Université du Québec à Montréal

Département d'informatique Construction et maintenance de logiciel INF3135

T R A V A I L P R A T I Q U E #2

Emmanuel Chieze Automne 2012

Manipulation de fichiers XML

Objectif du travail

Utiliser des structures de données dynamiques et la programmation modulaire pour manipuler des fichiers XML.

Comme TP2 vous aurez à écrire un programme C sous Unix qui affiche la hiérarchie de balises contenue dans un fichier XML, et qui vérifie en même temps que le fichier est bien formé. L'architecture de l'application est fixée. Le gros du travail consistant à développer les modules utilisés par l'application principale, et non l'application principale elle-même, nous allons présenter les modules par ordre de complexité avant de présenter l'application à écrire.

Modules à développer

Module Chaine

Ce module implémente un type `Chaine` comme type abstrait opaque, selon l'interface `Chaine.h` fournie. La différence avec des chaînes de caractères C standard est que le type `Chaine` gère l'allocation dynamique de l'espace mémoire requis au stockage de la chaîne de caractères. Seules les opérations requises au développement de l'application sont incluses dans le module, mais il va de soi que l'on aurait pu rajouter à ce module de nombreuses autres opérations de manipulation de chaînes de caractères. En choisissant la représentation interne des données, vous devrez veiller à ce que l'appel des fonctions `chaineLongueur` et `chaineAjoute` se fasse de façon efficace.

Module Balise

Ce module implémente un type `Balise` comme type abstrait opaque, selon l'interface `Balise.h` fournie. Une balise XML peut être de cinq types :

1. **balise ouvrante** : `<nomBalise nomAttr1=val1 nomAttr2=val2>` : le nom de la balise est obligatoire, les attributs qui suivent sont optionnels
2. **balise fermante** : `</nomBalise>`
3. **balise ouvrante et fermante** : `<nomBalise nomAttr1=val1 nomAttr2=val2/>` : le nom de la balise est obligatoire, les attributs qui suivent sont optionnels

4. **balise de commentaires :** `<!....>`
5. **balise de directive :** `<?....>`

Vous supposerez que les caractères < et > sont réservés à la définition des balises dans la mise en œuvre du module. Par ailleurs, la liste des attributs est vue comme un tout par le module (on aurait pu concevoir le module autrement bien sûr). En choisissant la représentation interne des données, vous devrez veiller à ce que l'appel aux fonctions `baliseLit`... se fasse de façon efficace.

Module FichierBalises

Ce module implémente un type `FichierBalises` comme type abstrait opaque, selon l'interface `FichierBalises.h` fournie. Ce module gère la lecture séquentielle de fichiers XML, en fournissant les primitives appropriées, dont la principale est `fichierBalisesLit`, qui retourne la prochaine information disponible dans le fichier. Cette information peut être soit une balise, soit une zone de texte de taille maximale.

Exemple : soit un fichier `toto.xml` ayant comme contenu

```
<FIC><ELEM type="toto">
123
</ELEM> </FIC>
```

Des appels successifs à `fichierBalisesLit` retourneront :

1. la balise FIC
2. la balise ELEM avec l'attribut type de valeur toto
3. la zone texte /n123/n
4. la balise /ELEM
5. une zone texte constituée d'un espace
6. la balise /FIC

Le module n'est pas conçu pour gérer les pseudo-fichiers standards que sont `stdin`, `stdout` et `stderr`.

Test des modules

En plus de vous assurer que les modules vous permettent de développer l'application demandée, vous pourrez également vérifier qu'ils fonctionnent correctement avec l'application `extraireTexte` fournie, qui extrait le contenu textuel d'un fichier XML.

Application à développer

Après avoir développé les modules ci-dessus, vous devez développer une petite application ayant pour but d'extraire d'un fichier XML la hiérarchie des balises. Cela vous permettra en même temps de valider que le fichier est bien formé, à savoir que chaque balise ouvrante est bien associée à une balise fermante, sans enchevêtrement de balises.

Exemples :

- <BAL1><BAL2></BAL2></BAL1> est une séquence valide de balises.
- <BAL1><BAL2></BAL1></BAL2> n'est pas une séquence valide de balises.

Pour cela vous disposez en plus d'un module générique implémentant une pile sous forme de machine abstraite, fourni avec l'énoncé.

L'exécutable, qui s'appellera *extraitStructure*, prendra un seul argument, obligatoire, qui est le nom du fichier XML à analyser. La hiérarchie de balises sera envoyée sur la sortie standard. Chaque niveau d'imbrication sera représenté par deux espaces. Seules les balises ouvrantes seront représentées. En cas d'erreur de structure dans le fichier d'entrée, vous afficherez un message d'erreur approprié dès que vous détecterez l'erreur, et vous arrêterez alors le processus d'analyse du fichier. Il se peut que vous ayez déjà affiché le début du résultat lorsque vous détectez l'erreur.

Exemples d'exécution de l'application

Exemple 1 : fichier OK (exemple tiré de la page Web http://fr.wikipedia.org/wiki/XML_Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<personne xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personne.xsd">
  <nom>de Latour</nom>
  <prenom>Jean</prenom>
  <date_naissance>1967-08-13</date_naissance>
</personne>
```

La sortie associée à ce fichier est :

```
personne
  nom
  prenom
  date_naissance
```

Exemple 2 : fichier NOK

```
<?xml version="1.0" encoding="UTF-8"?>
<personne xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personne.xsd">
  <nom>de Latour  <prenom>Jean </nom> </prenom>
  <date_naissance>1967-08-13</date_naissance>
</personne>
```

La sortie associée à ce fichier est :

```
personne
  nom
    prenom
Fichier XML ou HTML mal forme
```

Un troisième exemple plus complet de fichier XML, issu de la page Web http://fr.wikipedia.org/wiki/MathML, est disponible en pièce jointe avec sa sortie.

Remise

Vous devez remettre votre code (le source de vos modules et du programme principal et un *makefile*) ainsi que les fichiers de données utilisés pour vos tests. Vous ne devez pas remettre vos fichiers objets ni vos exécutables. Il ne faut pas modifier les fichiers **.h* fournis, ni le fichier *pile.c*, car ils seront remplacés par leur version originale avant compilation.

Un appel *make* générera un exécutable du nom de *extraitStructure*. Un appel *make clean* nettoiera le répertoire de travail en supprimant tous les exécutables et tous les fichiers objets.

Pour remettre vos fichiers, supposés tous résider dans le répertoire tp2, vous :

- Vous placerez dans le répertoire tp2
- Vous exécuterez `make clean`
- Vous archiverez le répertoire avec la commande tar (ce qui donne un fichier tp2.tar) :

  `tar cvf tp2.tar *`
- Vous compresserez l'archive avec la commande gzip (ce qui donne un fichier tp2.tar.gz) :

  `gzip tp2.tar`
- Vous préfixerez le résultat de vos codes permanents :

  `TOTO2004200001_TITI2001200002_tp2.tar.gz`
- puis vous remettrez le résultat sur Moodle.

Vous remettrez de plus un rapport papier, contenant :

- chacun des fichiers .c développés,
- le code des modules additionnels que vous auriez pu ajouter, le cas échéant.

Votre dossier papier doit contenir une page de garde donnant vos coordonnées (noms, prénoms, codes étudiant), et doit être broché. Inutile en revanche d'investir dans de coûteuses reliures ou couvertures cartonnées.

Évaluation

Votre TP sera évalué selon les critères suivants :

- **Fonctionnalité (50 pts)** : Compilation et fonctionnement correct du code, ce qui suppose entre autres que votre `makefile` est correct. ATTENTION : vos modules peuvent aussi bien être testés avec votre version de l'application qu'avec la mienne ou qu'avec l'application extraireTexte fournie.
- **Structure de l'application et bonne programmation (50 pts)** : Utilisation de structures de données et de contrôles appropriées. Découpage approprié en modules et en fonctions le cas échéant. Disposition du code, pertinence des commentaires, choix d'identificateurs significatifs. Gestion des cas d'erreur et utilisation d'assertions et de programmation défensive.

Vérification:

- Le programme remis doit se compiler sur rayon1 avec *make* sans générer d'avertissements (ni de messages d'erreurs évidemment), et doit générer un exécutable *tp2*.

- Le TP se fait en équipe de deux étudiants. Les noms des membres de l'équipe doivent aussi être indiqués au début de tous les fichiers sources.

Date de remise

Date de remise en classe du dossier papier (la remise électronique devant avoir été faite avant) : **mardi le 18 décembre 2012 au début du cours.**