Universidade Estadual de Mato Grosso do Sul

Curso de Ciência da Computação

Disciplina de Redes de Computadores

Trabalho de Redes de Computadores

IMPLEMENTAÇÃO DE UM WEB DOCS COMPARTILHADO

Dourados (MS), 17 de maio de 2023.

Web Docs Compartilhado

O nosso Web Docs Compartilhado é um serviço que permite criar, editar e visualizar documentos de texto e, compartilhá-los com amigos e contatos profissionais. A ideia é criar documentos colaborativos em que cada integrante possa acrescentar novas ideias. Também é possível definir os tipos de iterações que as pessoas terão sobre um documento, como permitir que editem, comentem ou apenas visualizem o texto. Há uma série de usos para um Web Docs Compartilho. Tais serviços podem ser usados remotamente por pessoas fisicamente distantes na composição de trabalhos acadêmicos ou, redação de textos profissionais.

Objetivos do Trabalho

O principal objetivo do trabalho é praticar a programação com a biblioteca Socket utilizando o protocolo TCP com o servidor tratando múltiplas conexões. O trabalho consiste em desenvolver um sistema que permita ao usuário enviar parágrafos de texto para um arquivo texto presente em um servidor. Cada usuário poderá registrar seu próprio documento e, será permitido a um usuário interagir com documentos de outros usuário. O usuário também poderá "baixar" uma cópia do documento em sua máquina local. Os programas poderão funcionar tanto em IPv4 ou IPv6 utilizando o protocolo TCP.

Programas a serem Desenvolvidos

O trabalho prático consistirá na implementação de dois programas, o cliente e o servidor. Ambos receberão parâmetros de entrada como explicados a seguir:

cliente <ip/nome_maquina> <porta>

servidor <porta>

O primeiro programa (cliente) irá se conectar ao servidor definido pelo endereço IP (ou nome da máquina) e porta passados como parâmetro.

Já o servidor executará um serviço, que responderá paralelamente a todos os clientes que se conectarem permitindo o acesso ao mesmo tempo. A porta a ser executada é dada pelo parâmetro único do programa.

Protocolo Rede

O cliente poderá enviar 4 tipos de mensagens ao servidor: **registro (N), envio (S), recebimento (R) e lista (L).** O cliente obrigatoriamente deverá ficar preso em um loop infinito, recebendo as mensagens, imprimindo seus resultados e, estar pronto para novamente para uma nova mensagem.

O modo como o cliente e o servidor se comunicam (se usam string ou struct) não faz diferença, porém a saída do programa cliente deve seguir exatamente as mensagens da tabela a seguir e nada mais.

A mensagem de registro (N) deverá ser lida da seguinte maneira:

N [nome_do_usuário] [nome_do_arquivo]

Exemplo: N charmander diario

Ao receber uma mensagem de registro (N), o servidor deverá verificar se já não existe um usuário com esse identificador e, também, se já não existe um arquivo com o nome passado. O servidor deverá responder o cliente informando o sucesso ou não da operação.

O programa do cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de registro:

Mensagem na Tela	Explicação
N 0	Sucesso, usuário criado.
N 1	Erro, usuário já existe.
N2	Erro, arquivo já existe.

A mensagem de envio (S) deverá ser lida da seguinte maneira:

S [nome_do_usuário] [nome_do_arquivo] [[conteúdo_parágrafo]] [posição]

Exemplo: S picachu diário [Frase importante do dia.] [início]

Ao receber uma mensagem (S), servidor precisará verificar se já existe um arquivo com o nome especificado e inserir o texto fornecido na posição indicada com a seguinte indicação de origem: **[picachu] Frase importante do dia.**

O argumento posição especificado no envio poderá ser de três tipos: **Início** do texto no arquivo (neste caso, o texto ocupará o primeiro parágrafo do texto). **Fim** do texto no arquivo (neste caso, o texto ocupará o último parágrafo do texto). **P_número (P 5)** do texto no arquivo (neste caso, o texto ocupará a posição do parágrafo indicado. Neste caso, o quinto parágrafo).

O programa cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de envio:

Mensagem na tela	Explicação
S 0	Sucesso, mensagem foi salva.
S -1	Erro, usuário não existe.
S -2	Erro, arquivo não existe.
S -3	Erro, parágrafo indicado não existe no texto.

A mensagem de recebimento (R) deverá ser lida da seguinte maneira:

R [nome_do_usuário] [nome_do_arquivo]

Exemplo: R lucario diario

Ao receber uma mensagem (R), o servidor deverá verificar se o arquivo existe, se o usuário existe e, se o parágrafo indicado está disponível para inserção e, enviar para o cliente o resultado do comando e o conteúdo do arquivo.

O programa cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de recebimento:

Mensagem na Tela	Explicação
R 0 [texto]	Sucesso, no lugar de [texto], coloque o conteúdo do arquivo. Para facilitar a visualização, coloque

	o texto na linha abaixo a R 0. Deixar uma linha em branco entre os parágrafos.
R -1	Erro, usuário não existe.
R -2	Erro, arquivo não existe.
R -3	Erro, parágrafo não existe.

A mensagem de lista (L) deverá ser lida da seguinte maneira:

L [nome_usuário]

Exemplo: L Mewtwo

Ao receber uma mensagem (L), o servidor precisará verificar se o usuário existe.

O programa cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de lista:

Mensagem na Tela	Explicação
L 0 [arquivo 1 [número parágrafos]][dono] [arquivo 2 [número parágrafos]][dono] ……..	Sucesso, no lugar de [arquivo 1] coloque o nome dos arquivos. No lugar de [número parágrafos] coloque a quantidade de parágrafos daquele arquivo. No lugar de [dono] coloque o nome do usuário que criou o arquivo.Se não houver nenhum arquivo, deixe apenas “L 0”.
L -1	Erro, usuário não existe.

IMPORTANTE 1: A maneira como as mensagens serão gerenciadas pelo seu programa não fará parte da avaliação. Não é necessário que se guarde os dados em arquivos. Sendo permitida a perda de todos os dados ao fechar o servidor.

IMPORTANTE 2: Não colocar nenhum outro tipo de saída na tela, como mensagens de boas vindas ou “esperando comando”. Ao iniciar o cliente, não imprima nada e apenas espere a entrada de dados do usuário e imprima o resultado de acordo com as tabelas apresentadas.

EXECUÇÃO DO TRABALHO

Este trabalho deverá ser feito da seguinte forma:

1. Trabalho individual;
2. Poderá ser feito em C ou Python.
3. Você poderá escolher entre multiplexing, fork ou threads.
4. Fazendo em C ou Python você deverá enviar sua solução com um Makefile.

ENTREGA DO CÓDIGO

O código e a documentação devem ser entregues em um arquivo Zip. Dentro deste arquivo Zip devem conter um **readme.txt** com o nome do estudante e o comando para a execução do código

e, um arquivo PDF da documentação. Inclua todos os arquivos fontes (.c, .h, makefile, **não inclua executáveis ou arquivos objetos**), em um único diretório. Um **Makefile** deve ser fornecido para a compilação do código.

Parte desse trabalho envolve o aprendizado de como construir um makefile e utilizar a ferramenta make. Este makefile, quando executado sem parâmetros, irá gerar os dois programas: cliente e servidor, EXATAMENTE com esses nomes. No caso da linguagem python você poderá gerar um binário por meio do comando chmod +x.

Submissões onde os programas não sigam as especificações de parâmetro e nomes, makefile não funcionando ou arquivos necessários faltando **NÃO SERÃO CORRIGIDOS.** Programas que não compilem também não serão corrigidos.

DOCUMENTAÇÃO

O texto da documentação deve ser breve, de forma que o professor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deverá conter os seguintes itens:

1. O sumário do problema a ser tratado;
2. Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, procedimentos e das decisões de implementação;
3. Decisões de implementação que porventura estejam omissos na especificação;
4. Como foi tratada a retransmissão de mensagens;
5. Testes, mostrando que o programa está funcionando de acordo com as especificações, seguidos de sua análise;
6. Print screens mostrando o correto funcionamento do cliente e do servidor e exemplos de testes executados;
7. Conclusão e referências bibliográficas.

AVALIAÇÃO

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Todos os alunos deverão apresentar e explicar em laboratório o trabalho.

Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, estruturado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc);
- Execução correta do código com entrada de testes a serem definidas no momento da avaliação. As entradas de testes irão exercitar a funcionalidade completa do código e, testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto;
- A aderência ao protocolo especificado. Usarei ferramentas de captura de tráfego da rede (wireshark) e analisarei o código para verificar se a comunicação está da forma definida na documentação;
- Conteúdo da documentação que deve conter os itens mencionados anteriormente;
- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua portuguesa, qualidade textual e facilidade de compreensão);
- CASOS DE CÓPIA NÃO SERÃO TOLERADOS;
- Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o protocolo funcione, deve ser descrito na documentação de forma explícita.

REFERÊNCIA

Para uso do programa Make e escrita de Makefiles:
http://www.gnu.org/software/make/manual/make.html

http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

ENTREGA E APRESENTAÇÃO DO TRABALHO

Data: 12/07/2023

O trabalho deve ser feito individualmente.