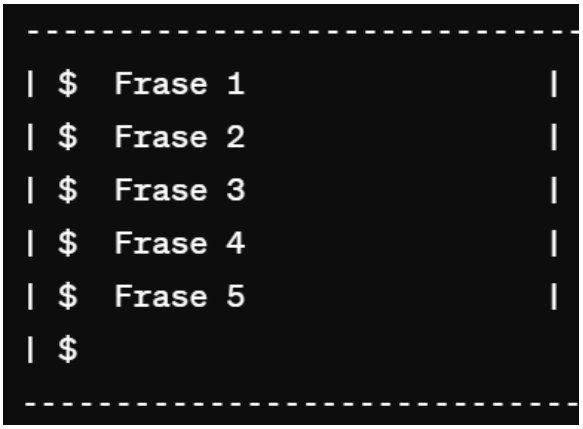

Figura 2 - Exibição das frases no cliente.

- Após recebidas as frases, o cliente deve voltar ao menu inicial. (Figura 1)
- Caso o cliente selecione Sair, o programa deverá ser encerrado.

- Servidor:
  - Implementar um programa servidor em formato de aplicação console capaz de lidar com múltiplas conexões de clientes utilizando multithreading.
  - O servidor deverá conter 3 listas de frases de filmes conforme listadas abaixo:
    - Senhor dos aneis:
      - "Um anel para a todos governar"
      - "Na terra de Mordor onde as sombras se deitam"
      - "Não é o que temos, mas o que fazemos com o que temos"
      - "Não há mal que sempre dure"
      - "O mundo está mudando, senhor Frodo”
    - O Poderoso Chefão:
      - "Vou fazer uma oferta que ele não pode recusar"
      - "Mantenha seus amigos por perto e seus inimigos mais perto ainda"
      - "É melhor ser temido que amado"
      - "A vingança é um prato que se come frio"
      - "Nunca deixe que ninguém saiba o que você está pensando"
    - Clube da Luta:

- "Primeira regra do Clube da Luta: você não fala sobre o Clube da Luta"
- "Segunda regra do Clube da Luta: você não fala sobre o Clube da Luta"
- "O que você possui acabará possuindo você"
- "É apenas depois de perder tudo que somos livres para fazer qualquer coisa"
- "Escolha suas lutas com sabedoria"

  - O servidor deve aceitar conexões UDP dos clientes e criar uma nova thread para cada cliente conectado.
  - A cada 3 segundos, o servidor deve enviar uma frase do filme selecionado pelo respectivo cliente para cada cliente conectado.
  - Após enviar todas as frases, a conexão deverá ser desfeita para aquele cliente.
  - A cada 4 segundos, o servidor deverá exibir na tela a quantidade de clientes conectados.

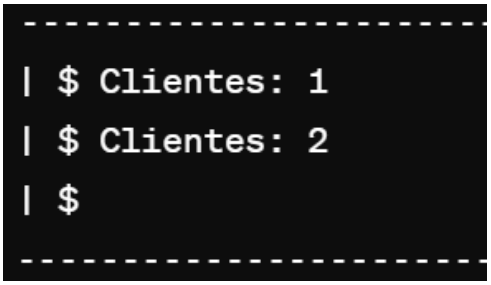

Figura 3 - Tela do servidor exibindo a quantidade de clientes conectados.

- Documentação:
  - A documentação deve ser composta de:
    - Pelo menos 3 (três) páginas.
    - Deve estar em formato PDF.
    - Deve conter o nome completo e matrícula do aluno.
    - Deve conter uma seção explicando o código do Servidor e do Cliente. Os trechos que envolvem conexão devem ser explicados detalhadamente. Isso inclui deixar explícito o funcionamento de métodos utilizados de qualquer biblioteca de rede.
    - Deve conter uma seção com prints dos testes simulando o caso em que o motorista aceita ou recusa a corrida. Todas as etapas devem estar explícitas nos prints. Cada print deve conter uma legenda com sua descrição.

Detalhes de Implementação:

- Servidor e Cliente devem estar em arquivos separados.

- O trabalho deve ser entregue utilizando a linguagem C (Não é permitido o uso de C++).
- O mesmo deverá ser desenvolvido para máquinas Linux (Não utilizem bibliotecas que não sejam compatíveis).
- Todas as linhas do código que remetem à conexão devem ser comentadas explicando sua função.
- Os alunos devem implementar o mecanismo de comunicação utilizando sockets UDP para permitir a comunicação entre o cliente e o servidor.
- O servidor deve ser capaz de gerenciar as conexões de forma concorrente utilizando multithreading, garantindo que cada cliente tenha uma thread dedicada para atendê-lo.
- Deve ser utilizado IPv4 e IPv6. Desta forma, o servidor deve ser configurado com o parâmetro que define o tipo de IP. O segundo parâmetro deverá ser a porta utilizada. Exemplo:
    - $ ./server ipv4 50501
    - $ ./server ipv6 50501

O cliente também deverá receber parâmetros. Primeiro a versão do IP (ipv4 ou ipv6), em seguida o IP do servidor e por fim a porta. Exemplo:

- $ ./client ipv4 127.0.0.1 50501
- $ ./client ipv6 ::1 50501

- **Dicas:**
    - Link para playlist de introdução à programação de programas de rede do professor Ítalo Cunha (https://www.youtube.com/watch?v=tJ3qNtv0HVs&list=PLyrH0CFXIM5Wzmbv-IC-qvoBejsa803Qk&index=1).
    - Link para artigo do Geeks for Geeks que explica como funciona multithreading no C. Lembre-se que a documentação requer explicações do código. (https://www.geeksforgeeks.org/handling-multiple-clients-on-server-with-multithreading-using-socket-programming-in-c-cpp/)
    - Utilize a metodologia de divisão e conquista. Divida o trabalho em problemas menores e solucione-os por partes.
    - Para guardar informações de um cliente, recomenda-se criar uma struct para o mesmo, contendo todas propriedades relevantes que estejam atreladas àquele cliente como no exemplo a seguir:

```
typedef struct {
    int socket;
      …
    int escolha; // Opção escolhida pelo cliente
    int frase; // Última frase exibida
} client_info;
```

- O exemplo anterior não é obrigatório. Fica a cargo do aluno como fazer esse gerenciamento, desde que esteja explícito na documentação como ele foi feito.
- Também é recomendável que o gerenciamento da conexão de cada cliente seja feito numa função específica. Esse tipo de função é comumente denominado “ClientHandler”.

Exemplos de execução:

Exemplo1:

Cliente 1	Cliente 2	Servidor	Obs
$ 0 - Sair 1 - Senhor dos Aneis 2 - O Poderoso Chefão 3 - Clube da Luta	$ 0 - Sair 1 - Senhor dos Aneis 2 - O Poderoso Chefão 3 - Clube da Luta	Clientes : 0	Cliente 2 seleciona Clube da Luta
$ 0 - Sair 1 - Senhor dos Aneis 2 - O Poderoso Chefão 3 - Clube da Luta	$ Frase 1	Clientes: 1	
$ 0 - Sair 1 - Senhor dos Aneis 2 - O Poderoso Chefão 3 - Clube da Luta	$ Frase 2	Clientes: 1	Cliente 1 seleciona Senhor dos Aneis
$ Frase 1	$ Frase 3	Clientes: 2	
$ Frase 2	$ Frase 4	Clientes: 2	
$ Frase 3	$ Frase 5	Clientes: 2	
$ Frase 4	$ 0 - Sair	Clientes: 1	Cliente 2 seleciona a

	1 - Senhor dos Aneis 2 - O Poderoso Chefão 3 - Clube da Luta		opção de sair.
$ Frase 5	<Programa encerrado>	Clientes: 1	
$ 0 - Sair 1 - Senhor dos Aneis 2 - O Poderoso Chefão 3 - Clube da Luta		Clientes: 0	

Entrega:

- Cada aluno deve entregar, além da documentação, o código fonte em C e um Makefile para compilação do programa.
- Será adotada média harmônica entre as notas da documentação e da execução, o que implica que a nota final será 0 se uma das partes não for apresentada.
- O Makefile deve compilar o cliente em um binário chamado “client” e o servidor em um binário chamado “server” dentro de uma pasta chamada "bin" na raiz do projeto.
- Seu programa deve ser compilado ao se executar apenas o comando “make”, ou seja, sem a necessidade de parâmetros adicionais.
- A entrega deve ser feita no formato ZIP, com o nome seguindo o seguinte padrão: TP2_MATRICULA.zip
- Pontuação:
    - Servidor (10 pontos):
        - Escutar e efetuar conexão (4pts)
        - Permitir múltiplas conexões (4pts)
        - Tratamento correto das opções de cada cliente (2pt)
    - Cliente (6 pontos):
        - Buscar conexão com o servidor (2pts)
        - Conectar e enviar opção selecionada (2pts)
        - Exibir corretamente as frases recebidas (2pts)
    - Documentação (9 pontos):
        - Descrição do código com explicações explícitas (6pts)
        - Prints (com descrições) do programa em funcionamento. (3pts)
- Penalidade por atraso:

- Para cada dia de atraso na entrega do trabalho ocorrerá uma penalidade de 25% da nota. Conforme tabela abaixo:

Dia	Penalidade
1 dia de atraso	25%
2 dias de atraso	50%
3 dias de atraso	75%
4 ou mais dias de atraso	100% (Nota zero no trabalho)