UFES

Universidade Federal do Espírito Santo - Centro Tecnológico

Departamento de informática

Estrutura de dados I (INF09292) – Turma EC

Professora: Patrícia Dockhorn Costa

Estrutura de Dados: NETMAP

Humberto Giuri Calente

Sérgio Vago Rodrigues de Melo

Outubro/2018

Sumário

1. Introdução

O trabalho apresentado é um simulador de uma rede, como a internet, que permite a passagem de dados entre terminais distribuídos.

A simulação proposta pelo trabalho apresenta terminais e roteadores que estão ligados uns com os outros de maneira que os roteadores podem estar ligados entre si e os terminais só podem estar ligados a um roteador. Cada roteador inserido pode estar conectado a vários outros roteadores, assim simulando, como na vida real, a possível troca de informações entre os componentes do NETMAP tendo os roteadores como canais de troca de informações utilizados pelos terminais.

Será passado um arquivo para os casos de teste, o entrada.txt em que será aberto pelo programa e dele serão extraídas as informações para rodar o programa. O programa criará o saída.txt, que será o arquivo de saída padrão, contendo os dados básicos do NETMAP, criará também o log.txt que indicará todos os erros que aconteceram durante a execução do programa. E por último, o saída.dot que indicará o estado do NETMAP sempre que pedido, utilizando a linguagem DOT e o GraphViz para gerar uma imagem do NETMAP.

No Roteador.c tem a estrutura lista de roteadores e a célula do tipo roteador onde essas estruturas possuem as características do roteador. Será criada e inicializada uma lista de roteadores que conterá uma célula para cada roteador do NETMAP que será inserido quando requisitado pelo entrada.txt, e para cada célula de roteador nessa lista haverá uma outra lista de roteadores, que indicará os roteadores que estão conectados no mesmo. Nessa parte do trabalho será possível observar qualquer característica dos roteadores como exemplo, a função FrequenciaOperadora que diz quantos roteadores são de uma determinada operadora e estão no NETMAP.

No Terminal.c estarão as estruturas lista de terminais e célula do tipo terminal, que apresentarão as características de cada terminal. Como no caso do roteador também será criada e inicializada uma lista de terminais contendo todos os terminais que serão criados no NETMAP respeitando o entrada.txt, mas cada célula do tipo terminal terá um respectivo roteador, que será o

roteador ao qual essa célula estará conectada. Também como no Roteador.c, no Terminal.c o trabalho resolve e analisa o que é voltado aos terminais, como exemplo a função FrequenciaTermianal que imprimi quantos terminais são de uma certa localização e a ImprimeTerminal que printa as características de um terminal.

2. Implementação

1. Roteador.c:
    - Contém 3 structs: roteador, celulaR, listaR. Na struct roteador possui o nome, a operadora e um ponteiro que irá apontar para um roteador que será o nó da lista de roteadores que aquele roteador estará conectado, os enlaces. Já a celulaR define a célula que contemplará a última struct, que é a listaR, sendo a sentinela para a lista simplesmente encadeada de roteadores. Na celulaR possui um ponteiro para um roteador e um ponteiro para uma celulaR com nome prox.

    - Função IniciaRoteador: Inicializa um novo roteador.
        - Inputs: Um nome e uma operadora.
        - Output: Um novo roteador inicializado.
        - Pré-condição: Nome e operadora validos.
        - Pós-condição: roteador inicializado e campo proximoEnlace aponta para NULL.

    - Função IniciaCelulaR: Inicializa uma nova célula do tipo roteador.
        - Inputs: Um ponteiro para um roteador.
        - Output: Uma célula do tipo roteador inicializada.
        - Pré-condição: Roteador não nulo.
        - Pós-condição: Célula do tipo roteador inicializada e campo prox aponta para NULL.

    - Função IniciaListaRoteadores: Inicializa uma sentinela da lista de roteadores.
        - Inputs: Nenhum.
        - Output: Sentinela inicializada.
        - Pré-condição: Nenhuma.
        - Pós-condição: Sentinela da lista de retorno existe e os campos primeiro e último apontam para NULL.

- Função CadastraRoteador: Cadastra um roteador a uma lista de roteadores.
    - Inputs: Um ponteiro para a lista de roteadores e um para um roteador.
    - Output: Nenhum.
    - Pré-condição: Lista de terminais e roteador não nulos.
    - Pós-condição: Lista com roteador inserido.

- Função FrequenciaOperadora: Imprime quantos roteadores são de uma determinada operadora.
    - Inputs: Um ponteiro para a lista de roteadores, uma operadora e um ponteiro do tipo FILE.
    - Output: Nenhum.
    - Pré-condição: Lista de roteadores não nula.
    - Pós-condição: Quantidade de roteadores de uma operadora impresso.

- Função ConectaRoteador: Conecta um roteador a um outro roteador.
    - Inputs: Um ponteiro para a lista de roteadores, um ponteiro para o roteador1 e um para o roteador2.
    - Output: Nenhum.
    - Pré-condição: Lista de roteadores, roteador1 e roteador2 não nulos.
    - Pós-condição: Roteador1 conectado a roteador2 e vice-versa.

- Função RemoveRoteador: Remove um roteador e todas as suas conexões da lista de roteadores.
    - Inputs: Um ponteiro para a lista de roteadores e um nome e um ponteiro do tipo FILE.
    - Output: Nenhum.
    - Pré-condição: Lista de terminais não nula e nome não nulo.

- Pós-condição: Lista atualizada sem o roteador e suas conexões.

- Função ImprimeListaRoteadores: Imprime os dados de todos os roteadores da lista.
  - Inputs: um ponteiro para a lista de roteadores e um ponteiro do tipo FILE.
  - Output: nenhum.
  - Pré-condição: lista de roteadores não nula.
  - Pós-condição: dados dos roteadores da lista impressos.

- Função LiberaListaRoteadores: Libera a lista de roteadores especificada.
  - Inputs: um ponteiro para a lista de roteadores.
  - Output: nenhum.
  - Pré-condição: lista de roteadores não nula.
  - Pós-condição: toda a lista liberada.

2. Terminal.c:
   - Contém 3 structs: terminal, celulaT, listaT. Na struct terminal possui o nome, o local e um ponteiro que irá apontar para o roteador que o terminal estará concetado. Já a celulaT define a célula que contemplará a última struct, que é a listaT, sendo a sentinela para a lista simplesmente encadeada de terminais. Na celulaT possui um ponteiro para um terminal e um ponteiro para uma celulaT com nome prox.

   - Função EnviaPacoteDeDados: Verifica se eh possivel enviar dados entre dois terminas.
     - Inputs: um ponteiro para a lista de terminais, um ponteiro para uma lista de roteadores dois nomes e dois ponteiros do tipo FILE.
     - Output: Nenhum.

- Pré-condição: lista e nomes não nulos.
- Pós-condição: Mensagem se pode ou não pode enviar dados de um terminal para o outro.

- Função IniciaTerminal: Inicializa um terminal.
  - Inputs: Nome e o local.
  - Outputs: Um ponteiro para o terminal criado.
  - Pré-condição: Nome e local válidos.
  - Pós-condição: Um terminal inicializado com nome e local definidos pelo input e o campo roteadorConectado apontando para NULL.

- Função IniciaCelulaTerminal: Inicializa a célula de um terminal.
  - Inputs: Um terminal específico.
  - Outputs: Um ponteiro para célula criada.
  - Pré-condição: Terminal inicializado e não-nulo.
  - Pós-condição: Célula existe, o campo terminal e aponta para o terminal de entrada e o campo prox aponta para NULL.

- Função IniciaListaTerminais: Inicializa a sentinela da lista de terminais.
  - Inputs: Nenhum.
  - Outputs: Sentinela Inicializada.
  - Pré-condição: Nenhuma.
  - Pós-condição: Sentinela da lista de retorno existe e os campos primeiro e último apontam para NULL.

- Função CadastraTerminal: Cadastra um terminal a uma lista de terminais.
  - Inputs: Um ponteiro para a lista e um para o terminal.
  - Ouputs: Nenhum.
  - Pré-condição: Lista e terminal não nulos.
  - Pós-condição: Lista atualizada com terminal insereido.

- Função RetiraTerminal: Retira um terminal da lista de terminais.
    - Inputs: Um ponteiro para a lista, um nome e um ponteiro do tipo File.
    - Outputs: Nenhum.
    - Pré-condição: Lista não nula e nome válido.
    - Pós-condição: Terminal especificado fora da lista.

- Função ConectaTerminalAoRoteador: Conecta um terminal a um roteador.
    - Inputs: Um ponteiro para uma lista de terminais, um ponteiro para uma lista de roteadores, dois nomes e um ponteiro do tipo FILE.
    - Outputs: Nenhum.
    - Pré-condição: Lista terminal e lista roteador não nulos.
    - Pós-condição: Terminal especificado conectado ao roteador especificado.

- Função FrequenciaTerminal: Indica a quantidade de terminais de um determinado local.
    - Inputs: Um ponteiro para lista de terminais, um para o local e um ponteiro do tipo FILE.
    - Outputs: Nenhum.
    - Pré-condição: Lista não nula e local válido.
    - Pós-condição: Impresso a quantidade de terminais de um local.

- Função DesconectaTerminal: Desconecta o terminal do roteador.
    - Um ponteiro para uma lista de terminais, um nome e um ponteiro do tipo FILE.
    - Outputs: Nenhum.
    - Pré-condição: Lista de terminais e nome não nulos.
    - Pós-condição: Terminal desconectado.

- Função ImprimiListaTerminal: Imprime todo a lista de terminais especificadas.
    - Inputs: Um ponteiro para a lista de terminais e um ponteiro do tipo FILE.
    - Outputs: Nenhum.
    - Pré-condição: Lista não nula.
    - Pós-condição: Toda a lista impressa.

- Função LiberaListaTerminais: Libera lista de terminais especificada.
    - Inputs: Um ponteiro para a lista de terminais.
    - Outputs: Toda a lista liberada.
    - Pré-condição: Lista de terminais não-nula.
    - Pós-condição: Toda a lista e seus itens não alocados.

- Função ImprimiNetMap: Imprime a situação atual do NETMAP.
    - Inputs: Um ponteiro para a lista de terminais e um para a lista de roteadores e um ponteiro do tipo FILE.
    - Outputs: Nenhum.
    - Pré-condições: Lista de terminais e roteadores não nula
    - Pós-condições: NETMAP impresso como especificado.

3. Conclusão

O trabalho serviu para mostrar possibilidades de uso de estrutura de dados como simulação da vida real e para ensinar melhor o uso de listas encadeadas e o uso de sentinelas para facilitar na manipulação das listas. O Simulador NETMAP, apesar de não ser tão complexo por não ter tempo suficiente para colocar mais condições e entregar o trabalho no prazo, apresentou um caso que se assemelha á vida real e demonstra a troca de informações entre computadores e aparelhos do tipo.

A principal dificuldade da dupla no trabalho foi na hora da modularização, onde começamos por uma modularização que deixava o nosso trabalho muito complexo na hora de conversar entre os .c em função das estruturas serem do tipo opaco e, no nosso caso, precisava chamar elas várias vezes em um .c que não era onde ela estava declarada. Devido a essa dificuldade tivemos até que recomeçar o trabalho do zero após conversar com a professora e definir uma melhor modularização.

4. Referência bibliográficas

- CELES, Cerqueira e Rangel, Introdução a Estrutura de Dados, Editora Elsevier, 2004.