Instituto **S**uperior de **E**ngenharia de **L**isboa

Licenciatura em Engenharia Informática e de Computadores

Semestre de Inverno 2015/2016

Programação em Sistemas Computacionais

Série 3

Trabalho elaborado por:

Carlos Florêncio Nº 39250

Índice

Introdução

Este é um trabalho que visa a aprendizagem de bibliotecas dinâmicas partilhadas em c e gestão de memória do software.

Para isso a tarefa consistia em consumir a books api da Google com pedidos http via libcurl a partir do nosso programa em c e criar uma biblioteca para que esta possa ser reutilizada por terceiros.

Para fazer parse dos json da api foi utilizada a biblioteca jansson dita no enunciado. Toda a informação dinâmica teve de ser alocada e libertada no final da sua utilização.

A utilização da ferramenta valgrind foi essencial para descobrir memories leaks que pudessem comprometer o nosso software.

Decidi incluir a biblioteca numa pasta à parte do código para estar mais arrumado, tendo deste modo dois makefiles, um na pasta da lib e outro na root do projecto. O makefile da biblioteca tem de ser gerado primeiro é necessário para a execução do makefile principal.

Na root do projecto existe um ficheiro api_key.txt que deve conter uma api valida. Existe um método em utils.c que lê a api do ficheiro.

Biblioteca libhttp

Libhttp.h

Este header tem todos os métodos que a biblioteca gerada irá ter, decidi incluir todos os métodos neste header em vez de separar pelos respectivos headers de cada ficheiro da biblioteca devido à importação ser mais facilitada para quem usa a biblioteca. Apenas tem de fazer um include em vez dos vários.

Contem constantes globais com os uris da api e um macro para facilitar a alocação de memoria.

A struct collection contem dois campos, um inteiro com o total de items e um ponteiro para ponteiros do tipo Volume.

A struct volume contem os dados id, title, selfLink, isbn13. Podia ter adicionado mais campos mas estava curto de tempo.

parser.c

Este ficheiro contem todo o código essencial da api, nomeadamente as funções googleBooksSearchVolumes e googleBooksGetEpubUrl. Assim como tem outros métodos uteis para a sua execução.

Contem os métodos de alocação de coleções e volumes assim como os da sua destruição.

http.c

Neste ficheiro estão as funções de fazer os pedidos http a um determinado uri da api, utilizado o libcurl.

Todas as boas práticas de utilização do libcurl estão aplicadas, fazendo sempre cleanup como diz a documentação do mesmo.

No entanto existem leaks de memória que pela pesquisa que fiz, conclui que era problema do openssl do sistema. Talvez noutro sistema com outra versão do libcurl e openssl os leaks já não aconteçam.

Estas funções usadas no parser.c para consumir a api.

io.c

Este ficheiro contem funções que permitem ler e escrever ficheiros no filesystem.

utils.c

Aqui estão pequenas funções uteis que permitem manipular strings.

Teste da biblioteca libtest.c

Neste ficheiro existem métodos para testar as funções da api e o método main que chama todos os testes e faz print no stdout se todos os testes passaram ou não.

Conclusão

Com esta série de exercícios pude aprender melhor como funciona a gestão dinâmica de memoria do software. Tirei algumas duvidas que tinha sobre a alocação de strings e estruturas em c.

Deparei me com alguns problemas durante a execução do trabalho, tais como segmentation fault ao correr o software. Mas o problema estava na alocação de memoria e de como estava a ser escrita.

Não consegui terminar o ultimo exercício do trabalho a tempo que seria fazer uma pequena app para usar os métodos da biblioteca que foi feita e portanto esta não está incluída.