Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação em Sistemas Computacionais

Semestre de Inverno de 2017/2018

Série de Exercícios 3 - Trabalho de Grupo

Construa os programas e bibliotecas indicados na Parte II, usando a linguagem C, tendo o cuidado de eliminar repetições no código fonte e de isolar funcionalidades distintas em diferentes ficheiros fonte. Entregue o código desenvolvido, devidamente indentado e comentado, bem como o `Makefile` para gerar os executáveis e bibliotecas a partir do código fonte. Assegure-se de que o compilador não emite qualquer aviso sobre o seu código com a opção **`-Wall`** activa, e de que no final da execução do programa não existem recursos por libertar (memória alocada dinamicamente e ficheiros abertos).

Apresente um relatório com a descrição sucinta das soluções, que deverá ser um guia para a compreensão do código desenvolvido e não a tradução do código para língua natural. Contacte o docente sempre que tiver dúvidas.

Encoraja-se a discussão dos problemas e das respectivas soluções com colegas de outros grupos (tenha em consideração que a partilha directa de soluções implica, no mínimo, a anulação das entregas dos grupos envolvidos).

O produto final desta série de exercícios é uma biblioteca que implemente serviços de consulta de informação sobre futebol e um programa que demonstre a funcionalidade implementada pela biblioteca. Os dados sobre futebol estão acessíveis no *site* **`football-data.org`** através de uma API *Web*. Toda a informação sobre este serviço pode ser consultada em http://www.football-data.org.

Embora o enunciado esteja organizado em 6 pontos, é feita apenas uma entrega final, com o código fonte da biblioteca (ponto 4), o programa de utilização (ponto 5) e um *Makefile* que define os *targets* **`lib`**, **`app`** e **`clean`**, que têm como finalidade, respectivamente, a geração da biblioteca, a geração do programa para utilização e a eliminação dos artefactos gerados pelos *targets* anteriores.

Parte I - Preparação do ambiente de desenvolvimento

O acesso à informação usa o protocolo HTTP (*Hypertext Transfer Protocol*) para aceder aos serviços definidos por uma API REST (*Representational State Transfer*). É prática comum que os serviços suportados em API REST usem o paradigma pergunta/resposta. A pergunta é representada pelo URL (*Uniform Resource Locator*) que é usado no pedido HTTP GET enviado ao servidor; a resposta ao pedido é codificada no formato JSON (*JavaScript Object Notation*), de acordo com o esquema definido pela API.

A documentação da API a utilizar está disponível em http://www.football-data.org/docs/v1/index.html.

CURL

O acesso ao serviço é feito estabelecendo ligações ao servidor usando o protocolo HTTP. Para suportar as comunicações com o servidor deverá ser utilizada a biblioteca *open source* **`libcurl`**.

Instalação: **`$ sudo apt-get install libcurl4-gnutls-dev`**

Documentação: http://curl.haxx.se/libcurl

JSON

Para interpretar as respostas do servidor em formato JSON deverá ser utilizada a biblioteca *open source* **`jansson`**.

Instalação: **`$ sudo apt-get install libjansson-dev`**

Documentação: https://jansson.readthedocs.io/en/2.7/index.html

Tal como noutras linguagens, a indentação da escrita em formato JSON facilita a leitura direta por parte do humano. Sugere-se a utilização de um visualizador JSON, que pode ser instalado no *browser* na forma de *plug-in*.

Valgrind

Para verificar se um programa liberta toda memória que reservou dinamicamente, deve utilizar-se a ferramenta **`valgrind`**.

Instalação: **`$ sudo apt-get install valgrind`**

Documentação: **`$ man valgrind`**

Parte II - Realização

1. Recorrendo à biblioteca **`libcurl`**, implemente a função **`http_get`** que realiza um pedido HTTP GET ao URL especificado através do parâmetro **`url`** e armazena o resultado no ficheiro cujo nome é especificado através do parâmetro **`filename.`** A função devolve um valor, que interpretado como booleano, indica se houve sucesso.

   ```
   int http_get(const char *url, const char *filename);
   ```

   Escreva um programa de teste que, recebendo como argumentos um URL e o nome de um ficheiro, permite verificar o correto funcionamento desta função, descarregando o conteúdo do recurso para o ficheiro. Teste o programa usando http://imagem.band.com.br/f_198156.jpg como URL.

2. Utilizando as bibliotecas **`libcurl`** e **`jansson`**, implemente a função **`http_get_json_data`**, que realiza um pedido HTTP GET ao URL especificado através do parâmetro **`url`**, que deve corresponder a um recurso HTTP do tipo **`application/json`** e retorna o ponteiro para uma instância do tipo **`json_t`** (definido pela biblioteca **`jansson`**) com o conteúdo da resposta. Se ocorrer algum erro durante a transferência, a função retorna **`NULL`** e escreve em **`stderr`** a mensagem que indica a razão do erro.
   Não deve ser usado um ficheiro temporário, isto é, a resposta ao pedido HTTP GET deve ser mantida em memória.

   ```
   json_t *http_get_json_data(const char *url);
   ```

   Para teste pode usar um programa que utilize esta função para aceder aos dados sobre as competições neste momento a decorrer, cujo URL é http://football-data.org/v1/competitions/, e afixe na consola a informação relevante.

3. Utilizando a função do ponto anterior, implemente as funções **`get_competitions`**, **`get_teams`** e **`get_fixtures`**.

   A função **`get_competitions`** retorna um *array* de elementos de informação sobre competições (**`Competition *`**) e devolve a sua dimensão através do parâmetro **`size`**. Esta informação é obtida diretamente em **`http://football-data.org/v1/competitions/?season={year}`**, sendo **`{year}`** o ano que define a época de competições.

   A função **`get_teams`** retorna um *array* cujos elementos contêm informação sobre as equipas (**`Team *`**) que participam numa dada competição e devolve a sua dimensão através do parâmetro **`size`**. Esta informação é obtida em **`http://football-data.org//v1/competitions/{id}/teams/`**, sendo **`{id}`** o identificador numérico da competição.

   A função **`get_fixtures`** retorna um *array* cujos elementos contêm informação sobre os jogos (**`Fixture *`**) de uma dada competição e devolve a sua dimensão através do parâmetro **`size`**. Esta informação é obtida em **`http://football-data.org/v1/competitions/{id}/fixtures/`**, sendo **`{id}`** o identificador numérico da competição.

   ```
   typedef struct competition { /* a definir */ } Competition;
   typedef struct team { /* a definir */ } Team;
   typedef struct fixture { /* a definir */ } Fixture;

   Competition *get_competitions(int season, size_t *size);
   ```

```
Team *get_teams(int competition_id, size_t *size);

Fixture *get_fixtures(int competition_id, size_t *size);

Fixture *get_team_fixtures(int team_id, size_t size);
```

Na programação destas funções deve sempre libertar a memória alocada dinamicamente no momento em que esta deixe de ser utilizada.

Faça um programa para teste das funções implementadas que mostre na consola a lista das competições, a lista de equipas e os jogos para uma dada competição.

4. Construa uma biblioteca de ligação dinâmica (*shared object*) com as funções desenvolvidas nos pontos anteriores e com as funções auxiliares que entender necessárias. Na organização do código, tenha em consideração que deve evitar repetições de código fonte.

5. Desenvolva um programa que permanece em execução aceitando e processando comandos para apresentação na consola, de informação legível na forma de quadros.

   O programa recebe a época de competição na linha de comando. Em caso de omissão considera a época corrente.

   O programa deve aceitar são os seguintes comandos:

c	Lista as competições.
t {competition_id}	Lista as equipas participantes na competição especificada.
x {competition_id}	Lista os jogos da competição especificada.
j {team_id}	Lista os jogos de uma dada equipa em todas as competições em que está envolvida.
q	Termina a execução do programa.

   Na construção do programa deve utilizar a biblioteca produzida no ponto anterior, sendo essa a única via de acesso à informação.

   Na execução de comandos posteriores deve reutilizar a informação adquirida em comandos anteriores, minimizando assim o número de pedidos ao servidor. Para retenção da informação deve criar em memória uma estrutura de dados adequada.

   Ao terminar a execução, o programa deve explicitamente libertar a memória alocada dinamicamente.

Data limite de entrega: 2 de Janeiro de 2018

ISEL, 22 de Novembro de 2017

Carlos Martins, Ezequiel Conde, João Patriarca

Referências

HTTP http://www.w3.org/Protocols/rfc2616/rfc2616.html
JSON http://www.json.org/