Jonathas Kerber

Desafío técnico para nivelamento C++

DICAS:

- O desafio avalia vários níveis.
- Execute o que conseguires.
- Entregue o que achares pertinente. Algo não funcionando, pode ser pertinente.

Desafio Técnico

- Este desafio consiste em desenvolver um software utilizando a linguagem C++ para ambientes Linux.
- O intuito do software é receber dados provenientes da rede e salvar estes dados em arquivos localmente, ou seja, será um servidor que receberá dados via stream (socket) e salvará os dados em arquivos de até X bytes.

Premissas:

1. **DONE** A porta que o software receberá os dados deverá ser configurável por um arquivo de configuração;
2. **DONE** O tamanho dos arquivos salvos também deverá ser configurável;
3. **DONE** Comunicação TCP;

Observações: Indicamos o uso da Boost como biblioteca utilitária.

O que você deve nos entregar após a conclusão do desafio:

1. Código-fonte de tudo que foi desenvolvido, esteja o software funcionando ou não;
2. Makefile, CMakeLists ou linha de compilação;
3. Informações sobre quais bibliotecas e versões foram utilizadas;
4. Qual o tempo aproximado gasto no desenvolvimento do software;
5. E por último, não menos importante, quais as dificuldades que você enfrentou no desafio.

PREMISSAS E REQUISITOS

Básicas

1. **DONE** A porta que o software receberá os dados deverá ser configurável por um arquivo de configuração;
2. **DONE** O tamanho dos arquivos salvos também deverá ser configurável;
3. **DONE** Comunicação TCP;
4. **DONE** TODOS os dados transmitidos devem ser armazenados corretamente;
5. **DONE** Os arquivos não podem exceder o tamanho máximo em qualquer hipótese;

Intermediárias

1. **DONE** Caso haja a necessidade de armazenar os dados em mais de um arquivo, o arquivo imediatamente anterior deve ter exatamente o tamanho máximo configurado, ou seja, não pode ser menor do que o limite máximo definido;
2. **DONE** O nome do arquivo deve ser configurável, como um prefixo, ao qual deve ser anexado uma marca de tempo do momento da abertura do arquivo, e.g.:

- nome_arquivo = PREFIXO
- Arquivos gerados:
- PREFIXO_20180730145530

1. **TODO** O servidor deve ativar um timer configurável e, caso o cliente não transmita dados por um tempo igual ou maior a este período, cancelar a conexão;

Avançadas

1. **TODO** Permitir a conexão simultânea de vários clientes;
2. **TODO** Garantir que os dados de cada conexão sejam armazenados em arquivos separados, cada qual com sua própria sequência.

Observações: Se faz necessário o uso da Boost como biblioteca para desenvolvimento das comunicações em rede.

ENTREGÁVEIS

- Código-fonte de tudo que foi desenvolvido, esteja o software funcionando ou não;
- Makefile, CMakeLists ou linha de compilação;
- Informações sobre quais bibliotecas e versões foram utilizadas;
- Qual o tempo aproximado gasto no desenvolvimento do software;
- E por último, não menos importante, quais as dificuldades que você enfrentou no desafio.

Instalação e configuração do ambiente

- **DONE** Download e setup de uma máquina virtual Linux Mint 19.2 - Virtual Box
    - link para download: https://www.osboxes.org/linux-mint/
- **DONE** Download, build e instalação da biblioteca Boost V1.71.0
    - link para download: https://www.boost.org/users/history/version_1_71_0.html
    - tutorial para build e install: https://www.boost.org/doc/libs/1_66_0/more/getting_started/unix-variants.html
- **DONE** Download e instalação do editor de texto Atom.
    - link para download: https://www.atom.io

Considerações para implementação

- Para permitir múltiplos clientes conectados e garantir que os dados de cada conexão sejam armazenados separadamente, optei por alocar um diretório específico para armazenar os dados de cada conexão. O nome do diretório segue padrão: cnx_YYYYMMddHHmmss

- Caso haja mais que uma conexão simultaneamente no mesmo instante, o nome do diretório será adicionado do sufixo _n: cnx_YYYYMMddHHmmss_n

- **DONE** Caso o volume de dados enviado pelo cliente seja grande comparado ao tamanho máximo de arquivo definido, pode acontecer de que seja necessário criar mais de um arquivo para salvar os dados no mesmo instante (mesma YYYYMMddHHmmss). Nesse caso o nome do próximo arquivo será adicionado o sufixo _n: prefixo_YYYYMMddHHmmss_n

Análise e modelagem do sistema

- Arquivo de configuração deve conter:
    - Porta de abertura do socket
    - Tamanho exato de cada arquivo
    - Nome do arquivo
    - Timeout de recepção
- O objeto Servidor deverá instanciar um objeto conexão para cada cliente conectado.

- Os parâmetros: timeout, save_path, file_name e file_size deverão ser informados à conexão.

- A cada nova conexão instanciada, deve ser também instanciado um timer para contagem do tempo de timeout da conexão. O timer deve ser atualizado a cada dado recebido.c

Estudo dos recursos disponíveis

Na página: https://www.boost.org/doc/libs/1_71_0/ está disponível a documentação de todos os recursos fornecidos pela biblioteca Boost.

- A mesma fornece um módulo denominado "ASIO" que disponibiliza de funcionalidades para comunicação utilizando sockets, bem como timers. **Funcionalidades** essas definidas como **necessárias** para o projeto.

- As estruturas: **acceptor e socke**t devem ser utilizadas em conjunto para permitir a implementação de um socket no modo TCP Server.

- O tutorial 3 (https://www.boost.org/doc/libs/1_71_0/doc/html/boost_asio/tutorial/tutdaytime3.html) forcene um exemplo de como utilizar a boost para a criação de um socket TCP assincrono.

- A estrutura: **steady_timer** pode ser utilizada para definir um timer assincrono para contagem do tempo de timeout da conexão caso não haja transferencia de dados.

- O tutorial 4 (https://www.boost.org/doc/libs/1_71_0/doc/html/boost_asio/tutorial/tuttimer4.html) fornece um exemplo de como utilizar um timer utilizando o bind de um handler para tratamento do timeout.

PREMISSAS E REQUISITOS

Básicas

6. **DONE** A porta que o software receberá os dados deverá ser configurável por um arquivo de configuração;
7. **DONE** O tamanho dos arquivos salvos também deverá ser configurável;
8. **DONE** Comunicação TCP;
9. **DONE** TODOS os dados transmitidos devem ser armazenados corretamente;
10. **DONE** Os arquivos não podem exceder o tamanho máximo em qualquer hipótese;

Intermediárias

3. **DONE** Caso haja a necessidade de armazenar os dados em mais de um arquivo, o arquivo imediatamente anterior deve ter exatamente o tamanho máximo configurado, ou seja, não pode ser menor do que o limite máximo definido;
4. **DONE** O nome do arquivo deve ser configurável, como um prefixo, ao qual deve ser anexado uma marca de tempo do momento da abertura do arquivo, e.g.:

- nome_arquivo = PREFIXO
- Arquivos gerados:
- PREFIXO_20180730145530

2. **TODO** O servidor deve ativar um timer configurável e, caso o cliente não transmita dados por um tempo igual ou maior a este período, cancelar a conexão;

Avançadas

3. **TODO** Permitir a conexão simultânea de vários clientes;
4. **TODO** Garantir que os dados de cada conexão sejam armazenados em arquivos separados, cada qual com sua própria sequência

PROPOSTA DE SOLUÇÃO

ESTRATÉGIA ADOTADA

- Manipulando arquivos em C++:
  - Abertura: configurações e dados **DONE**
  - Leitura: configurações e dados **DONE**
  - Escrita: dados **DONE**
  - Escrita condicional: dados **DONE**
  - Escrita condicional+: dados **DONE**
- Comunicação TCP:
  - Socket server **DONE**
  - Socket client **DONE**
  - Server assíncrono **WIP**
- Gerenciamento de conexões
  - Timeout entre sender e receiver **TODO**
  - Múltiplas conexões **TODO**

DIFICULDADES

- Configuração do ambiente de desenvolvimento:
    - Problemas de compatibilidade do driver de vídeo do Linux Mint 19.2 com o Virtual Box 6.1. Bug conhecido pela Oracle (Links: [1], [2]).
    - Familiarização com build das bibliotecas do pacote Boost (v1.71.0). (OBS: as bibliotecas são salvas em “/usr/include”.)
    - Local onde estão salvos os headers:
        - /usr/local/include/boost_1_71_0/boost/regex
        - /usr/include/boost
    - Local onde estão salvas as bibliotecas compiladas:
        - /usr/include/lib
    - Bibliotecas compiladas utilizadas
        - filesystem
- Familiarização com a linguagem C++ e com o paradigma de OOP;

ENTREGÁVEIS

- Código-fonte de tudo que foi desenvolvido, esteja o software funcionando ou não;
- Makefile, CMakeLists ou linha de compilação;
- Informações sobre quais bibliotecas e versões foram utilizadas;
- Qual o tempo aproximado gasto no desenvolvimento do software;
- E por último, não menos importante, quais as dificuldades que você enfrentou no desafio.