Trabalho Prático 1
Crawler
Recuperação de Informação

Daniel Ferreira Abadi[1]
2018088062

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

1. Introdução

O problema abordado consiste em construir um pequeno exemplo de um Crawler simples em C++ utilizando a biblioteca CkSiper da Chilkat. O objetivo do mesmo é fazer com que aprendamos a utilizar a nova biblioteca e nos acostumemos com o ambiente de compilação do mesmo, que é diferente do habitual.

Nosso programa deve receber 2 parâmetros: uma URL inicial, de onde devemos começar, e um número inteiro, o número de links adicionais que devemos buscar. Para cada página devemos imprimir na tela sua URL e seu título, e ao final do processo devemos imprimir os segundos gastos em média por cada página.

2. Implementação

Nos foi dado um código de exemplo que nos serviria como um ponto de partida, o mesmo apenas buscava as páginas e imprimia suas respectivas URLs. Para que o programa atendesse aos requisito dado, foi necessário mudar o código para que recebesse a URL inicial e o número de páginas, isto foi feito por meio de argumentos em linha de comando. Já para imprimir o título de cada página foi utilizada a função lastHtmlTitle da biblioteca apresentada.

E, por fim, para imprimir o tempo médio por cada página em segundos foi necessário utilizar uma biblioteca adicional chamada Chrono. Foram criadas duas variáveis que guardavam o tempo antes e depois do bloco de código que buscava as páginas, e uma terceira para fazer a conversão de tempo e receber a diferença das duas primeiras. Foi decido manter o comando SleepMs(1000), porém o tempo foi descontado na média.

3. Conclusão

Como foi proposto, testou-se o programa para duas URLs de diferentes domínios com pelo menos 10 novas buscas de links cada. Escolhi o site da globo de notícias, o g1.globo.com, e o site principal da UFMG, ufmg.br, com 10 novas buscas para cada. Como pode-se observar nas figuras abaixo, ambas as buscas tiveram o mesmo tempo médio de 0.272727s para cada link.

Há, entretanto, algumas peculiaridades no programa, quando se passa a URL como www.g1.globo.com não é buscado nenhum outro link após o primeiro. E quando utiliza-se https://g1.globo.com, o primeiro link é buscado duas vezes.

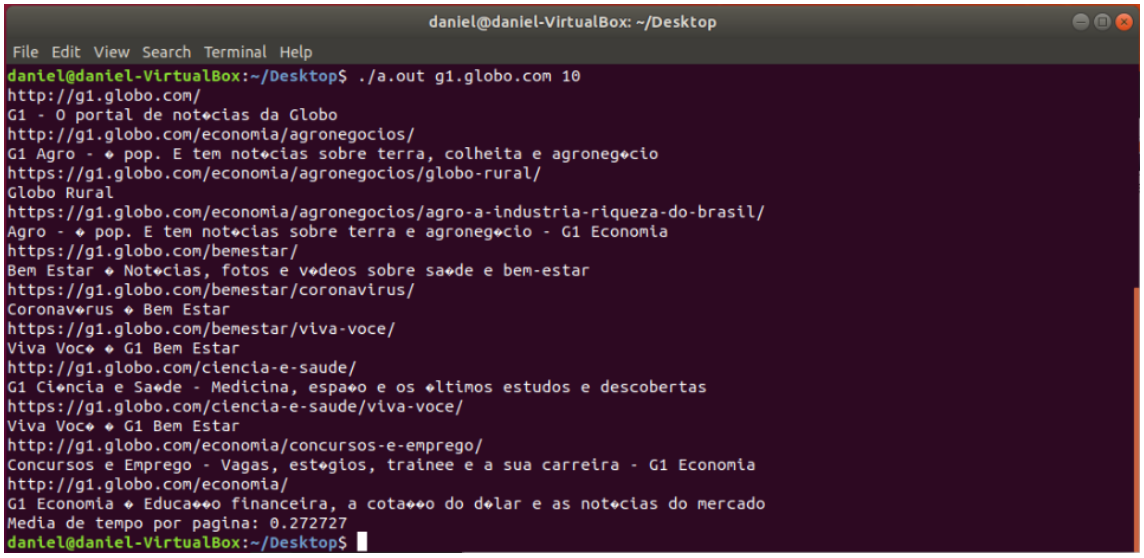

Figura 1. Site g1.globo.com

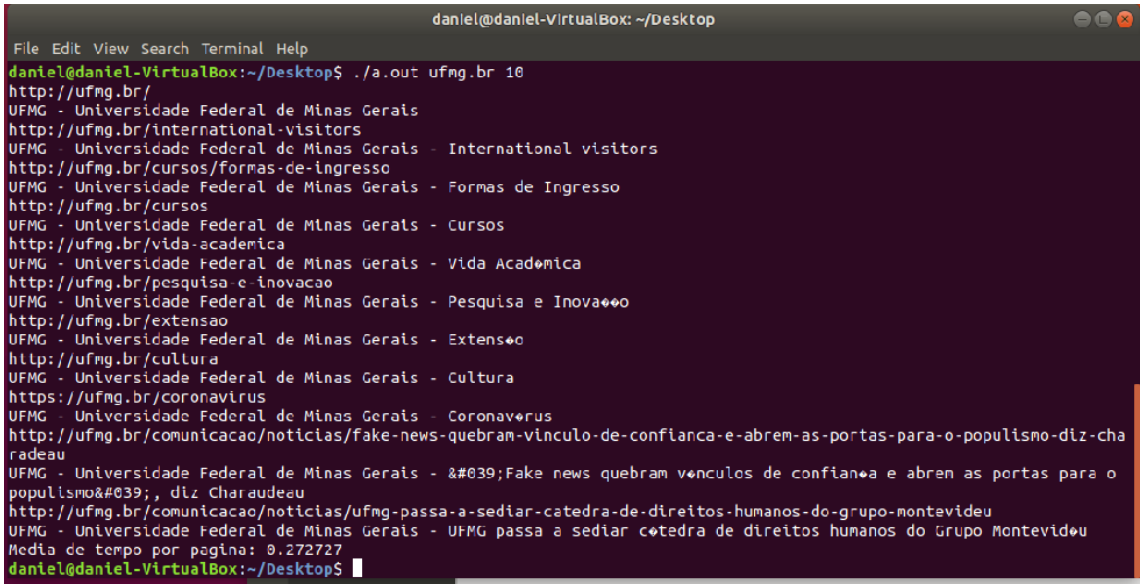

Figura 2. Site ufmg.br

4. Código

```
#include <CkSpider.h>
#include <iostream>
#include <chrono>
int main(int argc, char *argv[]){
   CkSpider spider;
   spider.Initialize(argv[1]);
   // Add the 1st URL:
   spider.AddUnspidered(argv[1]);
   // Begin crawling the site by calling CrawlNext repeatedly.
   auto start = std::chrono::high_resolution_clock::now();
```

```
    // Receive the current time
    int n = std::atoi(argv[2]);
    for (int i = 0; i <= n; i++) {
       bool success;
       success = spider.CrawlNext();
       if (success == true) {
          // Show the URL of the page just spidered.
          std::cout << spider.lastUrl() << "\r\n";
          // Show the title of the page just spidered.
          std::cout << spider.lastHtmlTitle() << "\r\n";
       }
       else {
          // Did we get an error or are there no more URLs to
             crawl?
          if (spider.get_NumUnspidered() == 0) {
             std::cout << "No more URLs to spider" << "\r\n";
          }
          else {
             std::cout << spider.lastErrorText() << "\r\n";
          }
       }
       // Sleep 1 second before spidering the next URL.
       spider.SleepMs(1000);
    }
    auto stop = std::chrono::high_resolution_clock::now();
    // Receive the current time
    auto duration =
       std::chrono::duration_cast<std::chrono::seconds>(stop-start);
    // Time subtraction and conversion to seconds
    float avg = (float)(duration.count()-(n+1)) / (float)(n+1);
    std::cout <<"Media de tempo por pagina: " << avg << "\n";
    return 0;
}
```