Département IMA / 3A (S6) / 2013 **Programmation Avancée**
Jonathan Aceituno, Vincent Aranéga, Nathalie Devésa, Rachid Elkouri et Laure Gonnord

TP7 Bibliothèque de listes chaînées de chaînes

Objectifs

- Savoir construire une bibliothèque à partir de code existant.
- Utiliser les fonctions de `string.h` (comparaison, copie).
- Connaître les options de gcc pour la compilation séparée.
- Savoir construire un Makefile.

Contexte et préparation Dans ce TP, nous allons récupérer du code existant pour construire une bibliothèque de listes que nous utiliserons au TP suivant (TP8). Cette bibliothèque contiendra les fonctions de base sur les *listes chaînées de chaînes de caractères, triées!*. Les chaînes de caractères des listes auront une taille maximale `MAXSIZE` fixée à 30.

1 Questions du TP (à faire impérativement)

Avant de commencer, récupérer sur le compte Polytech de Laure Gonnord le fichier `listechaines.h` situé dans le répertoire :

`/home/imaEns/lgonnord/PA2013/TP7/`

et placez le dans un répertoire nommé TP7.

1.1 Copier/coller rapides et ajustements

À partir des codes dont vous disposez, et en testant au fur et à mesure dans une fonction `main` (temporaire), créer un fichier `listechaines.c` conforme au `.h` fourni :

1. Inclusion de `listechaines.h` (avec la directive de preprocessing `#include`).
2. Les déclarations des types cellule, pointeur de cellule, liste de chaînes de caractères sont faites dans le `.h`, donc vous n'avez pas de déclaration à faire.
3. Récupérer et adapter à partir des tps précédents l'implémentation des fonctions d'ajout et suppression en tête, de taille, d'appartenance à la liste. **Pour l'appartenance, on pourrait utiliser le fait que les listes sont triées ici, mais on se satisfera de la version récursive du TP6**. Vous aurez besoin des fonctions `strcpy` et `strcmp` de la bibliothèque `string`.
4. Pour le chargement à partir d'un fichier, et l'insertion dans l'ordre alphabétique, on vous fournit deux fonctions déjà écrites sur la page Web du cours. Récupérer le fichier à la ligne de commande avec `wget` (récupérer l'adresse au clic droit).
5. Vérifier toutes vos fonctions en créant un fichier de noms et en chargeant les informations qui y sont incluses (à partir du `main`).

1.2 Création de bibliothèque

La création de la bibliothèque est quasiment terminée, à ce stade :

- Le .h contient les déclarations de types, des constantes symboliques, les inclusions de bibliothèque et les **entêtes des fonctions**.
- le .c contient l'implémentation de ces fonctions, et un main.

Déplacer le main de `listechaines.c` et le placer dans un fichier nommé `tp7.c`.

1.3 Compilation séparée, librairie statique

Vous vous reporterez au cours de Programmation avancée pour bien comprendre les lignes de commande que vous tapez.

Compilation séparée au sein d'un même projet

1. Compiler les deux `.c` précédents (avec -c) et les lier en compilant les deux .o ensemble :

   ```
   clang -c tp7.c -I.
   clang -c listechaines.c
   clang -o tp7 tp7.o listechaines.o
   ```

2. À quoi sert l'option `-I` de clang ? (man clang, ou le cours). Notez qu'ici il est facultatif (pourquoi ?)
3. Lancer le programme et tester.
4. Si vous modifiez `tp7.c`, quelles lignes de compilation/liaison devez vous faire ?

Dans ce cas, vous avez utilisé les fonctions écrites dans listechaines.c comme des fonctions "normales" de votre code. Dans la suite, on va compiler une librairie qu'on peut éventuellement distribuer.

Compilation d'une librairie et utilisation

1. Compiler la librairie `listechaines` (en statique) de façon à obtenir `liblistechaines.a` :

   ```
   clang -c listechaines.c
   ar rcs liblistechaines.a listechaines.o
   ```

   La librairie est compilée séparément de votre programme. La deuxième ligne n'effectue pas grand chose de plus qu'une compression du .o. Il reste à lier votre programme avec celle-ci

2. Compiler `tp7.c` en liant cette librairie statique (options -I et -L) :

   ```
   clang -c tp7.c  -I.                      //compilation
   clang -o tp7 tp7.o -L. -llistechaines    //liaison
   ```

   ou, plus simplement (compilation et liaison en même temps)

   ```
   clang -o tp7 tp7.c -L. -llistechaines
   ```

3. À quoi sert l'option -L ? tester la ligne de commande précédente sans cette option, et aussi sans `-llistechaines`, et soigneusement noter les messages d'erreur.

4. Lancer le programme et tester.

5. Si vous modifiez `tp7.c`, quelles lignes de compilation/liaison devez vous faire ?

IMPORTANT ! Le TP8 aura besoin de cette bibliothèque !

2 Questions s'il vous reste du temps

1. Vérifier que vous pouvez utiliser la librairie du voisin (copier son .a dans le répertoire courant).

2. Observer le contenu des binaires `tp7.o` et `listechaines.o` à l'aide de la commande unix `nm`.

3. Construire le (un !) Makefile pour la compilation séparée avec création de librairie statique.

Annexe : fichier d'entête `listechaines.h`

```
#include<stdio.h>
#include<stdbool.h>
#include<stdlib.h>
#include<string.h>

#define MAXSIZE 30

//Declaration de liste chainee de chaines de caracteres
typedef struct cell {
  char val[MAXSIZE];
  struct cell * suiv;
} Cellule, *Ptcellule, *Liste;

//Affichage de la liste en ligne
void afficherListe(Liste l);

//Ajout d'un mot en tete de la liste
void ajoutTete(Liste *pl, char mot[MAXSIZE]);

//Suppression du mot en tete de la liste
void suppTete(Liste *pl);

//Ajout un mot dans une liste supposee
// triee dans l'ordre alphabetique
void ajoutAlphab(Liste *pl, char mot[MAXSIZE]);

//Dit si un mot donne est dans la liste
//pas forcement triee
bool appartient(Liste l,char mot[MAXSIZE]);

//Donne la taille de la liste.
int taille(Liste l);

//construit une liste triee a partir d'un fichier
void chargeFichier(FILE *fp, Liste *pl);

//Destruction de Liste.
void detruireListe(Liste* l);
```