Dokumentace ke projektu do ISA

Čtečka novinek ve formátu Atom a RSS s podporou TLS

Maxim Pličkа(xplick04)

28. října 2022

Obsah

1 Úvod do problematiky

Cílem projektu bylo vytvořit konzolový program, který ze zdrojových URL adres zpracuje jejich feed a následně vypíše na standartní výstup požadované infromace. Podporované formáty feedu jsou atom a RSS 2.0. Jazyk zvolený pro vývoj tohoto programu byl C++.

2 Návrh a popis implementace

2.1 feedreader

Modul obsahuje hlavní smyčku, ve které se nachází instanciace tříd `ArgumentParser` a `URLProcesor` a také vyvolání jejich metod. Tyto metody se vyvolávají v tomto pořadí: `parseArgs()`, `parseURL()`, `processURL()`. V případě že kterákoli z těchto metod vrátí chybu, se program ukončí s návratovou hodnotou 1. Pokud se tak nestane tak vrací 0.

2.2 argumentParser

Tato třída slouži ke zpracování argumentů od uživatele a taky následnému předávání těchto argumentů do ostaních modulů.

Atributy:

- `url` - obsahuje zadanou URL adresu
- `feedfile` - obsahuje adresu feedfile, v němž se nachází URL adresy
- `certfile` - obsahuje soubor s certifikáty
- `certaddr` - obsahuje adresář, ve kterém se nacházi certifikáty
- `Tflag` - reprezentuje argument -T
- `aFlag` - reprezentuje argument -a
- `uFlag` - reprezentuje argument -u

Metody:

- `parseArgs()` - slouží k načtení argumentů do patřičných atributů. Jejími parametry jsou `argc` a `**argv`. Zpracovávání probíhá v cycklu. V tomto cyklu se pomocí funkce `getopt_long()` rozhoduje, který argument byl zadán a připadně se zde i přiřazují parametry těchto argumentů do příslušných atributů. V připadě chybného zadání argumentů funkce vrací false, který následně v `main()` ukončí program s chybou. Na jeho konci se dále kontroluje, jestli byl zadán soubor feedile a nebo konkrétní URL adresa. V případě zadání neznámého argumentu, nebo specifikace feedfilu i konkrétni URL adresy vrací false.
- `getURL()`, `getfeedfile()`, `getcertfile()`, `getcertaddr()`, `getTflag()`, `getaFlagL()`, `getuFlag()` jsou pomocné metody pro přístup k atributům této třídy.

2.3 urlProcessor

Tato třída se stará o kontrolu formátu vstupních URL adres a také o naplnění vektoru `URLList` těmito validními URL adresami .V této třídě se nachází instanciace tříd `urlClass` a `XMLProcessor` a také vyvolání jejich metod, které jsou potřebné při následném zpracovávání obsahu jednotlivých URL adres.

Atributy:

- `URLList` - vektor, který v sobě drží URL adresy
- `curIndex` - integer, který slouží pro určení aktuální pozice v `URLList`
- `argumentParser` - objekt, který se využívá k přístupu k atributům od uživatele
- `bio` - objekt, který drží připojení k URL adrese
- `url` - objekt, který repzrezenuje aktuálně zpracovávanou URL adresu
- `numberOfSuccesses` - počet úspěšně zpracovaných URL adres

Metody:

- `isValid()` - privátní metoda, která pomocí regulárního výrazu vrací, zda je URL adresa ve správném formátu. V případě, že tomu tak je vrací true, jinak false.
- `urlProcessor()` - konstruktor, který nastaví do `argumentParser` objekt z předchozí části programu.
- `parseURL()` - metoda, která do `URLList` nahraje zdrojové URL adresy. V případě zadání feedfile se zde ověřuje, zda šlo tento zdroj otevřít. Pokud by některá z uvedených URL adres nebyla validní, vypíše se na standartní chybový výstup a načítání pokračuje dál. V případě zadání jenom jedné adresy se zkontroluje její validita a následně se přidá do `URLList`. Nakonec se kontroluje, jestli `URLList` není prázdný. Pokud tomu tak je, vrací false.
- `processURL()` - v této metodě se nachází cyklus, který postupně prochází všechny URL adresy z `URLList` a nahrává je do třídy `url`. V případě že aktuálně zpracovávaná adresa není zabezpečená, se vytvoří spojení k této adrese, které se uloži do `bio`. V případě, že se jedná o zabezpečené spojení se nastaví certifikáty u kterých se ověřuje jejich platnost. Poté se vytvoři spojení stějně jako u nezabezpečené URL. Následně se na toto spojení zašle HTTP dotaz a čeká se na odpověď. Pokud selže zaslání dotazu nebo získávaní odpovědi, přeruší se zpracovávání URL a přeskočí se na další. Poté co odpověd dorazí se z ní odsekne HTTP hlavička a pošle se na dalši zpracovávání do `XMLParser`. Na konci cyklu se uvolní zdroje pro další cyklus. Dále se v této metodě počítá počet úspěšně zpracovyných URL adres. Děje se tomu proto, aby vrátila false, v případě že nebyla úspěšně zpracována ani jedna URL adresa. Toto ve `feedreader` ukončí program s chybovým kódem 1.

2.4 urlClass

Tato pomocná třída slouží k reprezentaci aktuálně zpracovávané URL adresy.

Atributy:

- `url` - obsahuje URL adresu
- `secured` - boolean, který určuje zda se jedná o HTTPS
- `host` - obsahuje hostName, který se využívá při vytváření požadavku na server
- `path` - obsahuje cestu, která se využívá při vytváření požadavku na server
- `connectionString` - obsahuje řetězec, který se využívá při připojení na server
- `response` - obsahuje odvpoděď, která byla zaslána serverem

Metody:

- `load()` - metoda, která nastaví do atributů jejich aktuální hodnotu na základě URL adresy v argumentu
- `cleanResponse()` - z `response` odřeže HTTP hlavičku
- `responseAdd()` - přidá k `response` další načtenou část odpovědi, která je zaslána v argumentu
- `getSecured()`, `getHost()`, `getURL()`, `getPath()`, `getResponse()`, `getConnectionString()` - jsou pomocné metody sloužící pro přístup k atributům objektu

2.5 xmlParser

Třída slouží pro zpracovávání XML odpovědi ze serveru. Tato odpověd se zde nahraje do XML stromu a následně se zpracovává. Také se zde děje vypisování požadovyných informací na standardní výstup. Vždy se nejprve vypíše hlavní název a poté podnázvy jednotlivých feedů. V případě, že uživatel zadal všechny tři argumenty `-T`, `-a`, `-U` se pod jednotlivými podnázvy vypíší dodatečné infromace v pořadí v kterém se najdou. Navíc se v připadě zadání alespoň jednoho argumentu vypíše prázdný řádek, který odděluje tyto podnázvy.

Atributy:

- `active` - ukazatel na zpracovávaný prvek
- `doc` - ukazatel na dokument
- `argumentParser` - objekt, který se využívá k přístupu k atributům od uživatele
- `fist` - pomocný boolean, který se využívá k zákazu výpisu prázdného řádku před prvním názvem.

Metody:

- `parseXML()` - metoda, která se pokusí nahrát ořezanou odpověď ze serveru do XML stromu. Následně se zde rozhoduje o který formát se jedná. V případě atomu se vyvolá metoda `parseAtom()`, v případě RSS se vyvolá `parseRSS()`. Pokud se nejedná ani o jeden z těchto formátů, nebo se vyskytla chyba při zpracovávání odpovědi, vrací false.
- `parseAtom()` - v této metodě se nachází cyklus, který projíždí celou první vrstu XML, pokud narazí na uzel se jménem title, tak se na výstup vypíše jeho hodnota v požadovaném formátu. Pokud narazí na uzel se jménem entry, vyvolá se `parseEntry()`.
- `parseEntry()` - v této metodě se nachází pomocný boolean `titleFlag`, který slouží pro vypsání pouze prvního nadpisu. Dále se zde zanořuje do 2. vrstvy XML a postupně se prochází všechny uzly obsažené v konkrétní entry a pokud se zde nacházejí, vypisuje požadované hodnoty.
- `parseRSS()` - na začátku se nachází cyklus, který se pokouší najít uzel se jménem channel. Poté co ho najde, se do něj zanoří a následně se prohledává další vrstva. V této vrstvě se hledají uzly se jménem title a item. Pokud narazí na title, tak se vypíše na výstup. Pokud narazí na uzel s názvem item, vyvolá metodu `parseItem()`.
- `parseItem()` - funguje v podstatě stějne jako `parseEntry()` až na rozdíly v názvech jednotlivých uzlů. Pořadí výpisu dodatečných informací je také stejné.
- `XMLParser()` - konstruktor, který nastaví do `argumentParser` objekt z předchozí části programu.

3 Testy

Testy jsou napsány formou bash skriptu, kde se kontrolují návratové hodnoty jednotlivých příkazů. U testování konkrétních stránek se neporovnávají výstupy, protože se obsah stránek mění a tak by nebylo možné porovnávat výstup s referenčním výstupem.

4 Návod na použití

Projekt se překládá pomocí přikazu `make`. Testy se spouští pomocí příkazu `make test`.

4.1 Spouštění programu

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

- `URL` - zdrojová URL
- `-f <feedfile>` - soubor, který v sobě má zdrojové URL adresy. Každá URL adresa má svůj řádek a řádek je ukončen pomocí znaku LF. Pokud řádek začíná #, je celý řádek ignorován.
- `-c <certfile>` - soubor obsahující certifikáty pro ověření platnosti certifikátu serveru
- `-C <certaddr>` - adresář obsahující certifikáty pro ověření platnosti certifikátu serveru

- `[-T]` - vypíše datum poslední aktualizace u jednotlivých zdrojů
- `[-a]` - vypíše autora u jednotlivých zdrojů
- `[-u]` - vypíše asociované URL adresy u jednotlivých zdrojů

5 Závěr

Podařilo se mi implementovat všechny části tohoto programu. Navíc mi projekt přinesl spoustu zajímavých zkušeností a to jak z pohledu zabezpečené komunikace se serverem tak i z pohledu programování v C++. Pro studium tohoto problému jsem využil návodu pro OpenSSL[2]. Dále jsem použil řešení na některé z regulárních výrazů [1][3]. Pro implementaci jsem využil knihoven `openSSl` a `libxml2`.

Reference

[1] url validation with http/https as prefix. [online]. Dostupné z: https://www.regextester.com/105539

[2] Ballard, K.: Secure programming with the OpenSSL API. [online], 2004, 15. srpna 2018. Dostupné z: https://developer.ibm.com/tutorials/l-openssl/

[3] Costa, C.: CPP + Regular Expression to Validate URL. [online], 2011, 7. listopadu 2021. Dostupné z: https://stackoverflow.com/questions/5620235/cpp-regular-expression-to-validate-url