FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Technická dokumentace

Čtečka novinek ve formátu Atom a RSS s podporou TLS

Síťové aplikace a správa sítí

Martin Hlinský (xhlins01@stud.fit.vutbr.cz)

Obsah

1 Úvod

Cílem projektu bylo vytvořit čtečku novinek ve formátu Atom/RSS. Projekt byl vypracován v jazyce C++ s použitím standardních knihoven a knihoven `OpenSSL`[8] a `libxml2`[6]. Program umí číst soubory XML ve formátu Atom[3] a RSS 2.0[7] podle zadání.

2 Návrh a popis implementace

Projekt byl implementován v podobě tří hlavních modulů: `feedreader` 2.1 zahajující běh programu a následně volající funkce zbývajících souborů, `argparse` 2.2, který zpracovává vstupní parametry a `xmlretrieve` 2.3 zajišťující připojení k jednotlivým URL, čtení a zpracovávání jejich obsahu. Deklarace funkcí a struktur jsou prováděny v hlavičkových souborech, které jsou odděleny od souborů se samotným zdrojovým kódem. Překlad programu probíhá pomocí `Makefile` 2.4 a testy jsou implementovány ve formě shell skriptu v souboru `feedreadertests.sh` 2.5.

2.1 Vstup do programu — feedreader.cc

Soubor obsahuje funkci `main()`, která přijme vstupní parametry a následně je ve formě pole (vektoru) `args` předá společně s prázdnou strukturou `parameters` funkci `argParse()` 2.2, která parametry zpracovává. Po úspěšném zpracování argumentů se přesune do funkce `retrieveXMLDocs()` 2.3, které předává strukturu `parameters` a ve které dochází k připojení na požadované URL a následné stahování souborů ze zdrojů a jejich zpracovávání.

2.2 Zpracování argumentů — argparse.cc

Ve funkci `argParse()` se na začátku nastaví 6 příznaků pro možné parametry a zároveň 1 příznak, který kontroluje, zda se parametr neopakuje v jednom cyklu. Ve funkci se kontroluje, zda má více než 1 parametr a v případě, že nemá, vypíše na standardní výstup instrukce pro použití. Stejný manuál se vypíše při použití parametru `-h`. Následně se vstupuje do cyklu, který kontroluje obsah všech parametrů. Ty jsou zpracovávány a ukládány do struktury `parameters` jako jednotlivé proměnné. Pokud ve zpracovávání argumentů nastane chyba, program končí s návratovým kódem `EXIT_FAILURE` [1] a vypíše na standardní chybový výstup `stderr` chybovou hlášku odpovídající důvodu selhání.

Pro parametr `-f` se kontroluje, zda se jedná o textový soubor a v případě, že ano, odstraní ze souboru bílé znaky a komentáře a seznam URL uloží do pole `feedURLs`.

Parametry `-c` a `-C` pouze uloží jméno souboru do pole `certStrings` a `certFolders` a kontrola jejich správnosti se provede v pozdějším běhu programu pomocí speciálních funkcí.

Parametry `-T`, `-a`, `-u` při svém výskytu pouze přepnou své příznaky na hodnotu `true` a následně se pomocí regulárního výrazu kontroluje, zda se například nevyskytuje kombinace `-Tau` nebo jakákoliv jiná kombinace, která je podle příkladů v zadání správná. Pro účely regulárního výrazu jsem použil a upravil příklad[5], který jsem také označil ve zdrojovém kódu.

Pokud se nevyskytuje parametr `-f <feedfile>`, musí se vyskytovat `URL` a to ve formátu `./feedfile URL`, `./feedfile "URL"`, nebo `./feedfile 'URL'`. Více URL se v tomhle formátu nemůže vyskytovat a musí se použít parametr `-f`. Adresa musí začínat `http://` nebo `https://`, což kontroluje regulární výraz.

Funkce `argparse` vrací hodnotu 0 při úspěšném návratu z funkce a potřebné hodnoty ukládá do struktury `parameters`.

[1] Makro sloužící pro přenositelnost značící neúspěšné ukončení, obvykle s hodnotou 1

2.3 Zpracování XML — xmlretrieve.cc

Hlavní funkce souboru, `retrieveXMLDocs()`, začíná přiřazením hodnot proměnným `CAfile` a `CApath`. Následně se podle návodu OpenSSL nastaví speciální proměnné `bio`, `ssl` a `ctx` a volá se funkce `initCTX()`, která kromě volání dalších pomocných funkcí `OpenSSL` nastaví proměnnou `ctx` a umožní ověřování certifikátů.

Pomocí struktury `parameters` a její proměnné `feedURLs` začne procházení jednotlivých adres, které se získaly v modulu `argParse` popsaného v sekci 2.2. Adresa se na začátku rozdělí do pomocných proměnných, jako například `hostname`, `port`, `protocol` a další.

V pomocné funkci `setupBIO()` se vytváří zabezpečené připojení, vytvořeného převážně na základě příkladu [1] a [2]. Pokud je funkce úspěšná, vrací hodnotu 0. V opačném případě vrací hodnotu 1, při návratu z funkce vypíše chybu a přejde se na zpracování dalšího požadovaného odkazu.

Při úspěšnem návratu ze `setupBIO()` se na nastavené zabezpečené připojení posílá požadavek ve tvaru:

```
GET <path> HTTP/1.0
Host: <hostname>
Connection: close
```

Poté se v `do{...}while(...);` cyklu čte obsah odpovědi serveru. Přes pomocnou proměnnou `responseHeader` se porovná status s hodnotou. Pokud je status odpovědi 301, 302, 307, nebo 308, přečte se `Location:`, pomocí kterého se můžeme připojit na přesměrovanou URL, dokud nedostaneme odpověď OK. Pokud je status odpovědi 200, začíná fáze zpracování XML. Při jiném statusu odpovědi se vypíše chybová hláška a program začne zpracovávat další odkaz v cyklu.

Zpracování XML bylo uděláno pomocí knihovny `libxml2` a většina probíhá ve funkci `printFormattedXML()`. Funkce je implementovaná rekurzivně podle dostupných příkladů na stránkách knihovny[4]. Po nalezení titulku článku ve feedu následně funkce hledá `item` nebo `entry` a při úspěšném nalezení se volá `retrieveXMLEntryContent()`. V téhle funkci se porovnáváním názvu elementů hledá datum poslední aktualizace, autor a URL článku.

Při návratu z funkce se porovnává, zda se mají tyhle informace vypisovat a pokud ano, je vždy formátovaný stejným způsobem, v pořadí "Autor", "URL", "Aktualizace". V případě, že se nepovede informaci vyhledat, vypíše se například řetězec typu: "Autor: Nepodařilo se vyhledat autora článku".

Po výpisu formátovaného článku dle zadání se volají pomocné funkce pro uvolnění dokumentu a uvolnění paměti a pokračuje se v cyklu.

2.4 Překlad programu — Makefile

Slouží k překladu programu, spouštění testů a vytvoření archivu k odevzdání. Překlad probíhá pomocí `g++` a verze `c++17`. `CXXFLAGS` a `LDLIBS` jsou nastaveny podle lokálního zařízení a zároveň podle umístění na serverech `merlin` a `eva`. Pomocí `make test` se spouští testovací modul. Pomocí `make clean` se vymažou veškeré soubory a složky, které mohl Makefile vygenerovat. `make pack` zabalí všechny požadované soubory k odevzdání do souboru `xhlins01.tar`.

2.5 Testy — feedreadertests.sh

Jedná se o jednoduchý testovací nástroj, který spouští program a porovnává návratové hodnoty s těmi, které očekává. Před koncem běhu programu skript vypíše kolik prošlo úspěšně testů z celkového počtu.

3 Spuštění programu

Program se spouští voláním `./feedreader [-h] <URL | -f <feedfile>> [-c <certfile] [-C <certaddr>] [-T] [-a] [-u]`, kde:

- `[-h | --help]` Slouží pro výpis nápovědy
- `<URL | -f <feedfile> | --file <feedfile>>` Slouží pro upřesnění zdroje XML. Může se vyskytovat pouze URL, nebo parametr -f, nebo parametr –file
- `[-c <certfile> | --certFile <certfile>]` Název souboru obsahující detaily certifikátu
- `[-C <certaddr> | --Certpath <certaddr>]` Cesta k adresáři obsahující certifikáty
- `[-T]` Přepínač určující zda se má při výpisu feedu ukazovat časová stopa poslední aktualizace
- `[-a]` Přepínač určující zda se má při výpisu feedu ukazovat jméno/e-mail autora
- `[-u]` Přepínač určující zda se má při výpisu feedu ukazovat URL článku

Pořadí parametrů je libovolné a program dovoluje například spuštění ve tvaru `./feedreader -uT -f <feedfile>`. Případné detaily jsou ujasněné v sekci 2.2.

4 Závěr

Zpracování argumentů bylo původně prováděno přes knihovnu `getopts`, která mi nevyhovovala a zvolil jsem proto metodu vlastní implementace zpracovávání argumentů. Pokračoval jsem studováním `OpenSSL` dokumentace[8]. Narazil jsem na dva příklady[1], [2], ze kterých jsem čerpal mnoho informací a jejichž implementaci jsem použil i ve svém projektu. Všechny části, kde jsem použil kód z příkladů, jsou označeny a ozdrojovány ve zdrojovém kódu. Po dokončení práce na zabezpečeném připojení jsem začal se zpracováním XML. Studoval jsem dokumentaci knihovny `libxml2` a našel jsem příklady[4], které mi pomohly v pochopení práce s touhle knihovnou. Převzaté části kódu jsou označeny, ale v tomhle případě jsou upravené pro potřeby projektu. Průběžně jsem pracoval na testovacím skriptu a zkoušel v něm své vlastní vstupy a odhadování návratových hodnot. Na závěr jsem předělal program tak, aby byl více modulární, dokončil jsem dokumentaci a všechny funkce a nejasné části kódu jsem okomentoval.

Literatura

[1] Ballard, K.: Secure programming with the OpenSSL API. 2004, [Online; navštíveno 13. 11. 2022]. Dostupné z: https://developer.ibm.com/tutorials/l-openssl/

[2] Dukhovni, V.: SSL/TLS Client. [Online; navštíveno 13. 11. 2022]. Dostupné z: https://wiki.openssl.org/index.php/SSL/TLS_Client

[3] Nottingham, M.; Sayre, R.: The atom syndication format. 2005, [Online; navštíveno 13. 11. 2022]. Dostupné z: https://www.rfc-editor.org/rfc/rfc4287

[4] Seketeli, D.: [Online; navštíveno 13. 11. 2022]. Dostupné z: https://gnome.pages.gitlab.gnome.org/libxml2/examples/

[5] Stribiżew, W.: How to match any combination of letters using regex? 2021, [Online; navštíveno 13. 11. 2022]. Dostupné z: https://stackoverflow.com/a/13546700

[6] Veillard, D.: XML parser. 1999, [Online; navštíveno 13. 11. 2022]. Dostupné z: http://www.xmlsoft.org/

[7] Winer, D.: RSS 2.0 specification (current). 2009, [Online; navštíveno 13. 11. 2022]. Dostupné z: https://www.rssboard.org/rss-specification

[8] Young, E. A.; Hudson, T.: OpenSSL. [Online; navštíveno 13. 11. 2022]. Dostupné z: https://www.openssl.org/