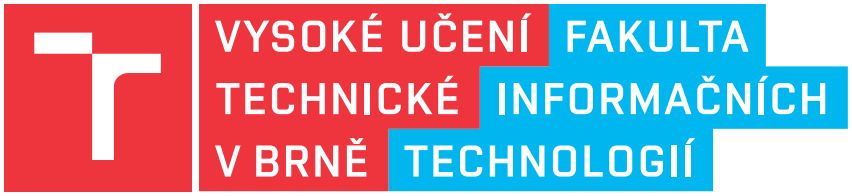

Projektová dokumentace

Čtečka novinek ve formátu Atom a RSS s podporou TLS

Síťové aplikace a správa sítí

17. listopadu 2018 Dominik Harmim (xharmi00)

Obsah

1 Úvod

Cílem projektu bylo vytvořit komunikující aplikaci v jazyce C/C++. Vytvořil jsem program, který stahuje XML soubory (tzv. `feedy`) přes síť Internet a tyto soubory zpracovává a vypisuje v nich uvedné informace.

Program zpracovává XML soubory ve formátu `Atom` a `RSS`, přičemž podporované verze jsou `Atom 1.0` [8], `RSS 1.0` [9] a `RSS 2.0` [6].

Program po spuštění stáhne zdroje a na standardní výstup vypíše informace požadované uživatelem. Vstupní zdroje i výstupní informace je možné nastavovat vstupními argumenty programu, viz 3.

2 Návrh a implementace

Aplikace je rozdělena do několika zdrojových souborů a tříd, které implementují požadované chování. Veškeré zdrojové soubory se nachází v adresáři `./src`. V této kapitole jsou stručně popsány jednotlivé části implementace.

2.1 Vstupní bod programu, `feedreader.cpp`

V tomto souboru se nachází funkce `main`, která příjme vstupní argumenty programu a následně jsou zpracovány třídou `ArgumentProcessor` 2.2. Pokud zpracování argumentů selže, je program ukončen s chybovým kódem.

Dále je vytvořen seznam URL zdrojů, které se budou zpracovávat. Tento seznam je vytvořen třídou `UrlListFactory` 2.4.

Seznam zdrojů URL je dále zpracováván třídou `FeedProcessor` 2.5. Pokud zpracování selže, program skončí s chybovým kódem, v opačném případě program úspěšně skončí.

2.2 Zpracování argumentů, `ArgumentProcessor`

Třída `ArgumentProcessor` je definovaná v souboru `ArgumentProcessor.hpp` a implementovaná v souboru `ArgumentProcessor.cpp`.

Tato třída zpracovává vstupní argumenty programu programem `getopt` [3]. Pro zpracování argumentů jsem použil funkci `getopt_long`, která umí pracovat i s dlouhými jmény argumentů a chová se na všech systémech ekvivalentně na rozdíl od funkce `getopt`.

Při chybně zadaných argumentech je výpsána nápověda programu, případně chybová zpráva.

2.3 Zpracování chyb, `error.hpp`

V tomto hlavičkovém souboru jsou definovaná makra pro výpis chybových zpráv na standardní chybový výstup.

2.4 Vytvoření seznamu URL zdrojů, `UrlListFactory`

Třída `UrlListFactory` je definovaná v souboru `UrlListFactory.hpp` a implementovaná v souboru `UrlListFactory.cpp`.

Tato třída vytváří seznam URL zdrojů buď ze zadaného URL zdroje jako argumentu programu nebo ze souboru `feedfile` zadaného jako argument programu, který obsahuje jednotlivé URL zdrojů, na každém řádku jedno URL.

Pokd zpracování souboru `feedfile` selže, je vypsaná chybová zpráva. Prázdné řádky a řádky začínající znakem # v souboru `feedfile` jsou ignorovány. Počítá se s tím, že každý řádek tohoto souboru bude zakončen znakem `LF`.

2.5 Zpracování zdrojů, `FeedProcessor`

Třída `FeedProcessor` je definovaná v souboru `FeedProcessor.hpp` a implementovaná v souboru `FeedProcessor.cpp`.

Tato třída stahuje jednotlivé zdrojové soubory a následně je dále zpracovává. Stahování probíhá pomocí knihovny `OpenSSL` [1]. Při práci na tomto úkolu jsem studoval tento článek [4], a proto jsem se při implementaci této třídy řídil některými doporučeními z tohoto článku.

Zpracovávám postupně jednotlivé URL zdrojů, jedno po druhém. Pokud je na vstupu zadaných více URL zdrojů a zpracování jednoho ze drojů selže, tak pokračuji zpracováváním dalšího zdroje.

Nejdříve provedu validaci a prsování URL třídou `UrlParser` 2.6. Pokud toto selže, tak vypisuji chybovou zprávu.

Dále provádím připojení k danému zdroji. Pokud se používá SSL, tak nastavím umístění certifikátů pro ověření platnosti certifikátů předložených serverem. Umístění těchto certifikátů nastavuji buď z argumentů programu, pokud jsou tyto zadány nebo použiji výchozí umístění, které definuje `OpenSSL` knihovna. Pokud ověření certifikátů nebo připojení k serveru selže, tak vypíši chybovou zprávu.

Pokud úspěšně provede připojení k serveru, tak se nejdříve odešle HTTP GET požadavek na daný zdroj daného serveru. Následně se přečtě HTTP odpověď serveru. Pokud při zápisu, či čtení HTTP požadavku, či odpovědi nastane chyba, tak se vypíše chybová zpráva.

Při úspěšném přečtení HTTP odpovědi se provede parsování této odpovědi. Oddělí se hlavička od těla odpovědi a provede se validace. Za validní odpovědi jsou považovány všechny odpovědi s HTTP kódem s hodnotou x, kde $200 \leq x < 300$. Pokud validace HTTP odpovědi selže, je vypsána chybová zpráva.

Nakonec je provedeno parosvání XML dokumentu v těle HTTP odpovědi a výpis patřičných inforamcí třídou `XmlParser` 2.7. Pokud toto parsování selže, je vypsána chybový zpráva.

2.6 Parsování URL, `UrlParser`

Třída `UrlParser` je definovaná v souboru `UrlParser.hpp` a implementovaná v souboru `UrlParser.cpp`.

Tato třída slouží k parsování URL. Tj. k zjištění, jestli je předložené URL validní URL podle `RFC 3986` [5] a rozložení daného URL na jednotlivé části. Podporované jsou pouze schémata `http` a `https`. Pokud v URL není uveden port, tak se doplní podle schématu, buď `80` nebo `443`.

Samotné parsování URL probíhá pomocí knihovny pro regulární výrazy, viz [2].

2.7 Parsování XML, `XmlParser`

Třída `XmlParser` je definovaná v souboru `XmlParser.hpp` a implementovaná v souboru `XmlParser.cpp`.

Tato třída slouží k parsování XML zdroje. Jsou podporovány pouze zdroje ve formátu `Atom 1.0` [8], `RSS 1.0` [9] a `RSS 2.0` [6].

Samotné parsování XML probíhá pomocí knihovny `Libxml2` [10], [7].

Pokud XML struktura parsovaného souboru není validní, v souboru se nenachází kořenový element nebo typ XML souboru není podporovaný, vypíše se chybová zpráva.

2.7.1 Vypisování inforamcí ze zdrojového souboru

Nejdříve je vždy vypsán titulek daného zdrojového souboru. Pokud titulek ve zdroji chybí, tak se zpracovávání daného zdroje ukončí. Výpis titulku je vždy uvozen znaky „*** “ a zanončen znaky „ ***“.

Následně jsou vypisovány jednotlivé položky zdrojových soburů. Nejdříve je vypsán titulek položky a potom případně další doplňující informace. Nejdříve čas vytvoření nebo změny záznamu uvozen znaky „Time: “, potom jméno, či e-mailová adresa autora dané položky uvozena znaky „Author: “ a nakonec asociované URL uvozené znaky „URL: “. Pokud některá z položek není ve zdroji uvedena, tak se nebude vypisovat.

2.8 Překladový systém

Překladový systém jako takový je vytvořen v programu `CMake`. Definice tohoto systému se nachází v souboru `CMakeLists.txt`. Nachází se zde definice názvu spustitelného souboru, definice zdrojových souborů, definice použitého C++ standardu, definice argumentů překladače a definice linkovaných knihoven.

Spouštení překladu je realizováno programem `GNU Make`, který po spuštění `make` nebo `make feedreader` přeloží program nástrojem `CMake` a vytvoří spustitelný soubor `feedreader` v kořenovém adresáři projektu. Spuštěním příkazu `make clean` je možné odstranit všechny soubory vytvoření při překladu.

3 Návod na použití

Po překladu programu, viz 2.8, je vytvořen spustitelný program `feedreader`.

3.1 Syntaxe a sémantika spouštění programu

`./feedreader <url | -f <feedFile>> [-c <certFile>] [-C <certDir>] [-T] [-a] [-u]`

- `url` URL zdroje.
- `-f <feedFile>` Soubor `feedfile`. (Textový soubor, kde je na každém řádku uvedena URL zdroje. Prázdné řádky a řádky začínající znakem # jsou ignorovány. Poslední znak na každém řádku musí být `LF`.)
- `-c <certFile>` Soubor s certifikáty pro ověření platnosti certifikátu předloženého serverem při použití SSL/TLS.
- `-C <certDir>` Adresář, ve kterém se vyhledávají certifikáty, které se použijí pro ověření platnosti certifikátu předloženého serverem při použití SSL/TLS.
- `-T` Pro každý zdroj se navíc zobrazí informace o čase změny, či vytvoření zázamu.
- `-a` Pro každý zdroj se navíc zobrazí jméno, či e-mailová adresa autora záznamu.
- `-u` Pro každý zdroj se navíc zobrazí asociované URL záznamu.

Povinně je potřeba uvést buď URL zdroje nebo soubor `feedfile`. Podporovaná schémata zdrojů jsou `http` a `https`. Parametry je možné zadávat v libovolném pořadí. Je možné zadávat parametry bez argumentů i následujícím způsobem `-Tau`. Protože je zpracování argumentů realizováno programem `getopt`, tak platí standardní chování definováné tímto programem, viz [3].

Při chybně zadaných parametrech se vypíše napověda programu. Nápovědu je možné si vypsat i příkazem `./feedreader -h` nebo `./feedreader --help`.

4 Závěr

Podařilo se mi implementovat všechny části aplikace, které byly uvedené v zadání projektu. Při konečném testování projektu jsem nezaznamenal žádné chybné chování programu.

Projekt pro mě nebyl nijak časově náročný. Nastudování potřebných informací a samotná implementace mi nezabrala více než několik hodin.

Při práci na projektu jsem se především zdokonalil v programování v jazyce C++, naučil jsem se pracovat s knihovnou `OpenSSL` [1], zjistil, jak funguje komunikace přes HTTPS a jak funguje ověřování certifikátů [4], naučil jsem se pracovat s knihovnou `Libxml2` [10], [7], nastudoval jsem si, jak je definován formát URI podle `RFC 3986` [5], naučil jsem se pracovat s knihovnou pro práci s regulárními výrazy [2], naučil jsem se

zpracovávat argumenty programu programem `getopt` [3] a nastudoval jsem si definici `Atom 1.0` [8], `RSS 1.0` [9] a `RSS 2.0` [6].

Literatura

[1] OpenSSL. [online], [vid. 2018-11-17]. Dostupné z: https://www.openssl.org

[2] Regular expressions library. [online], březen 2018, [vid. 2018-11-17]. Dostupné z: https://en.cppreference.com/w/cpp/regex

[3] Parsing program options using getopt. [online], listopad 2018, [vid. 2018-11-17]. Dostupné z: https://www.gnu.org/software/libc/manual/html_node/Getopt.html

[4] Ballard, K.: Secure programming with the OpenSSL API. [online], srpen 2018, [vid. 2018-11-17]. Dostupné z: https://developer.ibm.com/tutorials/l-openssl

[5] Berners-Lee, T.; W3C/MIT; Fielding, R.; aj.: Uniform Resource Identifier (URI): Generic Syntax. [online], leden 2005, [vid. 2018-11-17]. Dostupné z: https://tools.ietf.org/html/rfc3986

[6] Board, R. A.: RSS 2.0 Specification. [online], březen 2009, [vid. 2018-11-17]. Dostupné z: http://www.rssboard.org/rss-specification

[7] Fleck, J.: Libxml Tutorial. [online], rev. 7. srpen 2004, [vid. 2018-11-17]. Dostupné z: http://xmlsoft.org/tutorial/xmltutorial.pdf

[8] M. Nottingham, E.; R. Sayre, E.: The Atom Syndication Format. [online], prosinec 2005, [vid. 2018-11-17]. Dostupné z: https://tools.ietf.org/html/rfc4287

[9] Swartz, A.: RDF Site Summary (RSS) 1.0. [online], prosinec 2000, [vid. 2018-11-17]. Dostupné z: http://web.resource.org/rss/1.0

[10] Veillard, D.: The XML C parser and toolkit of Gnome. [online], [vid. 2018-11-17]. Dostupné z: http://www.xmlsoft.org