Projet d'Introduction au Système
Licence $2^{ème}$ année

Gilles Roussel

9 avril 2015

À réaliser seul.
`http://http://igm.univ-mlv.fr/ens/Licence/L2/2012-2013/System/projet.html`
À rendre avant le **22 mai 2015**, en envoyant un mail contenant l'archive de votre projet à votre chargé de TD : Sylvain.Cherrier@u-pem.fr ou Michel.Landschoot@u-pem.fr.

Le sujet est susceptible d'être modifié. Dans ce cas, les modifications seront clairement marquées par un changement de couleur. Il est donc conseillé de relire cette page régulièrement.

L'objectif de ce projet est d'écrire un interpréteur de commandes de manipulations listes de chaînes de caractères extensible par *plugin*. Le programme devra fournir une interface texte (en ligne de commande) capable de charger dynamiquement les *plugins* correspondants aux commandes tapées par l'utilisateur et leur passer les arguments.

1 Le mécanisme de plugin

Les plugins sont contenus dans une ou plusieurs bibliothèques partagées (`.so`). Au démarrage, l'application recherchent ces bibliothèques dans un ensemble de répertoires. Ceux-ci sont précisés dans un fichier `~/.aloha.cfg` ou par la variable d'environnement `ALOHA_PATH` sous la forme d'une listes de répertoires séparés par des « : ».

Une bibliothèque de plugin contient une méthode `char **getPlugins()` qui retourne l'ensemble des noms de plugins de la bibliothèque sous la forme d'un tableau de chaînes de caractères, terminé par `NULL`. Ces noms correspondent à des noms de fonctions (qu'on appellera maintenant plugins) présents dans la bibliothèque.

Au démarrage l'application récupère l'ensemble des noms de plugins disponibles dans les bibliothèques et les stockent dans une structure de données adaptée à la recherche d'un plugin en fonction de son nom (par exemple un arbre lexicographique, une table de hachage ou une liste triée). Au démarrage, les fonctions correspondant aux plugins, ne sont pas chargées (par `dlsym()`). Elles sont chargées au moment de leur première utilisation par une fonction de chargement dynamique.

Chaque plugin prend en argument les arguments de la ligne de commande sous la forme d'un tableau de chaînes de caractères (type `argv`) ainsi qu'une pile de listes de chaîne de caractères. Il retourne une nouvelle pile, éventuellement modifiée.

Le prototype d'un plugin est `pile plugin(char** argv, pile p)`. La pile sera représentée par une liste chainée de listes de chaînes de caractères. Le type de la liste de chaînes de caractères est `liste`. Les types `pile` et `liste` sont définis comme suit :

```
typedef struct l {
  char* s;
  struct l* suivant;
} *liste;

typedef struct p {
```

```
  liste l;
  struct p* suivant;
} *pile;
```

Une pile vide ou une liste de chaînes de caractères vide est représentée par `NULL`.

Vous veillerez dans vos plugins à libérer, si nécessaire, la mémoire liée à la pile ou aux listes.

2 Les plugins à implanter

Les plugins suivants devront être implantés et être placés dans plusieurs bibliothèques :

- `list a b c` qui ajoute en sommet de pile la liste de ces arguments (`"a" "b" "c"`) ;
- `dup` qui duplique la liste en sommet de pile et replace la liste et sa copie en sommet de pile ;
- `concat` qui retire les deux listes en sommet de pile, les concatène (la liste en sommet de pile au début) et replace le résultat en sommet de pile ;
- `print` qui affiche le contenu de la liste en sommet de pile, sans la retirer de la pile ;
- `file f` qui lit le fichier de nom `f` et ajoute en sommet de pile la liste des lignes du fichier ;
- `printstack` qui affiche le contenu de la pile ;
- `cat` qui affiche le contenu des fichiers dont les noms sont en sommet de pile et supprime le sommet de la pile ;
- `tr "a" "b"` qui remplace dans la liste en sommet de pile tous les `"a"` par des `"b"` ;
- `grep .*\.java` qui supprime de la liste en sommet pile tous les éléments qui contiennent pas une chaîne de caractères correspondant à l'expression rationnelle passée en argument (tous les éléments ayant pour suffixe `.java`). On utilisera pour ce plugin les fonctions `regex` de la bibliothèque C ;
- `dir` qui retire le sommet de la pile, le considère comme une liste de noms de répertoire et qui replace en sommet de pile la liste des noms de fichiers de ces répertoires ;
- `recdir` qui effectue la même chose, mais récursivement ;
- `tocommand c` qui exécute une commande shell `c` avec ses arguments en lui passant sur l'entrée standard la liste en sommet de pile sous forme de lignes. Par exemple, si le sommet de la pile contient `aa bb ca`, alors `tocommand grep a` affichera deux lignes, la première contenant `aa` et la seconde `ca` ;
- `fromcommand c` qui exécute une commande shell `c` avec ses arguments et place en sommet de pile la liste des lignes retournées par la commande. Par exemple, `fromcommand echo a b` placera en sommet de pile un liste contenant l'unique élément `"a b"`.

3 Évaluation

Le respect des noms de fichiers et de fonctions précisés dans ce sujet sont un aspect important de l'évaluation finale du projet car ceux-ci seront testés de façon automatique.

Le choix et l'utilisation d'une structure de données adaptée à la recherche des plugins seront pris en considération.

Une attention particulière devra être portée à la détection et à la gestion des erreurs qui pourront se produire lors de l'utilisation de votre bibliothèque.

Votre projet sera à rendre sous la forme d'une archive compressée `tar.gz` contenant : les sources dans un répertoire `src`, les bibliothèques dans un répertoire `lib`, les commandes dans un répertoire `bin` et une documentation de 5 pages maximum sur votre projet au format `PDF` dans un répertoire `doc`.