Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Inverno de 2022/2023

Série de Exercícios 3

Construa os programas e as 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 os ***`makefile`*** para gerar os executáveis e as 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). Deve verificar essa situação com o utilitário Valgrind. Em caso de erro irrecuperável, o programa não deve terminar descontroladamente, deve, no mínimo, emitir uma mensagem informativa e em seguida terminar. O código desenvolvido será valorizado segundo os seguintes critérios, em importância decrescente: correção, eficiência, clareza.

O produto final desta série de exercícios é uma biblioteca que implementa serviços de consulta de informação sobre produtos e utilizadores numa loja, um programa que utiliza essa biblioteca e um *shared object* que acrescenta uma nova funcionalidade ao programa, na forma de *plug-in*. A informação dos dados da loja é obtida no site dummy JSON utilizando uma API Web.

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://dummyjson.com/docs.

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: libcurl - the multiprotocol file transfer library.

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: Jansson Documentation

Valgrind

Para verificar se um programa liberta toda a memória alocada dinamicamente deverá utilizar a ferramenta **`valgrind`**.

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

Documentação: **`$ man valgrind`**

Parte II - Realização

1. 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`**. Retorna o ponteiro para uma instância do tipo **`json_t`** (definido no âmbito da biblioteca **`jansson`**) com o conteúdo da resposta. Se ocorrer um erro durante a transferência, a função retorna **`NULL`** e mostra em **`stderr`** a mensagem que indica a razão do erro.

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

   Para teste realize um programa que aceda à informação do produto 1, cujo URL é https://dummyjson.com/products/1 e afixe na consola alguma da informação obtida.

2. Utilizando a função do ponto anterior, implemente as funções **`products_get`** , **`users_get`** e **`carts_get`**.

   Na elaboração destas funções, utilize os seguintes tipos de dados para representar, respetivamente, um produto; um utilizador; um carrinho de compras:

   ```
   typedef struct {
      int id;
      float price;
      const char *description;
      const char *category;
   } Product;

   typedef struct {
      int id;
      const char *name;
   } User;

   typedef struct {
      int user_id;
      size_t n_products;
      struct {
         int id;
         size_t quantity
      } products[];
   } Cart;
   ```

   A função **`products_get`** retorna a informação sobre todos os produtos. Esta informação é obtida em https://dummyjson.com/products.

   ```
   Products *products_get();
   ```

   A função **`users_get`** retorna informação sobre todos os utilizadores. Esta informação é obtida em https://dummyjson.com/users.

   ```
   Users *user_get();
   ```

   A função **`carts_get`** retorna informação sobre todos os carrinhos de compras. Esta informação é obtida em https://dummyjson.com/carts.

   ```
   Carts *carts_get();
   ```

   Deve definir os tipos **`Products`**, **`Users`** e **`Carts`** de modo a representarem a informação sobre todos os elementos do respetivo conjunto.

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

Realize um programa de teste que crie um ficheiro para cada um dos conjuntos em formato **csv**.

3. Construa uma biblioteca de ligação dinâmica (*shared object*) com as funções definidas 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.

   Utilize o programa de teste anterior para verificar o correto funcionamento da biblioteca.

4. Desenvolva um programa que permanece em execução, aceitando e processando comandos para apresentação de informação na consola.

   O programa deve aceitar os seguintes comandos:

   u Listar utilizadores, ordenados por ordem alfabética dos nomes.

   c Listar carrinhos, com indicação do utilizador e ordenados por ordem decrescente da importância das compras.

   s Termina a execução do programa.

   Na construção do programa deve-se utilizar a biblioteca produzida no ponto anterior.

   Na execução de comandos posteriores deve-se reutilizar a informação adquirida em comandos anteriores, minimizando assim o número de pedidos remotos. Para retenção da informação deve criar em memória uma estrutura de dados adequada. Na construção dessa estrutura de dados estabeleça referências com ponteiros entre as entidades, de modo a suprimir ou minimizar as procuras exaustivas. Por exemplo, a variável que representar um carrinho deve ter um ponteiro para a variável que representar o respetivo utilizador.

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

5. Modifique o programa anterior de modo que possam ser acrescentados novos comandos na forma de *plug-in*.

   Exemplifique a utilização dessa funcionalidade acrescentando um novo comando que lista na consola as categorias de produtos, ordenadas da categoria mais solicitada para a menos solicitada.

Data limite de entrega: 8 de janeiro de 2023

ISEL, 28 de novembro de 2022