Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação em Sistemas Computacionais

Semestre de Verão 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 na plataforma *Thoth* e um programa utilitário que, utilizando a biblioteca realizada, produza um arquivo ZIP com os enunciados dos trabalhos práticos.

A plataforma *Thoth* está acessível no endereço **`https://adeetc.thothapp.com`** e dispõe de uma API *Web*, documentada em https://adeetc.thothapp.com/api/doc.

Embora o enunciado esteja organizado em 4 pontos, é feita apenas uma entrega final, com o código fonte da biblioteca (ponto 3), o programa de utilização (ponto 4) 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 outros *targets*.

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 https://adeetc.thothapp.com/api/doc.

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*.

ZIP

Para criar arquivos em formato ZIP poderá ser utilizada a biblioteca *open source* **`libzip`**.

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

Documentação: https://libzip.org/

Valgrind

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

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

Documentação: **`$ man valgrind`**

Parte II - Realização

1. Utilizando a biblioteca **`libcurl`**, implemente a função **`http_get_data`**, que realiza um pedido HTTP GET ao URL especificado através do parâmetro **`url`**, e retorna o ponteiro para um *buffer* em memória (alocada dinamicamente pela função) com o conteúdo da resposta e a respectiva dimensão através do parâmetro **`data_size`**. Se ocorrer algum erro durante a transferência, a função retorna **`NULL`** e escreve em **`stderr`** a mensagem que indica a razão do erro.

   ```
   char *http_get_data(const char *url, size_t *data_size);
   ```

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

2. Utilizando a função do ponto anterior, implemente as funções **`get_class_id`** e **`get_class_workitems`**.

   A função **`get_class_id`** retorna o identificador usado na API do *Thoth* para identificar a turma. A turma é especificada com a mnemónica da unidade curricular (e.g., PSC), o semestre lectivo (e.g., 1314v) e pela designação da turma usada nos serviços académicos do ISEL (e.g., LI31D).

   A função **`get_class_workitems`** retorna um *array* cujos elementos são instâncias do tipo `WorkItem` que contêm informação sobre os trabalhos de uma turma e devolve a respectiva dimensão através do parâmetro **`workitems_size`**.

   ```
   typedef struct workitem {
     int workitem_id;
     const char *doc_name;
     const char *attach_name;
   } Workitem;

   int get_class_id(char *ucShortName, char *lectiveSemester, char *className);

   Workitem *get_class_workitems(int class_id, size_t *workitems_size);
   ```

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

   Faça um programa para teste das funções implementadas que mostre na consola a lista dos enunciados de uma dada turma.

   Para obter informação do *Thoth*, utilize a respectiva API, considerando os seguintes exemplos de recursos:

   Lista geral de turmas: **`https://adeetc.thothapp.com/api/v1/classes`**

   Trabalhos de uma turma: **`https://adeetc.thothapp.com/api/v1/classes/<class_id>/workitems`**

   Detalhes de um trabalho: **`https://adeetc.thothapp.com/api/v1/workitems/<workitem_id>`**

3. 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.

4. Desenvolva um programa que, utilizando a biblioteca construída no ponto anterior, obtenha do *Thoth* os enunciados dos trabalhos práticos e respetivos anexos publicados numa turma.

   O programa recebe 3 argumentos na linha de comando: a sigla da unidade curricular (e.g., PSC) e a designação do semestre (e.g., 1314v) e da turma (e.g., LI31D).

   Como resultado, será gerado um ficheiro ZIP contendo os enunciados dos trabalhos práticos e os anexos.

   Exemplo de invocação do programa:

   ```
   $ thoth_get_work PSC 1314v LI31D
   ```

Os *links* para os enunciados e anexos, externos à API, são compostos como os seguintes:

Enunciado: **`https://adeetc.thothapp.com/classes/PSC/1314v/LI31D/workitems/<workitem_id>/document`**

Anexo: **`https://adeetc.thothapp.com/classes/PSC/1314v/LI31D/workitems/<workitem_id>/attachment`**

Data limite de entrega: 17 de Junho de 2018

ISEL,19 de Maio de 2018

Carlos Martins, Ezequiel Conde

Referências

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