

# Rapport projet XML

## Groupe 8

Introduction

Une analyse rapide de l'application

installation de l'application sous linux

**Utilisation**

Partie 1 - 3

Partie 4

Bilan

## Groupe 8

Pierre EBRAHIM

Swann HERRERA

## Introduction

La demande consiste en la réalisation d'une application avec le langage C, permettant de

verifier que le XML correspond à la DTD.

Ce travail a été réalisé par un groupe de deux personnes. Nous ne nous connaissions pas avant cette collaboration.

## Une analyse rapide de l'application

Il existe quelque fonction très importante dans l'application, tout d'abord le main qui est la fonction d'entré dans le programme, ce main fait appel a d'autre fonction qui sont aussi des fonctions de haut niveau

Parse DTD (**parse\_dtd**) : est la fonction qui va Créer un ensemble de structure pour modelisé la DTD. Cette Structure retient son parent et garde un tableau de ses enfants qui sont du même type qu'elle.

```

struct DTD_element
{
    DTD_element *parent;
    DTD_element **childs;
    DTD_attribute *attributes;
    char *name;
    int deepness;
    int childsCount;
    int childsCapacity;
    unsigned int numberOfAttribute;
    unsigned char occurrenceFlag;
    char occurrenceChar;
};

```

elle a aussi une liste chaîné d'attributs, on stock aussi le compte des attributs.

le caractère d'occurrence et le flag d'occurrence nous permettent de gérer l'occurrence des elements.

```

struct DTD_attribute
{
    DTD_attribute *next;
    AttributeType type;
    AttributeValue value;
    char *name;
};

```

Les Attributs de la DTD sont stocker dans une structure appeler **DTD\_attribute** qui est une liste simplement chaîné

```

enum AttributeValue
{
    NONE,
    REQUIRED,
    IMPLIED,
    FIXED
};

enum AttributeType
{
    VALUE,
    CDATA
};

```

Les valeur du type et de la valeur de l'attribut pour la dtd sont représenté par des **enums**

```
typedef struct XML_element
{
    char *name;
    unsigned int number_of_attribute;
    xml_attribute_linkedlist *attributes;
    char *content;
    struct XML_element *parent;
    struct XML_element **childs;
    int childs_count;
    int childs_capacity;
    int deepness;
    bool autoclosing;
} XML_element;
```

L'autre point important est la structure **XML\_Element**, elle contient les informations concernant la balise XML elle stocke aussi les attributs sous forme de liste chaînée.

elle a un niveau de profondeur, et une valeur booléenne pour gérer les balises autofermantes.

```
typedef struct xml_attribute
{
    char *name;
    char *value;
} xml_attribute;

typedef struct xml_attribute_linkedlist xml_attribute_linkedlist;

Swann, 23 days ago | 1 author (Swann)
struct xml_attribute_linkedlist
{
    xml_attribute *value;
    xml_attribute_linkedlist *next;
};
```




Cette fois pour les informations de l'attribut on utilise une autre structure.

La fonction qui parse le XML s'appelle **parse\_xml** elle fait appel à de nombreuses autres fonctions comme par exemple **make\_attribute**, **get\_content**...

Il existe une autre fonction très importante, c'est **check\_dtd\_correspond\_to\_xml** elle parcourt la structure XML et la structure DTD pour vérifier que le XML correspond bien à la DTD.

Afin de vérifier l'occurrence d'un enfant au sein d'un fichier XML on traduit les caractères d'occurrence avec un flag comme suit :

## Utilisation du flag d'occurrence

 Flag name	 Flag value(base2)	 Flag char
<u>OCCURENCE_1_1</u>	0b00	(NONE)
<u>OCCURENCE_0_1</u>	0b01	?
<u>OCCURENCE_1_N</u>	0b10	+
<u>OCCURENCE_0_N</u>	0b11	*

Ainsi lorsque le premier bit est à 0 on a forcément un élément et si il est à 1 on peut ne pas avoir d'élément. Également pour le second bit si il est à 0 on ne peut avoir de multiples enfants au contraire si il est à 1 on peut avoir une multitude d'élément.

Ce procéder permet dans le cas où l'on aurait une définition d'occurrence pour des enfants tel que

<!ELEMENT classrooms (classroom?)+> à l'aide d'opération binaire on peut donc assemblé les flags des char '?' et '+' qui nous donne le flag du char '\*'.

## installation de l'application sous linux

Ouvrir un terminal aller dans votre repertoire préféré

**git clone** <https://github.com/SwannHERRERA/xml-editor.git>

**cd** xml-editor

**./setup.sh** (utilisation d'APT donc de préférence utilisé une distribution debian ou basé sur debian)

**make build**

**make start**

## Utilisation

### Partie 1 - 3

**make build** construit un exécutable de l'application

**make start** lance l'exécutable celui ci a besoin d'un argument **inputFile** lui indiquant le fichier XML

sous forme d'un chemin soit relatif soit absolu

exemple

```
make build
make start inputFile=./xml-test/basic.xml
```

---

## Partie 4

### Make build

### Make start

ouvrir un fichier grâce a la barre d'action (la bar ALT)

Click droit → vérifié

## Bilan

Le fait que nous ne nous connaissions pas n'a pas facilité la prise de contact au début de notre travail.

Malgré ce confinement, la quasi totalité de notre projet a été réalisée grâce à la

disponibilité et la bonne entente de notre duo.

L'absence de rencontre physique de l'ensemble de la classe, due au confinement, nous a

privé de la co-formation et échanges qui nous enrichi toujours lorsque nous partageons notre travail.

Habituellement, cette rencontre naturelle, en présentiel, nous permet également d'avoir une

lecture des consignes et de se les confirmer.

Etant entrés trop rapidement dans le code et n'ayant pas analysé suffisamment en détail les

spécificités de DTD, nous avons réalisé tardivement que la gestion des types d'attributs

seraient aussi complexe.

Y revenir au moment où nous avons fait ce constat, nous aurait fait perdre trop de temps.

De ce fait, le type des attributs n'a pas été totalement géré.

La DTD exigeait qu'une valeur soit fixée. Contrairement aux attributs, nous en avons

conscience dès le départ. Néanmoins, devant la charge de travail nous avons fait le choix de ne pas fixer de valeur.

Notre impossibilité d'utiliser les bibliothèques externes ne nous facilite en rien l'accès aux DTD externes sur le réseau.

Par ailleurs, aller sur le réseau pour récupérer une DTD externe implique de gérer différents protocoles.

Ce sont les raisons pour lesquelles nous n'avons pas géré les DTD externes sur le réseau.

Sur le plan humain, mais aussi pédagogique, il nous semble important de préciser que ce travail nous a permis de faire connaissance, même si ce n'est que par écrans interposés.

Depuis septembre, le nombre de nouveaux arrivés dans cette classe est considérable, mais a aucun moment nous n'avons pu nous rencontrer à cause de la pandémie Covid 19.

Le fait d'avoir réalisé ce projet à deux, nous a permis de faire un peu connaissance, de se sentir solidaires et, de s'entraider même sur des aspects étranger à ce projet.