Vysoké učení technické v Brně
Fakulta informačních technologií

Čtečka novinek ve formátu Atom s podporou SSL/TLS

Dokumentace

28. duben 2015 Ján Jusko (xjusko00)

Obsah

1 Abstrakt **2**

2 Úvod do problematiky **2**

3 Návrh **2**

4 Použitie **2**

5 Popis implementácie **3**
5.1 Spracovanie argumentov 3
5.2 Vytvorenie spojenia 3
5.3 Spracovanie URL odkazov 3

6 Zaujímavé pasáže implementácie **3**
6.1 301 Moved Permanently 3
6.2 Transfer-Encoding: chunked 3
6.3 HTML tagy 4

1 Abstrakt

Cieľom bolo naimplementovať sieťovú aplikáciu *arfeed*, ktorá bude vypisovať požadované informácie uvedené v stiahnutých feed zdrojoch vo formáte Atom. Program po spustení stiahne konkrétne zdroje pomocou TCP/SSL spojenia a vypíše na štandardný výstup informácie zo zdroja (titulok, autor, link...). Implementačný jazyk aplikácie je C/C++.

2 Úvod do problematiky

Sieťová aplikácia *arfeed* komunikuje s webovými servermi pomocou protokolu SSL. SSL spojenie spočíva v princípe asymetrickej šifry, kde každá strana z komunikujúcich strán má dva klúče - verejný a súkromný. Verejný kľúč je možné zverejniť a pokud týmto kľúčom ktokoľvek zašifruje nejakú správu, SSL zaistí, že ju bude môcť rozšifrovať iba majiteľ použitého verejného kľúče svojim súkromným kľúčom. Počas SSL session je komunikácia šifrovaná.
Aplikácia sa do veľkej miery zaoberá formátom Atom. Jedná sa o webový štandard určený na publikovanie syndikovaného obsahu ktorý pomaly ale isto nahrádza formát RSS vďaka jeho dôslednému návrhu, otvorenosti a exaktnosti. Kanál *(feed)* aj jednotlivé položky *entry* musia mať uvedený svoj názov, jedinečný identifikátor *(URI)* a dátum aktualizácie. Povinné je rovněž jméno autora u každé položky. Atom rozoznáva rôzne formáty obsahu napr. prostý text, HTML, XHTML, XML, binárny obsah v kódovaní Base64 či odkaz na iný webový zdroj.

3 Návrh

Aplikácia je implementovaná ako jeden zdrojový súbor *arfeed.cpp* ktorý je však modulovaný do viacerých spolupracujúcich funkcií. Implementačný jazyk je C s integrovanými prvkami jazyka C++. Týmto spojením som skĺbil jednoduchú prácu s textovými reťazcami v jazyku C++ a zároveň som využil moje pokročilé znalosti jazyka C. Pre prácu so socketmi SSL/TLS som použil knižnicu *openssl* a na parsovanie stiahnutých xml súborov je použitá knižnica *libxml2*.

4 Použitie

Spúštanie: arfeed http://link/to/atom | -f feedfile [-c certfile] [-C certaddr] [-l] [-T] [-a] [-u]
Popis parametrov:

- Povinne je uvedený buď názov (URL) požadovaného zdroja, alebo parameter -f s dodatočným parametrom určujúcim umiestnenie súboru feedfile. Súbor feedfile je textový súbor, kde je na každom riadku uvedená jedna adresa zdroja vo formáte Atom.
- Voliteľný parameter **-c** súbor s certifikátmi, ktoré sa použijú pre overenie platnosti certifikátu SSL/TLS serveru.
- Voliteľný parameter **-C** určuje adresár, v kterom sa vyhledávajú certifikáty, které sa použijú pre overenie platnosti certifikátu SSL/TLS serveru. Defaultná hodnota je /etc/ssl/certs.
- Pri spustení s parametrom **-l** sa pre každý zdroj budú vypisovať iba informácie o najnovšom zázname.
- Pri spustení s parametrom **-T** sa pre každý záznam zobrazí naviac informácia o čase zmeny záznamu (ak je uvedená).
- Pri spustení s parametrom **-a** sa pre každý záznam zobrazí meno autora (ak je uvedené).
- Pri spustení s parametrom **-u** sa pre každý záznam zobrazí asociované URL (ak je uvedené).

5 Popis implementácie

5.1 Spracovanie argumentov

Spracovanie argumentov z príkazoveho riadku má na starosti funkcia *params*, ktorá používa vstavanú funkciu *getopt*. Snahou bolo dosiahnuť čo najvyššiu agílnosť a robustnosť, preto je povolené spúštanie programu s ľubovolným poradím všetkých argumentov. Avšak pri zadaní nepodporovaných parametrov sa aplikácia ukončí s návratovým kódom 99. Implementovaný je aj krátky a výstižný help ktorý sa spúšta parametrom -h. Funkcia params následne naplní a vráti pole príznakov korešpondujúcich so zadanými parametrami.

5.2 Vytvorenie spojenia

Pre vytváranie spojenia a sťiahnutie zdrojov sú implementované dve funkcie *connectoSSL / connectoNoSSL* ktoré ako návratovú hodnotu vracajú std::string obsahujúci odpoveď zo servera. Svojimi parametrami, návratovými hodnotami ani funkcionalitou sa funkcie nelíšia, ide o ekvivalentné prevedenie buď pomocou SSL spojenia alebo jednoduchého TCP spojenia. Pri chybe vzniknutej či už kôli certifikátom alebo neúspešnom pripojení sa funkcia neukončí, ale vracia error string ktorý sa ako získaná návratová hodnota testuje a príslušne spracuje. To je potrebné pri spracovaní viacerých zdrojov za sebou.

5.3 Spracovanie URL odkazov

Pod spracovaním odkazu sa rozumie nasledovná postupnosť akcií: rozparsovanie URL, vytvorenie spojenia, stiahnutie zdroja, parsovanie xml zdroja a nakoniec výpis požadovaných informácií.
V implementácii je všeobecne kladený dôraz na abstrakciu, preto je celý proces skrytý volaním jednej funkcie *processURL* ktorá postupne volá funkciu *parseURL* pre získanie potrebných reťazcov na vytvorenie spojenia extrahovaných zo vstupného URL smerujúceho k atom súboru. Ďalej zistí akým typom spojenia sa bude pripájať na server (SSL/TCP) a potom zavolá príslušnú funkciu *connectoSSL/connectoNoSSL*. Odpoveď zo servera sa potom pošle do funkcie *cut_header* ktorá analyzuje prijatý header, hľadá prípadné chyby a následne ho z reťazca odreže. Po tomto kroku je pripravený string vo formáte *XML* ktorý sa vo funkcii *myxmlParser* rozparsuje. Táto funkcia je rekurzívna, pre dosiahnutie nešpecifikovanej hĺbky zanorenia xml tagov. Následne sa naplní pole štruktúr *Entry* obsahujúcich informácie o jednom článku. Nakoniec funkcia *vypis* vypíše na štandardný výstup informácie zo štruktúr v požadovanom formáte.

6 Zaujímavé pasáže implementácie

6.1 301 Moved Permanently

Funkcia *cut_header* je zaujímavá svojou schopnosťou vyhľadať v hlavičke chybu *301 Moved Permanently* kedy je odkaz na daný zdroj presunutý na inú lokáciu. V takomto prípade je priamo z tejto funkcie volaná funkcia pre spracovanie novej lokácie a následne *cut_header* vracia string "PROCESSED"čo značí, že sa po návrate ďalej nemá pokračovať pretože link bol spracovaný iným volaním.

6.2 Transfer-Encoding: chunked

Opäť funkcia *cut_header* je schopná zároveň detekovať príjem tzv. chunk odpovede. To je ale nepodporovaný formát pre XML parser, preto je potrebné z textu vymazať hexadecimálne čísla.

6.3 HTML tagy

Aplikácia vypisuje na štandardný výstup terminálu, preto pre lepšiu čitateľnosť je pri výpise detekovaný výskyt HTML tagov ktoré sú následne v šikovnej funkcii *eraseTags* odstránené.

Literatúra

[1] K. Ballard. Secure programming with the OpenSSL API, Part 1 overview of the api, 2012.