UNIVERSITATEA DIN BUCUREȘTI FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Instrumente și Tehnici de Bază în Informatică

Proiect "XML"

XML_PARSER

Constantin Cristiana Georgiana

I. Scopul proiectului XML

Scrieți un program (script) shell care să parseze fișiere XML. Acesta trebuie să poată citi și scrie date din, respectiv, în format XML.

II. Introducere despre XML

XML (eXtensible Markup Language) este un limbaj de marcare folosit pentru a structura și stoca date într-un format ușor de citit și procesat.

1. *Utilizări principale:*

- Schimbul de date între aplicații ,baze de date și web services.
- Documente și fișiere de interschimb (de exemplu, SVG, MathML).

2. *Structura unui document XML:*

Un document XML are o structură ierarhică formată din elemente (tag-uri) care conțin date sau alte elemente. Fiecare document XML trebuie să aibă un element rădăcină care conține toate celelalte elemente.

3. *Caracteristici:*

- Structură ierarhică.
- Spre deosebire de HTML, XML permite utilizatorilor să definească propriile tag-uri, fiind astfel flexibil și adaptabil diverselor aplicații.

4. *Parser XML*

Un **parser XML** este un program care analizează un fișier XML și îl interpretează pentru a extrage date din acesta sau pentru a-l modifica. Un parser XML convertește un document XML într-o structură de date pe care aplicațiile o pot manipula ușor.

Dacă un program vrea să extragă numele unui student dintr-un fișier XML, un parser XML ar analiza documentul și ar returna valoarea conținutului din tag-ul <name>.

III. Rezolvarea cerintei

1. Cum functioneaza read_xml()

1. Argumente primite:
   - Numele fișierului XML ($FILE).
   - Numele tag-ului pe care dorim să-l citim ($TAG).
2. Fluxul funcției:
   - Verifică dacă tag-ul specificat conține doar un text simplu sau dacă este un tag mai complex (nested).
   - Pentru tag-uri simple: Extractează conținutul dintre <tag> și </tag> folosind sed si afișează valoarea găsită.
   - Pentru tag-uri complexe: Citește toate liniile dintre <tag> și </tag> folosind sed si reformatează structura pentru a afișa conținutul într-un format lizibil.

Exemplu de rulare și comportament :

Dificultăți întâmpinate la implementarea funcției read_xml

1. Gestionarea tag-urilor inexistente
2. Afișarea lizibilă a structurii (păstrarea indentării pentru structurile adânci.

3. Gestionarea tag-urilor nested

- Problemă: Tag-uri precum <album> conțin alte tag-uri, iar extragerea structurii complete era dificilă.
- Soluție: Utilizarea combinată a comenzilor sed pentru extragerea liniilor și reformatarea acestora într-un format ierarhic.

```
if [ -z "$content" ]; then
    lines=$(sed -n "/<$TAG>/,/<\/$TAG>/p" "$FILE" | sed 's:<\/[^>]*>::g' | sed 's/<//g' | sed 's/>/:
/g')
    echo "$lines" | sed "s/^$(echo "$lines" | head -n 1 | sed 's/^\( *\).*/\1/')//"
fi
```

4. Extragerea multiplă a valorilor pentru același tag

- Problemă: Tag-uri multiple cu același nume (ex. <song>) trebuiau afișate complet.
- Soluție: Folosirea sed -n pentru a itera prin toate aparițiile și afișarea fiecărei valori.

```
content=$(sed -n "s:.*<$TAG>\(.*\)</$TAG>.*:\1:p" "$FILE")
```

2. Cum functioneaza write_xml()

- Rolul funcției

Funcția write_xml permite modificarea sau adăugarea de noi elemente într-un fișier XML. Aceasta gestionează atât tag-uri simple (care conțin doar o valoare), cât și tag-uri complexe (nested), adăugând copii sub un tag părinte.

2. Argumente primite:

- Numele fișierului XML ($FILE).
- Opțiunea write: Indică faptul că funcția va scrie sau modifica elementele din fișier.Fluxul funcției:

3. Fluxul functiei:

- Verifică dacă tag-ul specificat conține doar un text simplu sau dacă este un tag mai complex (nested).

- Pentru tag-uri simple: Extractează conținutul dintre <tag> și </tag> folosind sed si afișează valoarea găsită.
- Pentru tag-uri complexe: Citește toate liniile dintre <tag> și </tag> folosind sed si reformatează structura pentru a afișa conținutul într-un format lizibil.

Exemplu de rulare și comportament :

<band>
<name>Måneskin</name>
<formationYear>2016</formationYear>
<members>4</members>
<album>Il Ballo della Vita</album>
</band>

```
cristiana@ubuntu:~/tema_ITBI/proiect$ ./proiect.sh file.xml write
Enter the number of child elements to add/update:
1
Enter name for child element #1 of band (add -delete to remove tag):
album
Enter value for album:
Il Ballo della vita
Updated existing tag: album
cristiana@ubuntu:~/tema_ITBI/proiect$ cat file.xml
<band>
    <name>Måneskin</name>
    <formationYear>2016</formationYear>
    <members>4</members>
  <album>Il Ballo della vita</album>
</band>
```

Adăugarea unei melodii într-un nou album:

```
cristiana@ubuntu:~/tema_ITBI/proiect$ ./proiect.sh file.xml write
Enter the number of child elements to add/update:
2
Enter name for child element #1 of band (add -delete to remove tag):
songs -nested
Enter the number of child elements for songs:
2
Enter name for child element #1 of songs (add -delete to remove tag):
song
Enter value for song:
Torna a Casa
Added new tag: song
Enter name for child element #2 of songs (add -delete to remove tag):
song
Enter value for song:
Moriro da Re
Updated existing tag: song
```

<band>
<name>Måneskin</name>
<formationYear>2016</formationYear>
<members>4</members>
<album>
<title>Il Ballo della Vita</title>
<songs>
<song>Torna a Casa</song>
<song>Morirò da Re</song>
</songs>
</album>
</band>

Dificultăți întâmpinate la implementarea funcției write_xml:

1. Adăugarea de tag-uri nested

- Problemă: Gestionarea tag-urilor nested (ex. <album> care conține <title> și <songs>) era complexă, necesitând o abordare recursivă pentru a adăuga copii sub un tag părinte.
- Soluție: Funcția write_children a fost implementată pentru a automatiza procesul de creare și gestionare a structurilor ierarhice.

Explicatie: Dacă utilizatorul specifică un tag nested (-nested), funcția verifică dacă tag-ul există deja.

Dacă nu există, este creat și apoi funcția recursivă write_children este apelată pentru a adăuga copii.

2. Menținerea indentării corecte

- Problemă: La adăugarea de noi elemente, indentarea trebuia să fie consistentă pentru a păstra lizibilitatea fișierului XML.
- Soluție: Utilizarea unor comenzi precise în awk și sed pentru a păstra formatul și indentarea la adăugarea sau actualizarea tag-urilor.

```
sed -i '/^$/d' "$FILE"
```

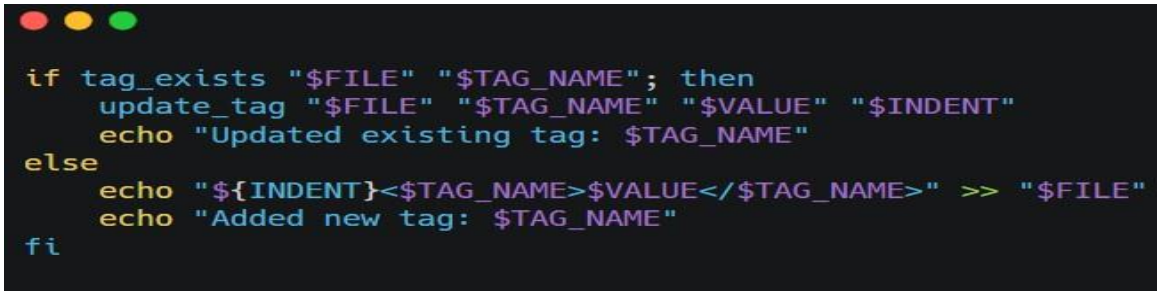
După adăugarea sau actualizarea tag-urilor, comanda sed curăță liniile goale pentru a menține un format consistent.

Indentarea este gestionată prin adăugarea spațiilor corespunzătoare în funcția write_children:

```
echo "${INDENT}<$TAG_NAME>$VALUE</$TAG_NAME>" >> "$FILE"
```

3. Gestionarea tag-urilor existente

- Problemă: Dacă tag-ul specificat de utilizator exista deja, funcția trebuia să actualizeze valoarea acestuia fără a afecta alte părți din fișier.
- Soluție: Funcția verifică existența tag-ului utilizând tag_exists. Dacă tag-ul există, valoarea este actualizată cu ajutorul unei comenzi awk sau sed.



```
if tag_exists "$FILE" "$TAG_NAME"; then
    update_tag "$FILE" "$TAG_NAME" "$VALUE" "$INDENT"
    echo "Updated existing tag: $TAG_NAME"
else
    echo "${INDENT}<$TAG_NAME>$VALUE</$TAG_NAME>" >> "$FILE"
    echo "Added new tag: $TAG_NAME"
fi
```

3. Cum functioneaza delete_tag()

Funcția delete_tag este responsabilă pentru ștergerea unui tag specific și a conținutului acestuia dintr-un fișier XML. Aceasta se asigură că ștergerea este completă și nu afectează alte părți ale fișierului.

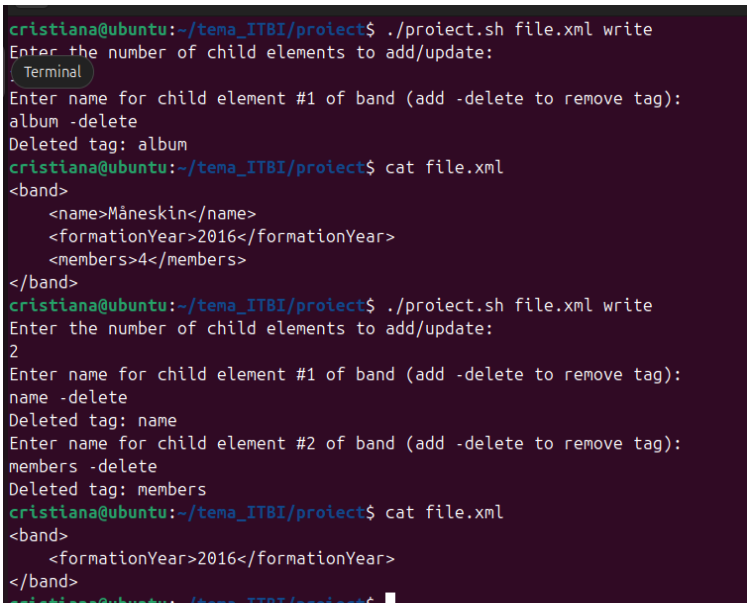
- **Argumente primite**

  1. Numele fișierului XML ($FILE): Fișierul în care se efectuează ștergerea.
  2. Tag-ul de șters ($TAG): Numele tag-ului care trebuie șters, inclusiv conținutul dintre <tag> și </tag>.

- Fluxul funcției

1. Identificarea tag-ului:
   - Funcția folosește awk pentru a găsi toate liniile dintre <TAG> și </TAG>.
2. Ștergerea completă:
   - Elimină atât tag-ul deschis și cel închis, cât și orice conținut dintre ele.
3. Curățarea fișierului:
   - După ștergere, fișierul este verificat pentru a elimina liniile goale și a menține un format consistent.

Exemplu de rulare și comportament :



Dificultăți întâmpinate la implementarea funcției delete_tag

1. Identificarea completă a tag-urilor nested:
    - Problemă: Tag-urile care conțineau alte tag-uri (nested) necesitau gestionare recursivă pentru a asigura ștergerea completă.
    - Soluție: Folosirea awk pentru a parcurge și șterge toate liniile dintre <TAG> și </TAG>.

```
delete_tag() {
    local file="$1"
    local tag="$2"

    # Create temp file
    temp_file=$(mktemp)

    # Delete only the specified tag and its contents
    awk -v tag="$tag" '
        BEGIN { skip=0; depth=0 }
        /<'$tag'>[^<]*<\/'$tag'>/ { next }  # Skip single-line tags
        /<'$tag'>/ { skip=1; depth=1; next }
        skip==1 && /<[^\/][^>]*>/ { depth++ }
        skip==1 && /<\/[^>]*>/ { depth-- }
        skip==1 && depth==0 { skip=0; next }
        !skip { print }
    ' "$file" > "$temp_file"

    mv "$temp_file" "$file"
}
```

2. Gestionarea liniilor goale:
    - Problemă: După ștergere, liniile goale din fișier afectau lizibilitatea.
    - Soluție: Curățarea fișierului cu comanda awk 'NF { print }'.

```
# Curățarea liniilor goale după ștergere
awk '
    NF { print }
' "$file" > "$temp_file"
mv "$temp_file" "$file"
```

3. Compatibilitatea cu tag-uri similare:
    - Problemă: Ștergerea unui tag specific putea afecta alte tag-uri cu nume similare.
    - Soluție: Utilizarea expresiilor regulate în awk pentru potrivire exactă

```
awk -v tag="$tag" '
    /<'$tag'>/ { skip=1; depth=1; next }
    skip==1 && /<\/'$tag'>/ { depth--; if (depth == 0) skip=0; next }
    !skip { print }
' "$file" > "$temp_file"
mv "$temp_file" "$file"
```

IV. Încheiere

Proiectul de creare a unui script shell pentru parsarea fișierelor XML a demonstrat că este posibil să gestionăm operațiuni complexe, precum citirea, scrierea și ștergerea tag-urilor, folosind unelte standard precum awk, sed și bash. Scriptul poate manipula atât structuri simple, cât și ierarhii complexe, fiind util în diverse scenarii.

Deși s-au întâmpinat dificultăți legate de gestionarea tag-urilor nested, păstrarea indentării și compatibilitatea cu tag-uri similare, acestea au fost rezolvate prin soluții eficiente.

V. Bibliografie

Documentația awk https://www.gnu.org/software/gawk/manual/gawk.html

Documentația sed https://www.gnu.org/software/sed/manual/sed.html

XML Standards https://www.w3.org/XML/

Tutorial Bash https://tldp.org/LDP/abs/html/

Acest proiect a fost realizat în echipă în cadrul laboratorului Instrumente si tehnici de baza in informatica .