UNIVERSIDADE ESTADUAL DO MATO GROSSO DO SUL
CIÊNCIA DA COMPUTAÇÃO

VICTOR MANOEL FERNANDES DE SOUZA

DOCUMENTAÇÃO: IMPLEMENTAÇÃO DE UM WEB DOCS COMPARTILHADO

Dourados - MS
2023

VICTOR MANOEL FERNANDES DE SOUZA

DOCUMENTAÇÃO: IMPLEMENTAÇÃO DE UM WEB DOCS COMPARTILHADO

Trabalho apresentado à Prof. Dr. Rubens Barbosa Filho, como parte da segunda avaliação deste bimestre pela Universidade Estadual do Mato Grosso do sul.

Docente: Profa. Dr. Rubens Barbosa Filho

Dourados - MS

2023

SUMÁRIO

1. Introdução

1.1 Visão Geral

O Trabalho consiste na implementação de um web docs compartilhado utilizando o modelo cliente e servidor. Alguns protocolos são definidos para essa troca de mensagem entre ambos. São eles: N(Registro), S(Envio), R(recebimento) e L(Listagem). O cliente mandará e receberá dados do servidor, onde os mesmos serão armazenados.

1.2 Principais Funções e Procedimentos do Cliente

Parâmetros: Deve ser passado o endereço e a porta que se deseja conectar

descrição: Esse procedimento representa o cliente propriamente dito. É nela que ocorreram as operações.

retorno: não tem retorno.

```
void Cliente(char *endereco, char *porta);
```

Parâmetros: Deve ser passado a estrutura onde será armazenado a mensagem que deve ser enviada, junto de seu tamanho.

descrição: Essa função irá ler e tratar a mensagem que se deseja enviar. Nela será identificado o protocolo. Se estiver tudo certo, atribui a mensagem e o tamanho da mensagem lidas aos atributos da variável buffer.

retorno: retorna por referência a variável buffer; retorna 0 caso o protocolo seja válido, e 1 caso não seja.

```
unsigned Menu(struct Mensagem *buffer);
```

Parâmetros: uma string que representa a mensagem

descrição: Verifica se a mensagem está no formato correto (protocolo S).

retorno: 0 se for válido e 1 se não for válido.

```
unsigned CasoEspecialS(char *string);
```

1.3 Principais Funções e Procedimentos do Servidor

Parâmetros: uma string que representa a mensagem vinda do cliente, e o socket.

descrição: Procedimento responsável por identificar o protocolo, fazer as operações necessárias e enviar uma resposta para o cliente.

retorno: não retorna nada.

```
void protocoloDeRede(char *protocolo, int socket);
```

Parâmetros: a mensagem vinda do cliente

descrição: caso a mensagem seja identificada como protocolo N, essa função será chamada. Ela irá separar os parâmetros da mensagem adequadamente em: usuário e arquivo(arg1, arg2), para que a função Registrar seja chamada.

retorno: retorna o retorno da função Registrar.

```
unsigned protocoloN(char *protocolo);
```

Parâmetros: Usuário e arquivo que se deseja registrar

descrição: Caso o usuário e o arquivo não exista, ambos serão registrados em uma lista. E um arquivo de texto será criado para salvar os parágrafos.

retorno: retorna se ocorreu sucesso ou falha.

```
unsigned Registrar(char *arg1, char *arg2);
```

Parâmetros: mensagem vinda do cliente

descrição: caso a mensagem seja identificada como protocolo S, essa função será chamada. Ela irá separar os parâmetros da mensagem adequadamente em: usuário, arquivo, conteúdo textual e a posição desse conteúdo(arg1, arg2, arg3, arg4),

retorno: retorna se ocorreu sucesso ou falha.

```
int protocoloS(char *protocolo);
```

Parâmetros: Usuário, arquivo, conteúdo textual e posição do parágrafo.

descrição:Responsável pela inserção de um parágrafo em um arquivo especificado, Caso o usuário e o arquivo exista, e o parágrafo especificado seja válido, o arquivo será aberto e o conteúdo textual será inserido no número de parágrafo indicado.

retorno: retorna se ocorreu sucesso ou falha.

```
int Envio(char *arg1, char *arg2, char *arg3, char *arg4);
```

Parâmetros: mensagem vinda do cliente.

descrição: caso a mensagem seja identificada como protocolo R, essa função será chamada. Ela irá separar os parâmetros da mensagem adequadamente em usuário e arquivo (arg1, arg2), para que a função recebimento seja chamada.

retorno: retorna o retorno da função recebimento.

```
char *protocoloR(char *protocolo);
```

Parâmetros: Usuário e arquivo que se deseja registrar

descrição: Caso o usuário e o arquivo exista, o programa irá indicar os parágrafos que esse usuário criou nesse arquivo.

retorno: retorna os parágrafos

```
char *recebimento(char *arg1, char *arg2);
```

Parâmetros: mensagem vinda do cliente

descrição: caso a mensagem seja identificada como protocolo L, essa função será chamada. A função irá separar o argumento para a chamada de função Lista.

retorno: retorna o retorno da função Lista

```
char* protocoloL(char *protocolo);
```

Parâmetros: um usuário

descrição: caso o usuário exista, a função vai listar todos os arquivos disponíveis, seu dono e a quantidade. A função irá formatar a mensagem que deve ser entregue ao cliente.

retorno: retorna a lista de arquivos, seus respectivos dono, junto da quantidade de parágrafos de cada arquivo

```
char *Lista(char *arg1);
```

1.4 - Como os Dados Serão Salvos no Servidor

Os dados no servidor serão salvos por meio de arquivos. Quando o programa for executado, automaticamente será criado dois arquivos: ArquivosUsuarios que irá armazenar os usuários armazenados, e o NomesArquivos que irá guardar o nome de cada arquivo feito.

Caso ocorra um Registro (N), um novo arquivo de texto será criado, o usuário que fez o registro irá ser o dono desse arquivo. O primeiro elemento do arquivo criado pelo usuário será o seu nome.

Caso ocorra um Envio(S), o texto será salvo no parágrafo indicado do arquivo também indicado.

1.5 - Decisões de Implementação

Decidi que o trabalho deveria ser implementado com alocação dinâmica de memória, pois além de ser o mais próximo da realidade, agrega muito na questão de conhecimento.

Decidi fazer em linguagem C pois quanto mais familiarizado eu estiver com a linguagem, menos dificuldades terei com PPD ou outros projetos. Com certeza no futuro, em trabalhos complexos, existirá todo tipo de dificuldades envolvendo pesquisa, e acredito que a linguagem C seja um bom treinamento para adquirir a habilidade de correr atrás de conhecimentos específicos (conhecimento que não se acha no google facilmente, por exemplo).

2- Cliente

-Partindo do princípio onde o servidor está executando, e a porta e ip foram passados adequadamente, o procedimento Cliente é chamado.

-Logo após, as variáveis para a criação do socket serão inicializadas.

-O código entra em loop infinito.

- A função Menu é chamada. Ela irá ler(alocado dinamicamente por meio da função getline) e tratar os dados para que o protocolo seja enviado adequadamente ao servidor. A função retorna 0 se o protocolo for válida e 1 caso não seja.

- se o protocolo for válido

-o socket é criado e a solicitação de conexão é enviada ao servidor.

- caso a conexão seja feita, ele envia a mensagem e espera uma resposta do servidor.

- uma mensagem com o tamanho do protocolo especificado é enviado, o servidor aloca a memória necessária e envia uma mensagem dizendo que a memória foi alocada. o cliente espera até que essa mensagem chegue. Quando a mensagem chega, o protocolo especificado será enviado. O servidor recebe o protocolo, interpreta e envia o tamanho da resposta.

- O cliente aloca o tamanho da resposta, avisa o servidor que a memória foi alocada, e o servidor manda a resposta.

- O cliente desconecta.

-fim se

-imprime a resposta na tela.

-fim do loop

3- Servidor

Pelo fato do servidor ser um pouco extenso, irei citar suas principais funcionalidades, com ênfase nas funções: Registrar, Envio, recebimento e Lista.

O servidor começa fazendo os procedimentos necessários para que um servidor inicie, até esperar pelas conexões.

caso haja uma conexão um processo filho é criado para atender essa conexão e o processo pai espera por mais conexões. Esse modelo permite que o servidor sirva o cliente paralelamente.

Caso ocorra uma conexão, o programa irá identificar o protocolo e eventualmente os argumentos serão enviados para a função do protocolo especificado.

Protocolo N: O responsável por esse protocolo é a função Registrar. Ela tem dois parâmetros: Nome de usuário que o cliente deseja registrar, e o nome do arquivo. Caso ambos não existam, um arquivo de texto é criado, O usuário será adicionado à lista de usuários e o nome do arquivo será adicionado à lista de nome de arquivos.

Protocolo S: O responsável por esse protocolo é a função Envio. A mesma tem quatro parâmetros: Nome de usuário, arquivo, conteúdo textual e o parágrafo indicado. Caso o usuário, arquivo e o parágrafo indicado não sejam válidos, o programa retorna erro. Já com os parâmetros válidos, será possível ter três casos de inserção de um parágrafo: Início, Meio ou Fim.

Protocolo R: O responsável por esse protocolo é a função recebimento. Ela deve receber dois parâmetros: Nome de usuário e arquivo. caso ambos sejam válidos, a função vai indicar os parágrafos deste usuário. Depois da função achar um parágrafo, ele formata a string para ficar pronta para impressão no cliente.

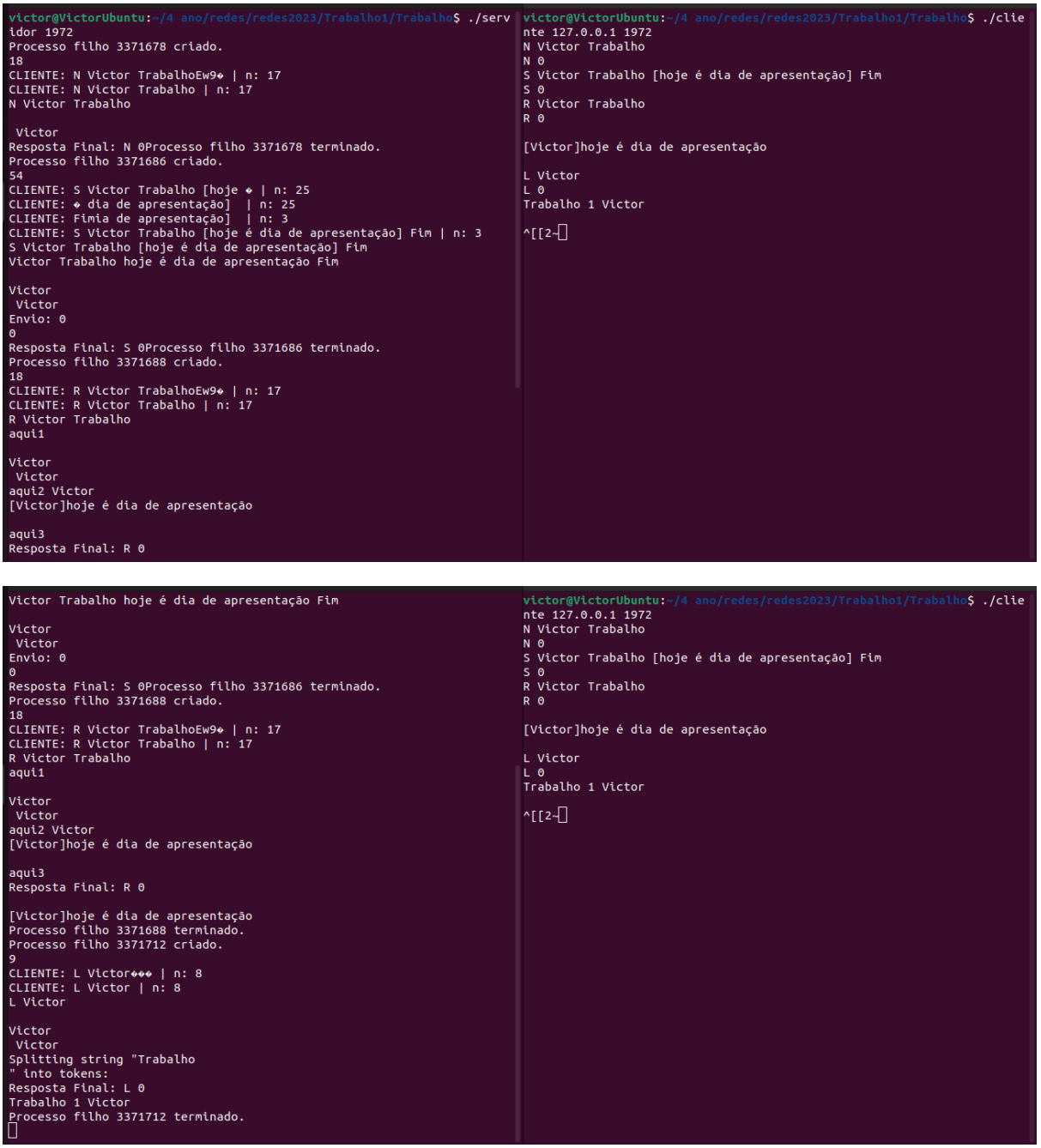
Protocolo L: O responsável por esse protocolo é a função Lista. Ela irá receber somente um parâmetro: Usuário. Caso seja um usuário válido, a função abre cada arquivo, lê o dono que se encontra no primeiro elemento do cada arquivo, e a partir daí começa a contagem de parágrafos. A função também faz o trabalho de formatação da string para que ela fique do jeito que o cliente deve ver.

4- Tratamento de Retransmissão de Mensagem

Logo após a mensagem ser interpretada, e a mensagem de resposta formatada corretamente, a retransmissão de mensagem ocorrerá igual a transmissão do cliente para o servidor. O servidor manda o tamanho do arquivo; O cliente recebe o tamanho, aloca a memória necessária e avisa o servidor que a memória foi alocada. o servidor manda a mensagem.

5- Testes

Abaixo segue alguns testes de protocolo



Decidi não colocar muitos casos de testes, pois o documento ia ficar poluído, mas todas as funcionalidades foram testadas corretamente. Acredito que esteja tudo nos conformes.

6- Conclusão

Com certeza um dos trabalhos mais produtivos se tratando de conhecimento. Utilizei praticamente todo conhecimento de programação em c. Acredito que o trabalho poderia ter ficado melhor, se eu não tivesse deixado para última hora. Por isso, talvez possa ser

encontrado um bug aqui ou ali. Outra razão para que possa ter algum bug é que, para manipular a memória em C, deve-se tomar muito cuidado. Qualquer erro o programa irá parar, e pode ser que em alguma entrada específica ocorra algum acesso de memória inválida. Porém todos os casos de testes que eu fiz não deram erros.

7- Referências

https://cplusplus.com/reference/

www.gnu.org/software/make/manual/make.html

https://www.comp.uems.br/~ojacques/SO/

https://www.youtube.com/watch?v=yovObHcGa1g&t=157s