ISA – protokol k projektu

Juraj Novosád

13. novembra 2022

Obsah

1 Zadanie

Cieľom projektu je vytvorenie nástroja na čítanie správ zo servera poskytujúceho RSS alebo atom správy a vypísať ich obsah na konzolu. Server musí komunikovať cez http alebo https protokol, pričom nástroj je schopný overiť platnosť certifikátu servera komunikujúcom cez https a rozšifrovať správu.

2 Teória

Rss a atom sú súbory, zapísané vo formáte XML. Primárne sú používané na štruktúrované zapisovanie krátkych správ o udalostiach.

2.1 Atom

Tento formát popisuje list informácií, ktoré nazývame `feed`. `Feed` je koreňová entita, obsahujúca jednotky, ktoré nazývame `entry`.
`Entry` zapúzdruje konkrétny záznam. Pre tento projekt je dôležité, že obsahuje `title`(nadpis), `link` a `author`, `link` a `author` nie je povinný. Voliteľne môže obsahovať `id`, `summary`, `updated`, ... Samotný feed môže obsahovať autora feedu a nadpis celého feedu. Tieto informácie sa môžu vyskytnúť hocikde v najvyššej vrstve xml dokumentu. Ak nie je v `entry` definovaný autor, ako autor `entry` je považovaný autor celého feedu. Informácie o atom formáte som čerpal z rfc4287

2.2 RSS

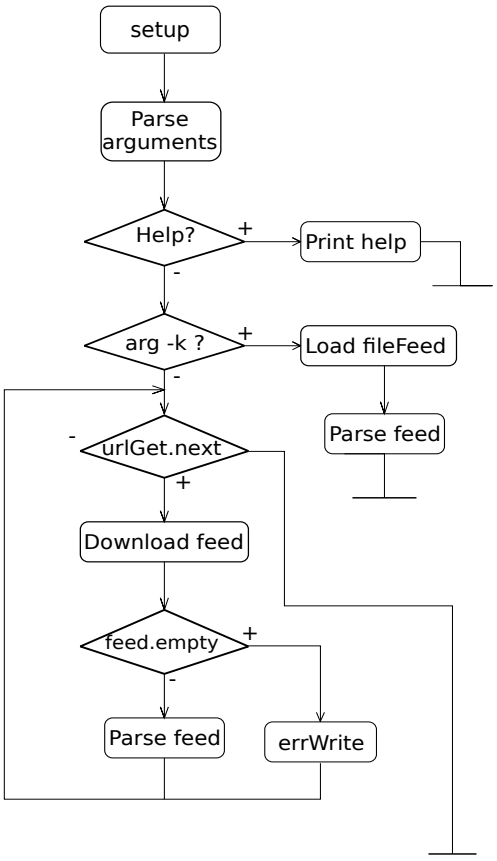
RSS formát zapúzdruje informácie o krátkych správach, je to akronym pre Really Simple Syndication. Koreň formátu je tag `rss`, v ktorom sa špecifikuje verzia. `Rss` tag obsahuje práve jeden tag `channel`.
`Channel` zapúzdruje informácie pre čitateľa. Musí obsahovať elementy `title, link, description`. Voliteľne môže obsahovať tagy `language, copyright, pubDate, ....` `Channel` voliteľne zapúzdruje samotné správy v tagoch item.
Item obsahuje `title, link a description`. Voliteľne môže obsahovať `author`, `category`, `comments`, `enclosure`, `guid`, `pubDate`, `source`. Informácie som čerpal z rss špecifikácie.

3 Priebeh implementácie

Projekt je písaný v c++ a primárne zostavovaný pomocou `CMakeLists`, ktorý vytvorí `Makefile`. Makefile je potom možné používať na zostavovanie a spúšťanie testov.
Na uľahčenie kontroly kompatibility s referenčným prostredím na serveri merlin som vytvoril pipeline, ktorá sa pri každom push na github spustí a pokúsi sa zostaviť program a spustiť testy. Referenčné prostredie podobné merlinovi som vytvoril ako docker image, ktorý podľa možností kopíruje verzie potrebných programov a knižníc. Základ pre docker image je v zložke `scripts/build.Dockerfile`.

4 Použité knižnice

Na komunikáciu so serverom je použitá knižnica `openssl`. Na spracovanie xml súborov je použitá knižnica `libxml2`. Na spracovanie url adries je použitá štandardná knižnica `regex`.



Obr. 1: Architektúra programu

5 Priebeh spracovania

Beh programu je načrtnutý na priloženom vývojovom diagrame 1.

Vo fáze setup sa vytvorí a nakonfiguruje objekt pre spracovanie argumentov a inicializujú sa pomocné premenné. Blok `Parse arguments` spracuje argumenty z príkazového riadku a poskytuje objekt. V prípade ak boli parametre chybne zadané, vyhodí výnimku a program sa ukončí. Kontroluje existenciu zadaných súborov a priečinkov, správnosť výrazov. Ak je zadaný prepínač -h program vypíše na konzolu nápovedu a korektne sa ukončí.
Ak užívateľ zadal parameter `-k <meno_súboru>`, ignorujú sa parametre pre url a feedfile. Má vyššiu prioritu. Načíta sa feed zo súboru, pričom feed musí byť v presne takom formáte ako by sa stiahol z internetu. Tento feed sa spracuje a program sa ukončí. Toto rozhranie je len pre testovacie účely.
`UrlGet` je objekt, ktorý v konštruktore prijíma meno súboru feedfile a URL. Spracuje Url a poskytuje ich pomocou metódy next, pomocou ktorej iterujeme cez všetky načítané adresy.
Na stiahnutie správ zo serverov je už vytvorený objekt `downloader`. Tento objekt je volaný so spracovanou štruktúrou url adresy. Ako návratová hodnota je reťazec so správou. Správa sa spracuje vo `feedparser`, on spracuje xml a vypíše hodnoty na konzolu.

5.1 Výpis do konzoly

Aby bola dosiahnutá istá modularita pri výstupe programu, sú všetky výpisy realizované cez loger. Tento objekt zapúzdruje výpisy cez rozhranie `write` a `errWrite`.
Logger má rozhranie, ktoré umožnuje formátovať, rovnako ako `printf`, vnútorne sa používa snprintf

na formátovanie. Konštruktor objektu berie dva parametre, referenciu na stream do ktorého potom zapisuje pre normálne výpisy a chybové výstupy. Takto viem kontrolovať veľmi ľahko, ak by sa malo zapisovať do súborov alebo do stringstreamu, stačí iba dodať iné streamy. Takisto by mohlo byť v budúcnosti ľahko implementovateľná úroveň debugovacích výstupov, kde by stačilo zmeniť iba implementáciu loggeru a bolo by možné povedať výpisy akej úrovne sa majú aj vypísať.
Logger je definovaný v `./src/include/utils/logger.h`.

5.2 Spracovanie argumentov

Implementácia spracovania bola architektúrne pojatá ako návrhový vzor builder. Z hľadiska užívateľa je použitie čo najjednoduchšie. Zadá aký parameter chce spracovávať, pričom špecifikuje pod akým prepínačom bude, alebo ak je pozičný aký názov bude mať handle. Pritom zadá aj či má byť za prepínačom hodnota. Následne môže zadávať functory, ktoré sú vlastne kontroly, a špecifikujú vlastnoti daného prepínača, ktoré musia byť platné aby prebehlo spracovanie argumentov. Tieto kontroly sa môžu za sebou reťaziť.

Algoritmus 1: Spracovanie argumentov

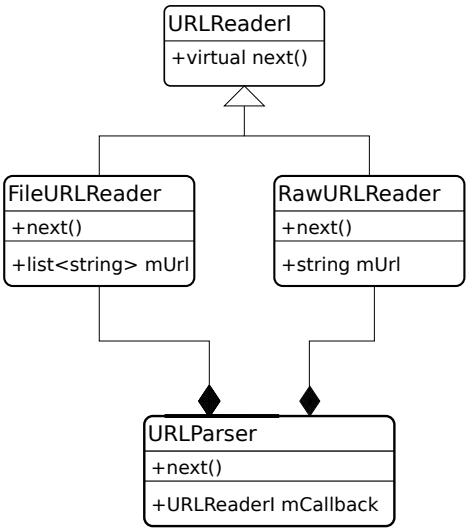
Input: $argc, argv$

```
1  argParser.addParameter('-h');
2  argParser.addValueParameter('-f').isFile();
3  argParser.addValueParameter('-C').isDirectory().isNotEmpty();
4  argParser.addPositionalParameter('nameforHandle', position);
5  try{
6     argParser.parse(argc, argv);
7  }
8  catch(const feedreaderException::argumentParsing& err)
9  {
10    logger->errWrite('Error parsing arguments: %s\n\n', err.what());
11    logger->errWrite(HELP_MSG);
12 }
```

Spracovanie argumentov je definované v `src\include\parameter_parser`.

5.3 Spracovanie url

`URLParser` 2 je objekt zapúzdrujúci buď `FileUrlReader` alebo `RawUrlReader`(spracovanie jednej url). `URLParser` potrebuje v konštruktore meno feedfile alebo URL. Ak sú obe možnosti zadané naraz, vyhodí výnimku a program sa ukončí. Inak podľa toho, čo je zadané vytvorí správny url parser a uloží si ho ako callback. Ponúka metódu next, ktorá vráti hodnotu ďalšej načítanej url zdroja a keď už nebudú ďalšie url vráti false. Metóda next len volá metódu next callbacku.
Url sa spracujú a uložia sa do pamäte ako štruktúry `URLAddress`. Rozdelenie na jednotlivé časti sa vykoná pomocou regulárneho výrazu, ktorý url rozdelí na jednotlivé skupiny. Zároveň ak je url zadaná bez špecifikácie protokolu (http/http) automaticky sa doplní https. Spracovanie url je implementované v `src\include\urlParser`.



Obr. 2: URLParser diagram tried

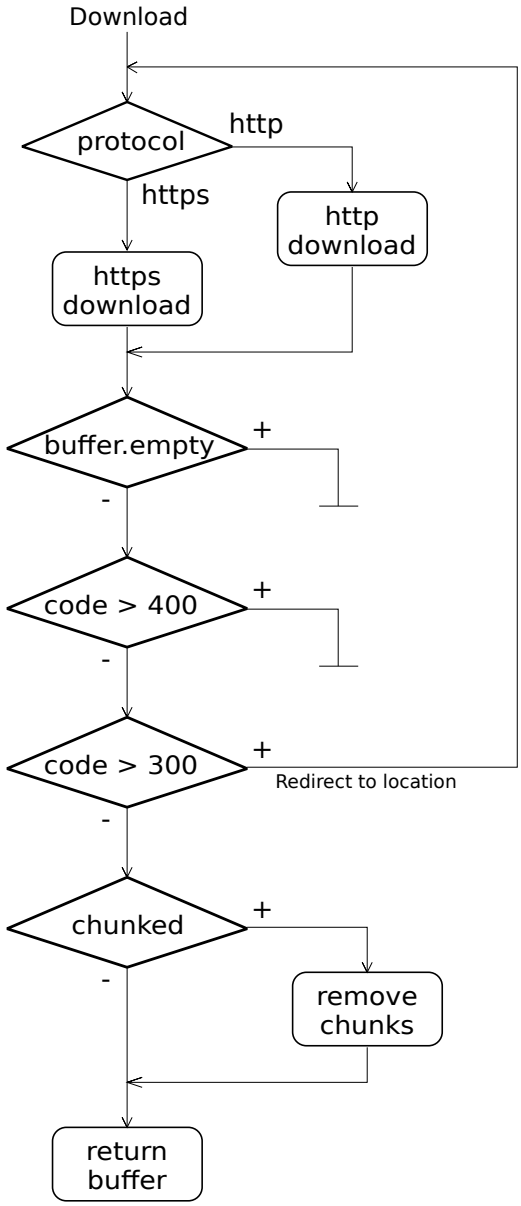
```
struct URLAddress{
    string protocol;
    string address;
    string path;
    string port;
    string options;
    string original;
};
```

5.4 Stiahnutie správ

Zodpovednosťou downloaderu 3 je nadviazať so serverom spojenie, stiahnuť správy, očistiť ich od http hlavičiek a poskytnúť správu ďalej. V downloaderi som implementoval aj mechanizmus na prípadné presmerovanie. Spracováva hlavičku http a ak je návratový kód v rozsahu <300, 400), podľa hlavičky location sa pokúsi načítať správu z danej stránky. Downloader posiela požiadavky HTTP verzie 1.1, to znamená že súbor môže byť rozdelený, čo je signalizované hlavičkou `transfer-encoding=chunked`. V tom prípade downloader značky chunkov odstráni.

5.5 Spracovanie xml

Samotné spracovanie xml je realizované pomocou knižnice libxml2. Spracovanie prebieha v objekte `xmlParser`, ktorý sa spolieha na `xmlNode`.
`XmlNode` je abstrakcia nad xml súborom. Ponúka metódy na prechádzanie uzlov a poskytovanie údajov o daných uzlov a poduzlov. Tieto uzly sa vytvárajú pomocou objektu `XmlBuilder`. `XmlParser` si vytvorí pomocou builderu `XmlNode` z koreňového uzlu. Podľa typu uzla rozhodne o aký typ správy sa jedná(rss, atom, iné) a tento uzol vloží do metódy určenej na jeho spracovanie. Spracovanie prebieha tak, že iterovaním cez uzly v najvyššej vrstve vyhľadávame tie, ktoré nás zaujímajú(title, entry, item, ...). Ak sa dostaneme na niektorý zo spomenutých uzlov, spustí sa funkcia na jeho spracovanie. Napríklad pri nájdení uzlu `item`, tak podľa zadaných prepínačov sa v jeho potomkoch vyhľadáva uzol autor, link alebo dátum zmeny. Ak sa spracováva atom, je zadaný argument '-a' a nenájde sa v atom feede autor



Obr. 3: Vývojový diagram stiahnutia správy

príspevku, tak sa ako autor príspevku použije autor celého dokumentu.
`XmlParser` je zodpovedný aj za vypísanie zistených informácií na konzolu. Ako členskú premennú má referenciu na logger a do neho vypíše už naformátované správy.

6 Testovanie

Testy sa spúšťajú v rámci frameworku CTest. CTest postupne spustí testovacie scenáre najprv pre unit testy a potom spustí skript pre integračný test. Unit testy sú implementované pomocou GTestu. Framework GTest sa stiahne z oficiálneho github repozitára počas kompilovania testov.
Samotné testovacie scenáre testujú funkciu jednotlivých modulov, pričom každý modul má skupinu

testov v podobe jednej binárky. Čiže je jedna testovacia binárka na jeden modul, v respektíve knižnicu. Integračné testy sa spúšťajú nad samotným programom feedreader. Feedreader má pridanú možnosť spracovania feedu zo súboru. Na testovacie účely je definovaný prepínač -k <meno súboru>. Testovanie potom prebieha tak, že sa po jednom spracovávajú súbory zo zložky tests/test_files a výstup z programu sa porovnáva s referenčným výstupom z tests/reference.

7 Popis použitia

Aplikácia je dodávaná v forme zdrojových súborov. Skompilovať je možné pomocou príkazu:

```
make
```

Makefile vytvorí zložku build, do nej nechá Cmake nagenerovať Makefile a ten potom spustí. Akonáhle sa program preložil môže sa otestovať spustením príkazu:

```
make test
```

Použitie:

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

Program podporuje možnosti:

-f	Špecifikuje súbor s URL adresami, z ktoré sa spracujú
-c	Špecifikuje certifikát, na overenie servera
-C	Špecifikuje zložku s certifikátmi na overovanie
-T	Na výstup vypíše dátum poslednej aktualizácie
-a	Na výstup vypíše meno autora záznamu
-u	Na výstup vypíše url z každého záznamu
-k	Špecifikuje offline súbor na spracovanie

Príklad použitia:

`./feedreader https://www.fit.vut.cz/fit/news-rss/`
Spracuje jednu stránku

`./feedreader -f urls.txt -aT`
Spracuje všetky url z daného súboru