Čtečka novinek ve formátu Atom a RSS s podporou TLS

Dokumentace k projektu z předmětu
Síťové aplikace a správa sítí (ISA)

Kirill Mikhailov(xmikha00)
12. listopadu 2022

Obsah

1) Úvod

Hlavním úkolem projektu je naimplementovat čtečku novinek, která by vypisovala informaci ze stažených zdrojů (tzv. **feeds**), a to ve formátech **Atom** a **RSS 2.0** a s možností volitelného výpisu dalších dostupných údajů ze zdroje. Pro tento projekt jsem vybral jazyk C++, protože on má spoustu užitečných knihoven a je objektově orientovaný, což mi přišlo jako dost dobrý důvod, proč si vybrat tento jazyk. Také práce s řetězci je mnohem jednodušší a flexibilnější než například v Cčku.

2) Návrh a implementace

Hlavním souborem celé čtečky je ***feedreader.cpp.*** V němž se vyvolávají jednotlivé části programu :

- Parsování argumentů (*proc_args.cpp, proc_args.hpp*)
- Parsování URL a sestavení seznamu odkazů (*proc_url.cpp, proc_url.hpp*)
- Spojení (*proc_conn.cpp, proc_conn.hpp*)
- Parsování XML (*proc_xml.cpp, proc_xml.hpp*)

Každá samotná část programu je implementována v odpovídajícím souboru.

2.1) Parsování argumentů

Tato část programu zpracovává vstupní údaje z příkazové řádky. Zaprvé projde celým vstupem a zjistí, zda je uveden nějaký konkrétní URL. Pokud ano, ho zapíše do objektu třídy **Arguments** a odkaz "vynechá" ze vstupu.

Potom pomocí konstrukce *switch-case* a knihovny **getopt.h** zpracujeme celý vstup, podle zadaných argumentů (možnosti argumentů viz. sekce 3.1) nastavíme odpovídající **flagy** (např. *authorViewFlag, timeViewFlag* atd.) na základě kterých potom budeme řešit vystup.

Pokud vstup nebude odpovídat očekávanému, program vypíše uživateli návod a tím se vykonání programu skončí

2.2) Parsování URL a sestavení seznamu odkazů

Pak objekt s argumenty se předává do parseru URL odkazů.

Seznamem odkazů je v mé implementací je vektor (*std::vector*) hodnot třídy **UrlDesc**, obsahující položky pro popis samotných částí odkazu (**scheme, authority, port, path**). Během inicializace objektu typu **UrlDesc** odkaz projde přes *regular expression* (je inspirován zdrojem [6] a trochu předělán pro potřeby projektu) a pomocí *std::regex_match()* a *std::smatch* dostaneme z toho URLu všechno potřebné.

Dodatkově ještě se na základě použitého schéma nastavuje i **port**, pokud není uveden v odkazu (http - 80, https - 443, ostatní protokoly nejsou podporované)

V případě, že byl zadán jenom jeden konkrétní, odkaz, ten prostě přidá do seznamu (seznam teda bude obsahovat jenom jeden prvek) a tím se práce této části programu skončí.
Pokud byl zadán **feedfile,** ten se otevře a projde celý soubor po řádcích a přidá odkazy do seznamu. Pokud řádek se začíná znakem '**#**' - je to komentář.

2.3) Spojení

V této části programu se pomocí doporučené knihovny **OpenSSL**(víz zdroj [1]) vytváří spojení. Táto knihovna nabízí rozhraní pro vytvoření spojení (jak bezpečného tak i nebezpečného) s použitím knihovny **BIO**, správu certifikátů a šifrování dat.
Většinou jsem se řídil návodem, daným v zadání (víz zdroj [5]), ale ten také potřeboval několik drobných úprav, jako například přidání inicializační funkce **SSL_library_init(),** bez použití které bylo hodně ztracených bytů, nebo program vůbec nefungoval (např. na školním serveru *merlin*).
Tato část programu přijímá seznam URL odkazů a spracovává ho v cyklu, který celý ten seznam prochází.
Při zabezpečeném spojení budeme muset ověřit certifikáty, které byly zadány uživatelem nebo se používají defaultní pomocí funkce **SSL_CTX_set_default_verify_paths()** (konkrétní popis rozdílu víz sekce 3.1).
Pro spojení bylo potřebné sestavit odkaz formátu **hostname:port**, pro toto je moje třída a **UrlDesc** velmi vhodná.

Po úspěšném spojení je třeba ručně vytvořit a poslat pomocí funkce **BIO_write()** (víz návod ve zdroji [5]) jednoduchý **HTTP Request** (viz zdroj [7]) na požadovaný web.
Pak třeba to přečíst pomocí funkce **BIO_read()**. Pro toto jsem si vytvořil buffer o velikosti 512 znaků (což znamená, že odpověď bude čtená po 512 bytech/znacích). Po úspěšném přečtení části **HTTP Response** buffer bude vyčištěn funkcí **std::fill_n()**.
Pokud se najde **XML** odpověď v responsu, tak bude přidana jako odkaz na řetězec do vektoru **XML**ů.

2.4) Parsování XML

Na parsování **XML** jsem použil velmi užitečnou a vhodnou pro tento projekt knihovnu **libxml2** (víz zdroj [10]). Jediný problém s tou knihovnou je ten, že většina dostupných zdrojů informace - jsou velmi staré stránky, nejsou transparentní a hledání na takovýchto webech je dost těžký proces.

Po získávání XML parserem vektoru XML dokumentů, začneme průchod tim vektorem. Pro každý dokument vytvoříme XML-strom pomocí funkce **xmlParseDoc()**.
Pokud jmeno kořenového úzlu je "feed", jedná se o **Atom** feedu, v případě že jeho jméno je "rss", je to **RSS 2.0** feed.
Program projde sestaveným stromem na základě struktury daného feedu. Zaprvé vytiskne hlavní titul feedu ve formátu "****** feed title ******", potom bude vyhledávat jednotlivé články a případné dodatečné informace, pokud jsou požadované na vystup(víz sekci 3.1).
V případě nějakých chyb - uvolnit zdroje funkcemi **xmlFreeDoc() a xmlCleanupParser()**.

Po tisknutí všech dostupných feedů, vyčistí zdroje, alokované příkazy "***new***" a tím běh programu se skončí.

2.5) Překlad a kompilace

Překlad je realizován klasický pomocí nástroje **Makefile** (soubor **makefile**), v němž se provádí linkování všech mých hlavičkových souborů a propojení knihoven **OpenSSL** a **libxml2.** Po překladu se vytvoří spustitelný soubor *feedreader*.

Pomocí příkazu ***make clean*** je možné smazat soubor *feedreader* a všechny objektové soubory (*.o).
Příkazem ***make test*** se spustí základní testy.

3) Návod ke spuštění

3.1) Syntaxe spuštění

./feedreader <URL | -f <feedfile>> [-c <certfile>] [-C <certaddr>] [-T] [-a] [-u]

- Povinně je uveden buď URL požadovaného zdroje (podporovaná schémata jsou **http** a **https**) nebo parametr **-f**, ktery za kterým pak nasleduje umístění souboru *feedfile*.
- Volitelný parametr -c definuje umístění souboru *certfile* s certifikáty pro ověření platnosti certifikátu SSL/TLS předloženého serverem.
- Volitelný parametr -C definuje umístění složky *certaddr*, kde se budou vyhledávat certifikáty pro ověření platnosti certifikátu SSL/TLS předloženého serverem.
- Pozn.: Pokud není uveden -c ani -C, použije se úložiště certifikátů získané funkcí **SSL_CTX_set_default_verify_paths()**.
- Při spuštění s parametrem **-T** se pro každý záznam zobrazí informace o čase změny či vytvoření záznamu
- Při spuštění s parametrem **-a** se pro každý záznam zobrazí jméno autora či jeho e-mailova adresa
- Při spuštění s parametrem **-u** se pro každý záznam zobrazí asociované URL

Poznámka k posledním třem bodům :
- Při spuštění s parametrem -T dává se přednost informaci o čase poslední změny (pokud se najdou obě varianty, vypíše se čas poslední změny, nebo pokud se nenajde čas poslední změny, vypíše se čas vytvoření záznamu)
- Stejně i s parametrem **-a** - přednost se dává jménu autora

3.2) Testování

Součástí programu je i test na odhalení chybných vstupů. Při spuštění testů se vytvoří složka **test_outputs** s výstupy, které je možné porovnat s

očekávanými hodnotami ze složky **test_expects.** Je několik možností spuštění testů :

- **make test** - spuštění testů při jíž přeloženém projektu
- **make with_test** - kompilace a spuštění testů
- **make clean_test** - odstranění složky **test_expects**

4) Závěr

Během implementace tohoto projektu naučil jsem se pracovat síťovou knihovnou **OpenSSL** (víz zdroj [1]), zjistil jak se provádí bezpečné a nebezpečné připojení, jak funguje ověření certifikátů SSL/TLS. Naučil jsem se pracovat taky i s knihovnou **libxml2** (víz zdroj [10]), zdokonalil jsem svou práci s regulárními výrazy v C++ a zjistil jsem jak s nimi používat **smatch,** naučil jsem se rozebírat .
Nastudoval jsem strukturu feedů typů **Atom**(víz zdroj [3]) a **RSS 2.0** (víz zdroj [2]) a jak z nich vytáhnout potřebná data.

Tento projekt pro mě byl téměř první projekt v jazyce C++, takže vylepšení mých C++ schopností je určitě dobrým bonusem.

Projekt se mi povedl, podařilo se mi implementovat všechny jeho části a spojit je dohromady a celý program funguje podle zadání a očekávání.

Jako systém kontroly verzí jsem použil GitHub, takže překládám odpovídající repozitář :
https://github.com/playfulFence/feedreader_atom.
(Celou dobu implementace projektu repozitář byl privátní a neviditelný pro ostatní, to se změní hned po termínu odevzdání na public repo)

5) Literatura

[1] OpenSSL toolkit. (online), [vid. 2022-11-13].
Dostupné z: https://www.openssl.org/

[2] RSS Advisory Board : RSS 2.0 format Specification. (online), [vid. 2022-11-13].
Dostupné z: https://www.rssboard.org/rss-specification

[3] M. Nottingham, Ed. R. Sayre, Ed. December 2005 : The Atom Syndication Format. (online), [vid. 2022-11-13].
Dostupné z: https://www.rfc-editor.org/rfc/rfc4287

[4] Mozilla Developer Pages : What is URL? (online), [vid. 2022-11-13].
Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

[5] Kenneth Ballard. August 2018 : Secure Programming with the OpenSSL API. (online), [vid. 2022-11-13].
Dostupné z: https://developer.ibm.com/tutorials/l-openssl/

[6] T. Berners-Lee, W3C/MIT, R. Fielding, Day Software, L. Masinter, Adobe Systems. January 2005 :
RFC 3986: Uniform Resource Identifier(URI): Generic Syntax.
(online), [vid. 2022-11-13]. Dostupné z: https://www.rfc-editor.org/rfc/rfc3986

[7] Subbendu Majumder, Citrix. September 2020 : Notes on the format of HTTP requests and responses. (online), [vid. 2022-11-13]. Dostupné z:
https://docs.citrix.com/en-us/citrix-adc/current-release/appexpert/http-callout/http-request-response-notes-format.html

[8] IBM Documentation. March 2023 : Atom feed format. (online), [vid. 2022-11-13].
Dostupné z:
https://www.ibm.com/docs/en/baw/19.x?topic=formats-atom-feed-format

[9] John Fleck. 2002,2003: Libxml Tutorial. (online), [vid. 2022-11-13].
Dostupné z: https://gnome.pages.gitlab.gnome.org/libxml2/tutorial/

[10] Daniel Veillard : Reference Manual for libxml2. (online), [vid. 2022-11-13].
Dostupné z: http://xmlsoft.org/html/book1.html