TP N°2 - Arbre n-aire de mots

Préparé pour : Bapstite Emery

Préparé par : Benjamin DEBOTTÉ,, 1A App.Info

20 mars 2015

SOMMAIRE

I. CRÉATION DE L'ARBORESCENCE DE TRAVAIL

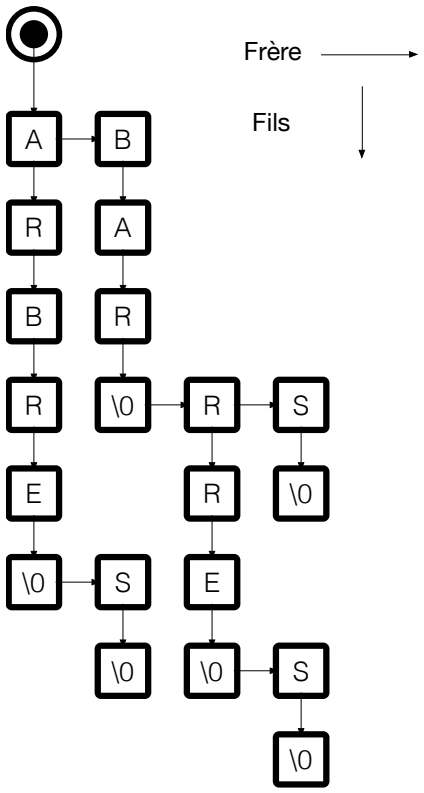
Ce TP a été développé avec les outils VIM/GCC/GDB+TUI sous Mac OS X

```
TP3:
ArbreNAire      Makefile        dico.ang        dico.fr
nth_word_tree.c nth_word_tree.h
```

- nth_word_tree.c/h : fichiers sources des fonctions pour manipuler les arbres n-aire de mots
- Makefile : directives de compilation de l'outil Make
- ArbreNAire : Résultat de la compilation
- dico.ang/fr : Fichiers texte contenant des listes de mots propres à la langue en lien avec l'extension de ce fichier.

II. CONCEPTION

Ce TP a été réalisé par le développement de fonctions de gestion d'Arbre n-aire de mot. Bien que proche d'un arbre binaire du TP précédent, un Arbre n-aire peut être utilisé comme ici pour réaliser un « Arbre de complétion » pour construire des relations entre des lettres, représentées par des noeuds, pour former des mots au fil de son parcours. Ci-contre un exemple formé des mots {Arbre, Arbres, Bar, Bars, Barre, Barres}



La difficulté principale rencontrée ici fût d'appréhender l'arbre binaire de recherche et de bien comprendre les mécaniques préfixe/infixe/suffixe.

La réalisation a consisté en un développement d'un ensemble de fonction permettant l'initialisation, la construction, la recherche et l'affichage de ces arbres.

1. Initialisation

L'initialisation d'un arbre n-aire consiste à initialiser la racine principale, qui contiendra tout les éléments du premier niveau, via une allocation mémoire.

2. Construction

À partir de la racine, il est possible d'ajouter des mots qui seront ajoutés lettre par lettre dans l'arbre de manière récursive. Plusieurs cas ont été identifiés lors du traitement de l'ajout d'un noeud :

- L'arbre est vide : on ajoute directement le noeud en tant que fils du noeud principal et on retourne son adresse.
- Il existe au moins un fils: on vérifie que le noeud est présent parmi la liste composée du fils actuel et de tout ses frères. On conserve l'adresse du dernier noeud dont la valeur est inférieure à la valeur recherchée initialisée à NULL nommé 'beforeNode'.
    - On vérifie que le noeud est présent parmi la liste composée du fils actuel et de tout ses frères. S'il est présent, on retourne l'adresse de ce noeud.
    - Sinon on vérifie s'il faut insérer en tête ou entre deux noeuds. Si 'beforeNode' vaut NULL, le noeud à ajouter doit être mis en tête en tant que fils du noeud principal . Sinon, on ajoute le nouveau noeud en tant que frère de beforeNode et ayant pour frère celui de beforeNode.

  On rappelle ensuite la fonction avec le fils du noeud ajouté. Si cependant le caractère est le marqueur de fin de chaîne, on arrête le parcours et on retourne l'adresse du noeud.

3. Recherche

La recherche consiste à chercher étape après étage ,dans la liste du fils et de ses frères du noeud actuel, la valeur du noeud égale à la valeur recherchée. Si aucun noeud n'est trouvé, alors la fonction retourne faux, sinon elle rappelle la fonction avec le fils du noeud trouvé.

III. CONCLUSION

Ce TP permet d'illustrer le fonctionnement d'une implémentation de recherche et complétion automatique grâce à des arbres n-aire. Le compte d'allocation permet par contre de se rendre compte qu'un grand nombre d'allocations mémoires est utilisé en comparaison avec le nombre de mot contenu par le dictionnaire :

```
MBP-de-Benjamin:TP3 benjamindebotte$ ./ArbreNAire
Fichiers chargés avec succès (479539 allocations <=> 11239 kB).
Mot à rechercher : ^C
MBP-de-Benjamin:TP3 benjamindebotte$ wc -l dico*
   45370 dico.ang
   87051 dico.fr
  132421 total
```

On pourrait imaginer essayer d'initialiser qu'un seul noeud par valeur dont l'adresse serait pointée par tout les noeuds y faisant référence, plutôt qu'en recréer un à chaque recherche infructueuse au sein des frères.