get_file.c

Programa de teste para a função *http_get_data* da biblioteca *libthoth.so*.

Faz um simples pedido HTTP GET ao link especificado no primeiro argumento de execução. Escreve a resposta do pedido para um ficheiro com o nome especificado no segundo argumento de execução.

get_class_assignment.c

Programa de teste para a função *get_class_id* e *get_class_workitems* da biblioteca *libthoth.so*.

Começa por fazer uma chamada a função *get_class_id* passando os três argumentos de execução. Esta função retorna o ID da turma correspondente aos argumentos passados a função.

Faz uma chamada a função *get_class_workitems* passando o ID obtido pela função anterior, esta função retorna um array de *Workitems* contendo a informação pedida sobre cada trabalho da turma correspondente ao ID.

Finalmente, escreve para a consola o nome de todos os trabalhos obtidos.

thoth_get_work.c

Programa que cria um zip na diretoria onde o programa é chamado contendo todos os trabalhos da turma especificada pelos parâmetros de execução.

Começa por obter o ID da turma correspondente aos argumentos de execução, semelhante ao programa descrito anteriormente. Com este ID, obtem todos os trabalhos da turma.

Prepara as variáveis para a escrita do ficheiro .zip. Este ficheiro terá o nome dos argumentos de execução e será escrito na diretoria de execução do programa.

Prepara um array de strings que irá conter o nome dos ficheiros necessários serem apagados no final da execução do programa.

Declara todas as variáveis necessárias para a iteração dos trabalhos da turma.

Começa por iterar cada um dos trabalhos obtidos no inicio do programa. Por cada iteração irá:

-Preparar a string do URL do enunciado. A string do URL do anexo. A string do nome do ficheiro do enunciado (.pdf). A string do nome do ficheiro do anexo (.zip).

-Fazer duas chamadas a função *http_get_data.* A primeira chamada obter o ponteiro que aponto para o espaço em memoria que contem os dados do pedido GET ao URL do enunciado. A segunda chamada obter o mesmo mas para o URL do anexo.

-Abrir um FILE stream. Nesta stream escrever os dados do enunciado apontados pelo ponteiro obtido na linha anterior. Fechar a stream após a escrita. Abrir um novo FILE stream para escrever os dados do anexo apontados pelo outro ponteiro. Fechar a segunda stream. Estes dois ficheiros terão o nome preparado pelas strings no inicio da iteração e encontram-se na diretoria que o programa foi chamado.

-Utilizar a biblioteca *libzip* para escrever um ficheiro .zip os dois ficheiros criados na linha anterior (enunciado e anexo). Este zip é o mesmo para todas as iterações.

-Adicionar ao array de strings o nome dos ficheiros escritos.

Após ter iterado por todos os trabalhos. O zip é fechado.

Finalmente, apaga todos os ficheiros com o nome dos ficheiros adicionados ao zip, deixando assim apenas o zip na diretoria onde o programa é chamado.

thoth.h

Ficheiro header que contem a declaração de todas as funções de *thoth.c*.

thoth.c

size_t WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp)

Função auxiliar para a função *curl_easy_setopt* da biblioteca *libcurl*. A função é chamada quando o pedido GET é realizado. A função irá escrever para a struct *MemoryStruct* a resposta de dados do pedido GET e também calcular o tamanho do pedido. Esta função vem dos exemplos presentes em https://curl.haxx.se/libcurl/c/example.html.

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

Função pedida para o Ex1 do enunciado.

A função utiliza varias funções da biblioteca *libcurl***.** Estas funções agilizam o processo do pedido GET.

Utilizando a função *curl_easy_setopt* , especific: o URL para qual realizar o pedido; A função a chamar quando o pedido é realizado (função descrita acima); A struct para onde escrever a resposta.

Realiza o pedido GET com a função *curl_easy_perform.* De seguida, utiliza a função *curl_easy_cleanup* para libertar a memoria reservada.

A função escreve no parâmetro de saída **data_size* o tamanho dos dados do pedido. Retorna um ponteiro para o inicio dos dados em memoria se tudo correr bem. Se não conseguir realizar o pedido, retorna NULL.

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

Função pedida para o Ex2 do enunciado.

A função começa por fazer um pedido HTTP ao URL de todas as turmas. Este pedido retorna uma resposta no formato JSON.

Utiliza as funções da biblioteca *libjansson* para extrair da resposta a informação desejada. Neste caso queremos o array JSON “classes”. É libertado os dados do pedido HTTP realizado no inicio da função.

Constrói a string com o nome na turma baseado nos argumentos de chamada da função.

Itera sobre todas as turmas obtidas no array JSON “classes” obtido previamente. Por cada turma compara o nome com a string preparada no ponto anterior. Quando encontrar a turma no array JSON com o mesmo nome da turma que estamos a procura, é retornado o seu ID. Caso não encontre nenhuma turma no array JSON com o nome da turma desejado, a função retorna 0.

Workitem *get_class_workitems(int class_id, size_t *workitems_size)

Função pedida para o Ex2 do enunciado.

Esta função é muito semelhante a função descrita acima. A função realiza um pedido HTTP ao URL que contem a resposta JSON para todos os trabalhos de uma turma. Com a resposta JSON, obtém o array JSON “workItems”.

Itera sobre todos os elementos do array JSON, obtendo diversas informações sobre cada elemento. Coloca esta informação na struct Workitem e de seguida coloca este Workitem no array de Workitems.

No final, atualiza o parâmetro de saída *workitems_size* com o numero de iterações realizadas pelo ciclo anterior. Retorna o ponteiro para o array de Workitems.