Raport – XML Parser

1. Rezumat

1.1. Overview

Problema constă în scrierea unui program (script) shell care să se ocupe cu parsarea fișierelor XML pentru a citi și scrie date din, respectiv în, format XML. Fișierele XML au o utilizare mare în stocarea și transferul datelor, fiind ușor de procesat. Crearea unui parser simplu este importantă pentru a gestiona ușor fișiere fără a utiliza limbaje de programare mai avansate.

1.2. Capitole

Vor fi abordate următoarele capitole în cadrul raportului:

- Descrierea problemei. Specificarea soluției
- Design și implementare
- Evaluarea soluției
- Concluzii

2. Descrierea problemei. Specificarea soluției

2.1. Context general al problemei

Contextul general al problemei constă în manipularea datelor structurate, automatizarea procesării fișierelor XML și asigurarea accesibilității soluției pentru toți utilizatorii, inclusiv pentru cei cu puține cunoștințe de programare. Obiectivul principal este de a crea un script shell care să ofere funcționalitățile de bază (citire și scriere) ale tipului de fișiere precizat anterior.

2.2. Descriere a soluției

Soluția oferită se bazează pe utilizarea comenzii xmlstarlet, un utilitar puternic pentru procesarea fișierelor XML. Acesta permite citirea, modificarea și validarea fișierelor XML din linia de comandă, ceea ce îl face ideal pentru integrarea în scripturi shell. Printre funcționalități se enumeră citirea fișierului XML, ștergerea unui tag XML și, în plus, adăugarea unui tag XML.

2.3. Condiții de implementare

Condițiile necesare pentru a implementa funcționalitățile dorite sunt legate de structura fișierelor XML. Este necesar ca fișierul XML să existe și să aibă o structură validă pentru o bună funcționare. Mai mult decât atât, trebuie să fie instalat xmlstarlet, un instrument important în manipularea fișierelor XML pe Linux. Acesta poate fi instalat folosind comanda *sudo apt-get install xmlstarlet*. Este necesară și deținerea permisiunii de scriere în fișier.

2.4. Alte soluții posibile și avantaje, dezavantaje

Soluțiile posibile pentru rezolvarea problemei pot include utilizarea unor limbaje de programare mai avansate (Python cu xml.etree.ElementTree, Perl), folosirea unui parser XML dedicat în shell scripting (xmlstarlet) sau crearea unui script simplificat folosind comenzi de bază (precum 'cut' sau 'while').

- Utilizarea limbajelor de programare avansate: oferă flexibilitate, opțiuni de gestionare a erorilor, ușurință de întreținere și extensibilitate și au funcții externe, însă necesită cunoștințe suplimentare, instalarea limbajului și al librăriilor și complexitatea este crescută pentru sarcini simple, un exemplu concret fiind chiar manipularea unui fișier XML, ceea ce complică lucrurile.
- Utilizarea unui parser XML dedicat (xmlstarlet): este ușor de utilizat, rapid, independent de limbaje de programare și nu are dependențe majore, însă are limitări în manipularea complexă a XML-ului cu namespace-uri sau structuri adânci și este nevoie să fie preinstalat tool-ul xmlstarlet.
- Scripturi Shell simple: conferă simplitate și rapiditate, însă sunt multe limitări în manipularea fișierelor XML complexe, gestionarea tag-urilor fiind destul de greu de manipulat.

2.5. Evaluarea soluției

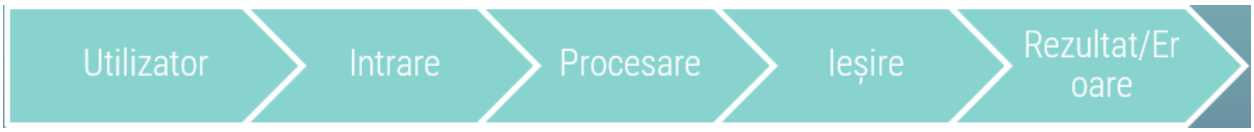
Evaluarea se face prin intermediul meniului programului. În funcție de ce dorește utilizatorul să realizeze, este pus să precizeze fișierul XML căruia să-i atribuie modificările sau afișările. Dacă, în urma adăugării sau ștergerii, fișierul este modificat, iar dacă în urma citirii, fișierul este afișat în consolă, atunci programul funcționează corespunzător și afișează un mesaj de succes pe ecran.

2.6. Presupuneri, constrângeri și dependente

Se presupune că fișierul XML dat este valid și bine format, nu este prea mare ca dimensiune pentru a fi procesat cu xmlstarlet, tool-ul xmlstarlet este preinstalat, iar fișierul se află pe sistemul local al utilizatorului, având drepturi de modificare. Programul nu poate funcționa fără xmlstarlet și nu poate manipula fișiere XML complexe, având suport limitat pentru fișiere mari. Scriptul este destinat să funcționeze pe un sistem bazat pe Unix (Linux, macOS).

2.7. Caracteristicile prototipului

Prototipul are o interfață simplă, cu un meniu și comenzi simple pentru interacțiunea cu utilizatorul, are o performanță redusă pentru fișiere mari sau complexe și poate fi modificat rapid pentru a adăuga funcționalități suplimentare sau pentru a rezolva probleme semnalate în timpul testării.



2.8. UI/UX Design

Utilizarea programului se realizează în consolă (fiind un script Shell) și este ușor de utilizat de către toți utilizatorii, inclusiv de cei care nu posedă cunoștințe avansate în programare.

3. Design și implementare

3.1. Descrierea arhitecturii software folosite

Arhitectura software este una simplă, bazată pe script Shell, care integrează comenzi externe ce manipulează fișiere XML. Programul urmează un model procedural, unde funcțiile sunt definite pentru a citi, adăuga și șterge tag-uri din fișiere XML. Scriptul folosește xmlstarlet pentru a edita și formata fișiere XML. Interfața este realizată prin linia de comandă, unde utilizatorul poate selecta ce dorește să facă, iar modificările sunt salvate în fișierul original.

3.2. Explicația alegerii mecanismelor și algoritmilor folosiți

Scriptul utilizează xmlstarlet pentru a manipula fișiere XML datorită fiabilității și capacității sale de a păstra structura validă a documentului. Algoritmii folosiți (citire, adăugare și ștergere de tag-uri) sunt direcți și eficienți. Meniul în linia de comandă oferă o interfață intuitivă pentru utilizatori, iar utilizarea fișierelor temporare previne pierderea datelor originale în caz de corupere.

3.3. Exemplificarea mecanismelor/algoritmilor folosiți

Citirea datelor dintr-un fișier XML:

- Comanda utilizată este *xmlstarlet fo “$file”.*
- Se citește fișierul XML și îl formatează pentru afișare într-un mod lizibil.
- De exemplu, dacă fișierul catalog.xml este dat, comanda va afișa conținutul acestuia într-un format ordonat.

Ștergerea unui tag specific:

- Comanda utilizată este *xmlstarlet ed -d "//$tag_name" "$file" > "$file.tmp".*

- *-d "//$tag_name"* identifică nodurile cu numele *$tag_name* din întreaga structură XML și le șterge.
- De exemplu, dacă utilizatorul introduce *tag_name*, toate nodurile *tag_name* vor fi eliminate.

Adăugarea unui tag specific:

- S-a utilizat comanda *xmlstarlet ed -s "/*[1]" -t elem -n "$tag_name" -v "$tag_value" "$file".*
  -s "/[1]"*: Adaugă un nou copil în primul tag.
  -t elem: Specifică faptul că adăugăm un element XML.
  -n "$tag_name": Numele noului tag, introdus de utilizator.
  -v "$tag_value": Valoarea noului tag, introdusă de utilizator.

3.4. Particularitățile adaptării acestor mecanisme

Mecanismul folosește XPath */*[1]* pentru a identifica primul tag din documentul XML, asigurând plasarea precisă a noilor tag-uri. Comanda *xmlstarlet ed* permite adăugarea dinamică de elemente în XML, menținând structura validă. Modificările sunt aplicate într-un fișier temporar, prevenind coruperea datelor în cazul unor erori.

3.5. Componentele software ale proiectului

Printre componentele software ale proiectului se enumeră scriptul Bash, xmlstarlet, fișiere XML, interfața interactivă a mecanismului din script și sistemul de fișiere propriu-zis al sistemului de operare.

3.6. Biblioteci software folosite

În acest proiect, biblioteca software principală utilizată este xmlstarlet, dar au fost folosite și instrumente native Bash, precum read, if și [[ ]] pentru verificări logice.

3.7. Probleme tehnice întâlnite

În timpul realizării programului, pot apărea următoarele probleme tehnice:

- Uneori, comanda xmlstarlet nu păstreaza corect formatarea fișierului XML, rezultând fișiere XML destul de greu de citit.
- Dacă strucutra fișierului este incorectă, xmlstarlet poate genera erori la aplicarea modificărilor, ceea ce poate duce la erori de procesare.
- Probleme de permisiuni, dacă scriptul nu are permisiuni de scriere.

3.8. Rezolvări probleme tehnice

Pentru a rezolva majoritatea problemelor tehnice, s-a verificat validitatea fișierului XML înainte de orice modificare, s-a verificat existența tag-ului înainte de adăugare pentru a nu produce o eroare, tag-urile se adaugă la un loc specificat (în primul element) pentru a nu compromite structura fișierului, iar la modificarea fișierului XML, dacă procesul ar fi întrerupt, fișierul ar putea rămâne într-o stare coruptă, de aceea s-au folosit fișiere temporare.

4. Evaluarea soluției

4.1. Descrierea arhitecturii hard/soft folosite

Arhitectura este bazată pe un sistem local de operare Linux, care gestionează fișiere XML și le manipulează folosind scripturi shell. Componenta software principală este un script Bash care interacționează cu fișierele XML prin xmlstarlet.

Arhitectura hardware este reprezentată de un sistem cu o unitate centrală de procesare (CPU), memorie RAM și stocare locală. Este esențială pentru rularea scripturilor Bash si manipularea fișierelor XML. Fișierele sunt stocate pe discul local și pot fi accesate și modificate direct prin intermediul scripturilor.

A fost utilizat un calculator cu 16 GB RAM, conectat la internet wireless pentru descărcarea utilitarului xmlstarlet. Mai mult decât atât, s-a utilizat și o mașină virtuală cu două nuclee din procesorul fizic, 4 GB RAM din cele 16 GB, stocare virtualizată și rețea virtualizată (bridge) pentru accesul la internet. Sistemul de operare gazdă folosește Linux, la fel ca mașina virtuală, unde s-a utilizat și un software de gestionare a resurselor.

4.2. Detalii despre teste

S-a utilizat un fișier XML existent pe disk, care are următorul conținut:

```xml
<catalog>
    <status>Disponibil</status>
    <descriptor>Acesta este un tag nou.</descriptor>
    <book id="bk101">
        <author>Gambardella, Matthew</author>
        <title>XML Developer's Guide</title>
        <genre>Computer</genre>
        <price>44.95</price>
        <publish_date>2000-10-01</publish_date>
    </book>
</catalog>
```

```
(base) flavius@flavius-IdeaPad-Gaming-3-15IMH05:~/Desktop$ ./XML2
Alegeți o opțiune:
1. Citește fișierul XML
2. Adaugă un tag
3. Șterge un tag
4. Ieși
2
Introduceti numele tag-ului pe care doriti sa-l adaugati:
carte
Introduceti valoarea pentru tag-ul carte:
Carte
Introduceti numele fisierului XML:
books.xml
Tag-ul <carte> a fost adăugat în primul tag din fișier.
```

La fiecare pas, utilizatorul este pus să introducă ceea ce dorește să modifice în fișierul XML specificat, așa cum se observă și în poze. Testele în acest proiect au vizat validitatea și corectitudinea manipulării fișierelor XML. De asemenea, s-a testat funcționalitatea adăugării de tag-uri, asigurându-se că nu sunt duplicate și că sunt plasate corect în structura XML, fără a afecta alte elemente.

4.3. Evaluarea performanței

```
flavius@flavius-IdeaPad-Gaming-3-15IMH05: ~/Desktop
books.xml
Conținutul fișierului XML:
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
  </book>
</catalog>

real    0m4.471s
user    0m0.002s
sys     0m0.007s
```

Timpul de execuție a fost măsurat cu *time*, monitorizarea utilizării CPU și RAM s-a făcut cu *top,* iar dimensiunile fișierului XML înainte și după modificare s-a realizat prin comanda –ls. S-a constatat că timpul necesar pentru citirea și formatarea fișierelor cu *xmlstarlet fo* a fost în medie de 0,2 secunde pentru fișiere mici și de 1-2 secunde pentru fișiere mari.

Adăugarea și ștergerea au fost, de asemenea, rapide, cu un timp mediu de mai puțin de o secundă pentru fișiere mici.

În testele cu fișiere XML de până la 10 MB, scriptul a reușit să adauge și să șteargă tag-uri într-un timp acceptabil (între 2 și 4 secunde). La testele de scalabilitate cu fișiere de până la 1000 de linii XML, scriptul

a rămas eficient. S-a observat un consum minim de CPU în timpul rulării scriptului, iar scriptul a utilizat aproximativ 50-100 MB de memorie RAM.

5. Concluzii

Proiectul demonstrează eficiența unui script pentru manipularea fișierelor XML, având o performanță bună pentru fișiere de dimensiuni medii. Au fost rezolvate majoritatea problemelor tehnice care puteau interfera cu buna funcționare a programului. Testele efectuate au arătat că scriptul poate fi utilizat pentru gestionarea fișierelor XML în diverse condiții nu foarte favorabile, însă pentru fișiere foarte mari ar fi necesare niște optimizări suplimentare.