UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MIKAELA

LEONARDO DEORCE LIMA DE OLIVEIRA

**1º TRABALHO PRÁTICO:**

NETMAP

VITÓRIA

2018

MIKAELA

LEONARDO DEORCE LIMA DE OLIVEIRA

**1º TRABALHO PRÁTICO:**

NETMAP

Trabalho apresentado à disciplina Estrutura de Dados I (INF09292) do Curso de Ciência da Computação da Universidade Federal do Espírito Santo no 2º semestre do ano 2018, como requisito para avaliação.

Orientador: Prof.ª Patrícia Dockhorn Costa

VITÓRIA

2018

**RESUMO**

TEXTO

**SUMÁRIO**

TEXTO

**1 INTRODUÇÃO**

TEXTO

**2 BIBLIOTECAS**

**2.1 Leitura e execução de comandos da entrada**

**2.1.1 Visão geral**

A biblioteca que define as funções de manipulação do arquivo de entrada foi denominada netmap.h. O arquivo de entrada é do tipo texto e contém comandos de operação do netmap. A biblioteca possui duas funções, sendo uma responsável pela criação do netmap a partir do arquivo e outra responsável pela escrita do netmap em linguagem DOT.

A sintaxe esperada de cada linha do arquivo de entrada denominado entrada.txt consiste em nome do comando seguido de seus argumentos, sendo um máximo de dois argumentos por comando e mínimo de nenhum argumento e cada item da mesma linha deve ser separado por um espaço. A implementação desenvolvida permite flexibilidade na quantidade de espaços entre itens e após o último item em uma linha.

**2.1.2 CriaNetmap**

A função lê e executa os comandos escritos no arquivo de entrada. Ela recebe o nome a quantidade de argumentos passados para o programa (int) e o vetor de strings contendo o nome do arquivo de entrada (char\*\*). Não há retorno. Após execução: comandos de entrada executados e arquivos criados de acordo.

Ao iniciar, a função verifica

**2.1.3 Leitura**

TEXTO

**2.1.4 ExecutaComando**

TEXTO

**2.1.5 ImprimeNetMap**

TEXTO

**2.2 Lista de terminais**

**2.2.1 Visão geral**

A biblioteca que define as funções de manipulação da lista de terminais do netmap foi denominada terminal.h. Esta é dividida em funções principais, que atendem requisitos de funcionamento do netmap, e funções auxiliares, que facilitam a comunicação com as implementações de outras bibliotecas.

A lista de terminais é do tipo sem sentinela e simplesmente encadeada. A escolha desse tipo de lista parte da sua simplicidade e da variedade que trás ao projeto. Abaixo seguem descrições dos elementos principais presentes na implementação da biblioteca terminal.h.

**2.2.2 Estrutura do terminal (struct terminal)**

IMAGEM

**2.2.3 Estrutura de uma lista/célula de terminais/terminal (struct celTerm)**

Uma lista sem sentinela, simplesmente encadeada não circular é manipulada com o auxílio de um ponteiro para a primeira posição da lista. Portanto, a lista e suas células compartilham a mesma estrutura.

IMAGEM

**2.2.4 CadastraTerminal**

A função adiciona um terminal à lista de terminais do netmap. Ela recebe como parâmetros o nome do terminal (char\*), o nome da localização do terminal (char\*) e o ponteiro para lista de terminais (CelTerm\*). A função retorna a lista de terminais atualizada, pois esta é sem sentinela e o endereço da lista é sujeito à mudança. Após execução: uma estrutura terminal deve ter sido criada e preenchida com os dados fornecidos; uma célula criada e encadeada na lista preenchida com ponteiro para o terminal.

Ao iniciar, duas células (CelTerm\*) são declaradas, uma auxiliar apontando para a mesma célula para a qual a lista fornecida aponta e outra alocada dinamicamente que será encadeada na lista. Em seguida, chama-se a função interna criaTerminal que aloca um tipo terminal preenchido com nome e localização fornecidos e que não está conectado a roteador.

Para facilitar a impressão do arquivo saida.dot, novas células são encadeadas ao final da lista de terminais. Portanto, a célula nova recebe um ponteiro para o terminal criado e seu ponteiro para a próxima célula recebe NULL.

A função então verifica se a lista fornecida está vazia. Caso esteja, o ponteiro para a lista passa a apontar para a nova célula, tornando-a o primeiro e único elemento da lista. Caso não esteja, a célula auxiliar percorre a lista até a última posição, permitindo o encadeamento da nova célula ao fim da lista.

**2.2.5 RemoveTerminal**

A função remove um terminal existente da lista de terminais do netmap. Ela recebe como argumentos o nome do terminal (char\*) e o ponteiro para a lista de terminais (CelTerm\*). A função retorna a lista de terminais atualizada, pois esta é sem sentinela e o endereço da lista é sujeito à mudança. Após execução: o tipo terminal cujo nome é idêntico ao nome fornecido será liberado; a célula que aponta para o tipo terminal em questão é desencadeada da lista e liberada.

Ao iniciar, são declaradas três células auxiliares: uma que recebe a célula da lista fornecida que aponta para o terminal cujo nome foi fornecido através da função auxiliar BuscaTerminal; outra que aponta para a primeira posição da lista fornecida; e outra não inicializada encarregada de reter a posição anterior a medida que a lista é percorrida.

A lista de terminais é percorrida pelas células auxiliares. Uma vez que o terminal não é encontrado na lista, uma mensagem de erro é escrita em um vetor que é passado como parâmetro para a função EscreveLOG, que imprime mensagens de erro no arquivo log.txt.

Caso o terminal seja encontrado, a função verifica se sua célula é a primeira da lista. Caso seja, a lista de terminais passa a apontar para a próxima posição (que pode ser NULL). Caso não seja, a posição anterior à célula a ser removida passa a apontar para a próxima posição. Por fim, célula e seu tipo terminal são liberados.

**2.2.6 ConectaTerminal**

A função conecta um terminal a um roteador, ambos existentes no netmap. Ela recebe como argumentos o nome do terminal (char\*), o nome do roteador (char\*), o ponteiro para a lista de terminais (CelTerm\*) e o ponteiro para a lista de roteadores (char\*). Não há retorno pois a lista de terminais permanece inalterada. Após execução: o tipo terminal do terminal cujo nome foi fornecido agora aponta para a célula do roteador cujo nome foi fornecido na lista de roteadores.

Ao iniciar, é declarada uma célula de terminal auxiliar que recebe um ponteiro para a célula do terminal cujo nome foi fornecido através da função BuscaTerminal, que retorna NULL caso não encontre o terminal passado. Se a célula for NULL, o terminal não existe no netmap e uma mensagem de erro é então criada e passada como parâmetro para a função EscreveLOG, que imprime a mensagem em log.txt.

Caso o terminal seja encontrado, a função verifica se o roteador existe no netmap, e resgata sua célula caso exista através da declaração de uma célula auxiliar para roteador e chamada da função BuscaRoteador. Novamente, se a célula receber NULL, uma mensagem de erro é criada e passada como parâmetro para a função EscreveLOG, que imprime a mensagem em log.txt.

Se terminal e roteador existem no netmap, o tipo terminal do terminal cujo nome foi fornecido passa a apontar para a célula do roteador cujo nome foi fornecido na lista de roteadores.

**2.2.7 DesconectaTerminal**

A função desconecta um terminal existente no netmap de seu roteador. Ela recebe como argumentos o nome do terminal (char\*) e o ponteiro para a lista de terminais (CelTerm\*). Não há retorno pois a lista de terminais permanece inalterada. Após execução: o tipo terminal do terminal cujo nome foi fornecido agora aponta para NULL, indicando que está desconectado.

Ao iniciar, é declarada uma célula de terminal auxiliar que recebe um ponteiro para a célula do terminal cujo nome foi fornecido através da função BuscaTerminal, que retorna NULL caso não encontre o terminal passado. Se a célula for NULL, o terminal não existe no netmap e uma mensagem de erro é então criada e passada como parâmetro para a função EscreveLOG, que imprime a mensagem em log.txt.

Caso a célula do terminal seja encontrada, o tipo terminal para o qual ela aponta passa a apontar para um roteador NULL, indicando que o terminal está desconectado.

**2.2.8 EnviarPacotesDados**

A função avalia a possibilidade de envio de pacotes de dados de um terminal para outro. Ela recebe como argumentos os nomes de ambos os terminais (char\*) e a lista de terminais (CelTerm\*). Não há retorno. Após execução: nomes dos terminais e resultado SIM ou NÃO escrito em arquivo saida.txt.

Ao inciar, declara células auxiliares para os terminais passados e, com o auxílio da função BuscaTerminal, verifica se os terminais existem no netmap. Caso algum terminal não exista, uma mensagem de erro é criada e passada como parâmetro para a função EscreveLOG, que imprime a mensagem em log.txt.

A função então verifica se os terminais passados estão conectados a algum roteador, caso algum não esteja, a comunicação entre eles não é possível, portanto uma mensagem de resultado NÃO é criada e passada como parâmetro para a função EscreveSAIDA, que imprime a mensagem em saida.txt.

Caso ambos estejam conectados a um roteador, o nome de cada roteador é resgatado e comparado. Se o nome for o mesmo, ambos estão conectados ao mesmo roteador e o envio de dados é possível, portanto uma mensagem SIM é criada e passada como parâmetro para a função EscreveSAIDA. Caso estejam conectados a roteadores diferentes, é chamada a funcaoBusca, detalhada na descrição de roteador.h, que retorna 1 para SIM e 0 para NÃO. O procedimento de escrita do resultado segue de acordo com o retorno da funcaoBusca.

**2.2.9 FrequenciaTerminal**

A função conta a quantidade de terminais cadastrados no netmap que são de uma determinada localização. Ela recebe como argumentos o nome da localização (char\*) e a lista de terminais. Não há retorno. Após execução: o nome da localização e sua frequência absoluta de ocorrência na lista de terminais são escritos no arquivo de texto saida.txt.

Ao iniciar, é declarada uma variável do tipo int encarregada de armazenar o valor da contagem de ocorrências de terminais pertencentes à localização fornecida (inicialmente zero). Também é declarada uma célula auxiliar de terminal inicializada apontando para a primeira posição da lista de terminais.

A função anda pela lista com a célula auxiliar e compara a localização de cada terminal apontado por cada célula com a localização desejada, realizando a contagem com o auxílio da variável declarada no início.

Por fim, uma mensagem contendo o nome da localização e sua frequência é passada como argumento para a função EscreveSAIDA, que imprime a mensagem em saida.txt.

**2.3 Lista de roteadores**

**2.3.1 Visão geral**

A biblioteca que define as funções de manipulação da lista de roteadores do netmap e seus enlaces foi denominada roteador.h. Esta é dividida em funções principais, auxiliares internas, que serão usadas somente dentro da própria biblioteca para deixar o código mais legível e para o reuso nas funções, e auxiliares externas, que serão usadas em outras bibliotecas, como na terminal.h.

**2.3.2 Estrutura do roteador (struct roteador)**

TEXTO

**2.3.3 Estrutura da célula de uma lista de roteadores (struct celRot)**

Em vários problemas é necessário buscar a célula de um roteador em uma lista, por exemplo, ao remover um roteador da lista do netmap e ao desconectar um roteador do outro, por isso, se faz útil uma função para realizar a busca e com isso é necessário saber qual a célula anterior ao roteador encontrado a fim de fazer o desencadeamento daquela célula, logo, para facilitar, foi pensado em uma lista duplamente encadeada.

IMAGEM

**2.3.4 Estrutura da lista de roteadores (struct lsRot)**

Foi pensada em uma lista com sentinela.

IMAGEM

**2.3.5 CadastraRoteador**

A função irá adicionar um roteador à lista de roteadores do netmap. Os argumentos são um ponteiro para o nome do roteador (char\*), um ponteiro para o nome da operadora do roteador(char\*) e o ponteiro pra lista de roteadores do netmap (LsRot\*). Não há retorno na função. Ao final, o roteador deve ter sido criado, e a célula que aponta para ele encadeada na lista de roteadores do netmap.

Primeiro, cria-se um tipo roteador, o qual armazena o nome e a operadora do roteador e inicializa a lista de roteadores conectados, sendo ela vazia. Utiliza-se uma função auxiliar interna para fazer isso, chamada criaRoteador.

Depois cria uma célula que apontará para o roteador criado. E então encadeia essa na célula na lista de roteadores do netmap. Os roteadores cadastrados no netmap serão adicionados sempre ao final da lista. Utiliza-se uma função auxiliar interna chamada EncadeiaRoteador para fazer isso, que será utilizada em outra função.

**2.3.6 RemoveRoteador**

A função irá remover um roteador existente da lista de roteadores do netmap. Não há retorno e os argumentos são o nome do roteador (char\*), a lista de roteadores do netmap (LsRot\*) e a lista de terminais do netmap, que é do tipo void\* (mas na verdade é CelTerm\*) como uma forma de não ter erros na compilação já que a roteador.h inclui a terminal.h e vice-versa. Ao final, o roteador deve ter sido removido do netmap e desconectado de todos os roteadores e terminais que antes estava.

Primeiro a função busca a célula do roteador na lista de roteadores do netmap. Caso encontre-o, é preciso desconectá-lo de todos os outros roteadores ao qual está conectado, nisso é feito um loop que percorre a lista de roteadores conectados chamando a função DesconectaRoteadores.

Depois disso, é necessário desconectar os terminais desse roteador, utiliza-se a função DesconectaRoteador que está definida na terminal.h. O roteador é desencadeado utilizando a função DesencadeiaRoteador da lista de roteadores do netmap e então liberado usando a função auxiliar LiberaTipoRoteador.

Caso o roteador não exista no netmap, uma mensagem de erro é escrita no log.txt e a função encerrada.

**2.3.7 ConectaRoteadores**

A função irá conectar dois roteadores existentes no netmap. Não há retorno e os argumentos são o nome de cada roteador (char\*) e a lista de roteadores do netmap (LsRot\*). Ao final, cada roteador deve estar na lista de roteadores conectados um do outro.

Primeiro a função busca as células dos roteadores na lista de roteadores do netmap para ter acesso às listas de roteadores conectados e o tipo roteador de cada um. Caso encontre-os, novas células são criadas e apontam para os roteadores, cada uma. Note que não é criado um novo tipo roteador.

Então a célula que aponta para um roteador será encadeada na lista de roteadores do outro roteador e vice-versa usando a função EncadeiaRoteador.

Caso um ou os dois roteadores não existam no netmap, uma mensagem de erro é escrita no log.txt e a função encerrada.

**2.3.8 DesconectaRoteadores**

A função irá desconectar dois roteadores existentes e conectados no netmap. Não há retorno e os argumentos são o nome de cada roteador (char\*) e a lista de roteadores do netmap. Ao final, a lista de roteadores conectados de cada roteador não deve conter uma célula apontando para o roteador um do outro.

Primeiro a função busca as células dos roteadores no netmap para ter acesso à lista de roteadores conectados de cada um.

Depois é preciso buscar o roteador1 na lista de roteadores conectados do roteador2 com a função auxiliar BuscaRoteador para poder chamar a função DesencadeiaRoteador passando como argumento a célula encontrada e a lista de roteadores conectados do roteador2 e vice-versa. Para finalizar, as células são liberadas.

Caso um ou os dois roteadores não existam no netmap ou eles não estejam conectados, uma mensagem de erro é escrita no arquivo log.txt e a função encerrada.

**2.3.9 FrequenciaOperadora**

A função contará quantos roteadores do netmap são da operadora que é passada como entrada. Não há retorno, já que a quantidade é escrita diretamente no arquivo saida.txt. Os argumentos de entrada são o nome da operadora (char\*) e a lista de roteadores do netmap.

Um ponteiro do tipo CelRot\* percorre a lista de roteadores do netmap verificando qual a operadora é cada roteador e se for igual à operadora de entrada, então o contador acrescenta +1.

**2.3.10 funcaoBusca**

Outra função importante na roteador.h é a funcaoBusca que tem papel principal na função EnviarPacotesDados definida na terminal.h. Ela foi implementada no roteador.c porque é preciso manipular estruturas que são definidas somente na biblioteca roteador.h.

A função tem como argumento uma célula do tipo roteador (CelRot\*), o nome dor roteador que está buscando (char\*), um vetor vet que irá guardar o nome dos roteadores que já foram analisados e um contador i (int\*) que indica a primeira posição livre do vetor. O retorno é 1, caso o roteador seja encontrado e 0, caso contrário.

É usado um ponteiro p do tipo CelRot\* que irá apontar, inicialmente, para a primeira célula da lista de roteadores conectados do roteador de entrada.

Em um loop (irá parar quando p = NULL) são feitas verificações sobre o roteador que p aponta. Primeiro, se o nome do roteador está contido no vetor, usando uma função auxiliar chamada BuscaNomeVet. Caso esteja, p recebe o endereço da próxima célula, caso contrário, verifica se o nome do vetor é igual ao vetor que está buscando. Se não for o roteador, então guardará o nome desse no vetor e é chamada a funcaoBusca. Após a chamada, verifica se o retorno foi 1, se não, p recebe a próxima célula.

Um exemplo do funcionamento dessa função é dado pelo esquema a seguir. Ele mostra o caso em que o roteador é encontrado. O roteador que se busca é “Rot5”.

IMAGEM CAIXA

IMAGEM CAIXA

CAIXA IMAGEM

FAZER O **OBS** ABAIXO VIRAR COMENTÁRIO

**Obs:** desconsiderou-se as setas dos ponteiros ant e o tipo roteador foi colocado “dentro” da célula para facilitar a representação.

Em p17, o roteador é encontrado, então a função encerra retornando 1. Observe que, mesmo se não encontrasse o roteador, a função pararia retornando 0. Foi pensado da forma a “varrer” todas as listas possíveis com o while e a chamada recursiva da função, mas evitando um loop infinito, utilizando o vetor.

**3 CONCLUSÃO**

TEXTO

**REFERÊNCIAS**

TEXTO