

Licence informatique (CDA, IFA, II, INT)

Année 2018-2019

Projet PCA "follow"

Stéphane Rubini

23 octobre 2018

Le but du projet est de développer un logiciel qui sera baptisé *follow*, pour le suivi *a posteriori* des modifications documentaires. Ce logiciel a pour objectif d'aider à la comparaison de deux textes, normalement relativement proches, en faisant apparaître leurs différences. Un alignement des deux textes est recherché afin de minimiser le nombres d'opérations à réaliser pour passer d'un texte à l'autre. Par opération, on entend des opérations d'édition de type *remplacement*, *insertion* ou *suppression* de mots. Le logiciel ne fait pas apparaître les modifications de la mise en page (retour à la ligne, blanc, tabulation). En revanche, la ponctuation et les caractères spéciaux sont pris en compte.

La figure 1 montre l'interface graphique du logiciel.

Les fonctionnalités du logiciel sont les suivantes :

- Chargement du texte à comparer au texte de référence, qualifié de "nouveau" par le suite, et affichage dans la zone de texte à droite de la fenêtre. Le nom du fichier chargé apparaît en dessous de cette zone.

Le nom du fichier est fourni sur la ligne de commande (second argument) au moment du lancement du logiciel, ou obtenu à partir d'un sélecteur de fichiers déclenché par le menu déroulant *Fichier*, option *Ouvrir*.

- Chargement du texte de référence.

Si le nouveau texte n'est pas encore chargé, alors le texte de référence est affiché tel quel dans la zone de texte à gauche de la fenêtre. Dans le cas contraire, les différences avec le nouveau texte sont affichées dans la zone de texte à gauche par une présentation en couleur :

- un mot affiché en rouge apparaît dans la référence, mais a été supprimé dans le nouveau texte,
- un mot affiché en vert a été inséré dans le nouveau texte,
- un couple de mots affiché en rouge et bleu indique le remplacement du mot rouge dans le texte de référence par le mot bleu dans le nouveau texte.

La mise en page est celle du nouveau texte. Le nom du fichier de référence apparaît en dessous de la zone de texte.

Le nom du fichier est fourni sur la ligne de commande (premier argument) au moment du lancement du logiciel, ou obtenu à partir d'un sélecteur de fichiers déclenché par le menu déroulant *Fichier*, option *Ouvrir référence*.

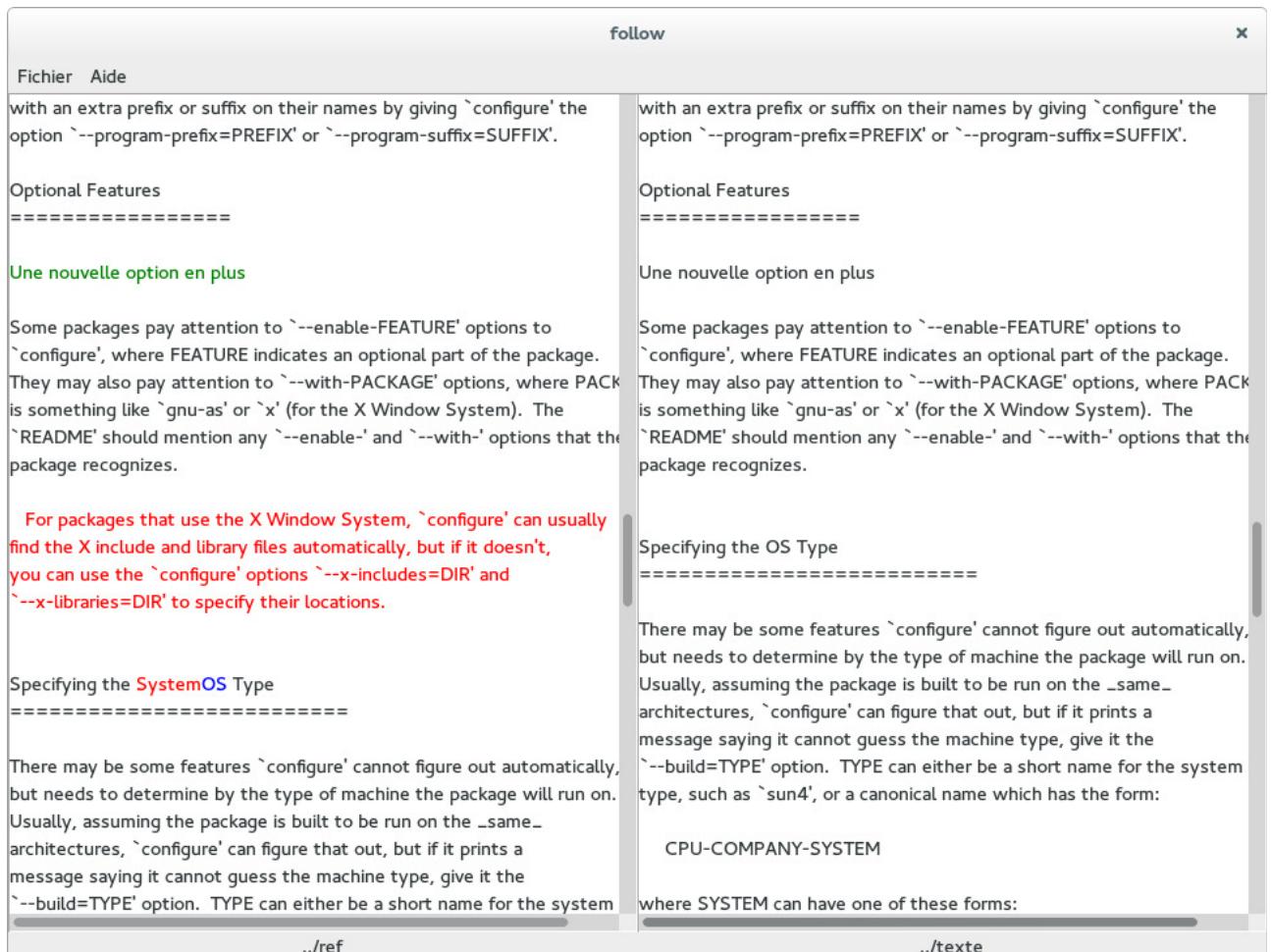


FIGURE 1 – Interface graphique du logiciel follow

- Affichage d'une fenêtre "A propos" qui décrit le nom, la version et l'auteur du logiciel. Son affichage est déclenché par le menu déroulant *Aide*, option *À propos*. La fenêtre est fermée par l'activation d'un bouton *OK*.
- Arrêt du logiciel, déclenché par le menu déroulant *Fichier*, option *Quitter* ou par un clic sur le bouton de destruction de la fenêtre graphique.

Planning prévisionnel, organisation et notation

Le planning prévisionnel de déroulement du projet est décrit dans le tableau ci-dessous.

Séance (2H)	Taches	Durée
1 et 2	Présentation du projet Développement et test du module <i>liste</i>	30 min 3H
3 et 4	Table de hashage (cours) Développement et test du module <i>strhash</i>	30 min 3H30
5 et 6	Construction de la représentation interne des textes Algorithme de recherche des PLSC (cours) Recherche de l'alignement des textes	1H 30 min 4H00
7 et 8	GTK, présentation Conception de l'IHM	30 min 2H
9, 10 et 11 12	Intégration de l'ensemble Démonstration finale	6H 2H
	Total	24H

Les trois premières étapes de développement seront validées par un test unitaire.

De plus, chaque étape fera l'objet d'une évaluation par l'enseignant. Le passage à l'étape suivante du développement est conditionné à une évaluation positive de la part de ce dernier. Il est précisé que vous devez présenter à cette occasion une mise en œuvre que vous avez vous-même développée.

Pour les 2 premières étapes uniquement, et dans le cas où vous considérez que vous y avez consacré trop de temps, il est possible "d'acheter"¹ la mise en œuvre du module à votre enseignant. Cet achat aura évidemment un coût au niveau de la notation de l'étape correspondante.

1. symboliquement bien entendu !

Séance 1

Module Liste Le module liste gère une liste simplement chaînée. Les données enregistrées dans les noeuds sont des pointeurs génériques (`void *`). La gestion mémoire des objets pointés n'est prise en charge dans le module.

Le module devra apporter les fonctionnalités minimales suivantes :

- création d'une liste,
- insertion d'une donnée en tête de liste,
- ajout d'une donnée en queue de liste,
- suppression d'une donnée en tête de la liste,
- suppression de la première instance d'une donnée dans la liste,
- application d'une fonction sur les données de la liste (la fonction, passée en paramètre, prend en argument l'adresse du nœud courant et un paramètre sans signification prédéfinie, et retourne un booléen qui conditionne la suite du parcours dans la liste),
- insertion d'une donnée dans une liste ordonnée (une fonction fournie en paramètre est appelée pour comparer deux données),
- destruction d'une liste complète.

L'interface publique du module est donnée ci-dessous.

```
typedef struct _list_node s_node;
s_node * list_create(void);
    /* creation d'une nouvelle liste vide
void * list_get_data(void s_node * node);
void list_set_data(s_node * node, void * data);
    /* lire ou ecrire la donnee d'un noeud
s_node * list_insert(s_node * head, void * data);
    /* creation et insertion d'un noeud en tete de liste
    /* retourne la tete de liste
s_node * list_append(s_node * head, void * data);
    /* creation et ajout d'un noeud en queue de liste
    /* retourne la tete de liste
int list_process(s_node * head, int (*fct)(s_node * node,
    void * param), void * param, s_node ** last);
    /* Application d'une fonction sur les donnees enregistrees
    /* dans le liste. last est le dernier noeud traite'
    /* retourne 1 sur le parcours est arrete' avant la fin de la liste
s_node * list_remove(s_node * head, void * data);
    /* suppression de la premiere instance d'une
    /* donnee dans la liste, retourne la tete de liste
s_node * list_headRemove(s_node * head);
    /* suppression de la tete de liste
    /* retourne la nouvelle tete de liste
void list_destroy(s_node * head);
    /* destruction d'une liste
    /* (La liberation des donnees n'est pas prise en charge)
```

Séance 2 : table de hachage

Table de hachage Le module *strhash* gère une table de hachage. Les données enregistrées dans la table sont des chaînes de caractères. Un seul exemplaire de chaque chaîne est enregistré dans la table. Le nombre d'entrées dans la table est fixé lors de sa création. Les chaînes associées à chaque entrée sont enregistrées dans une liste simplement chaînée, et sont triées par ordre alphabétique.

Le calcul de la clef de hachage est implanté dans une fonction privée du module, dont l'algorithme est montré ci-dessous.

```
clef ← 0
pour chaque lettre du mot à "hacher"
    clef ← clef × 2
    clef ← clef + code ASCII de la lettre
fin pour
clef ← reste(clef, nombre d'entrées dans la table)
```

Le module *strhash* devra apporter les fonctionnalités minimales suivantes :

- création d'une table de hashage,
- destruction d'une table de hashage,
- libération des données enregistrées dans une table de hashage,
- ajout d'un mot dans la table si le mot concerné ne s'y trouve pas déjà,
- suppression d'un mot dans la table
- affichage de données statistiques sur la table de hashage (nombre total d'éléments, nombre minimum et maximum d'éléments par entrée, écart type du nombre d'éléments par entrée).

Séances 3 et 4 : chargement, analyse et alignement des textes

Chargement des textes Les textes à comparer sont chargés à partir de fichiers par des opérations d'entrées/sorties haut niveau, et enregistrés dans deux tableaux de caractères. Les tailles des fichiers ne sont *a priori* pas contraintes.

Travail à effectuer Écrire et tester la fonction permettant de charger un texte en mémoire.

Analyse des textes Une fois chargé, un texte est analysé pour le découper en jetons. Un jeton représente soit :

- un mot (type WORD), c'est à dire un groupe de caractères autres que des délimiteurs qui se suivent,
- une "courte" zone blanche (type SHORT_SPACE), c'est à dire un groupe d'au plus 4 caractères délimiteurs (espace, tabulation ou retour à la ligne) qui se suivent,
- une zone blanche (type SPACE), c'est à dire un groupe de plus de 4 délimiteurs qui se suivent.

Les chaînes de caractères associées à chaque jeton sont enregistrées dans une table de hachage. La même table est utilisée pour les mots des deux textes. La distinction entre les zones blanches courtes et longues devrait permettre d'améliorer les performances du logiciel.

Chaque jeton est positionné dans le texte d'origine par son décalage par rapport au début de celui-ci (*offset*).

En langage C, un jeton est défini par la structure suivante :

```
struct token {
    enum { WORD, SHORT_SPACE, SPACE, ERASE,
           INSERT, REPLACE, EMPTY } type;
    int textOffset;
    union {
        char * word;
        char space[4];
    } data;
};
```

Les types ERASE, INSERT, REPLACE et EMPTY seront utilisés par la suite pour représenter les différences entre les textes.

Les jetons seront référencés par des tableaux de pointeurs, dont la taille sera ajustée dynamiquement en fonction des besoins (cf la fonction `realloc()`).

Travail à effectuer Écrire et tester le code qui permet de découper le texte en unités lexicales WORD, SPACE ou SHORT_SPACE. Les jetons qui les décrivent sont enregistrés dans des tableaux.

Alignement des textes Lorsque les deux textes sont chargés, le logiciel recherche le meilleur alignement possible des deux textes. Un algorithme qui détermine la *plus longue sous-séquence commune (PLSC)* est utilisé. Son fonctionnement est décrit ci-dessous.

Soient A_i , $i = 1, \dots, m$ les m mots du texte de référence et B_i , $i = 1, \dots, n$ les n mots du nouveau texte. On note $lg(i, j)$ la longueur de la PLSC qui peut être obtenue à partir des i premiers mots de A et des j premiers mots de B . $lg(i, j)$ peut être calculée à partir de l'équation de récurrence qui suit :

$$\begin{aligned} lg(i, 0) &= 0 \\ lg(0, j) &= 0 \\ lg(i, j) &= \begin{cases} lg(i - 1, j - 1) + 1 & \text{si } A_i = B_j \\ \max(lg(i - 1, j), lg(i, j - 1)) & \text{sinon} \end{cases} \end{aligned}$$

À partir de la matrice des longueurs ainsi construite, il est possible de déterminer la PLSC en partant de $lg(n, m)$, et en cherchant par un chaînage arrière les opérations d'insertion, de suppression et de remplacement qui conduisent à la détermination de la PLSC.

Travail à effectuer Mettre en œuvre l'algorithme de calcul de la longueur de la PLSC.

Ensuite, construire un nouveau tableau de jetons qui contient les noms en commun, les mots remplacés, supprimés ou insérés, et les espaces. Les types de jeton **ERASE** (mot supprimé), **INSERT** (mot inséré), et **REPLACE** (mot remplacé) sont utilisés, en complément avec les autres types précédemment définis, pour caractériser les jetons. Le jeton **EMPTY** est disponible si la présence de jeton vide permet de simplifier la mise en œuvre de l'algorithme.

Séance 4 (suite)

Une très brève introduction à GTK+ L'acronyme GTK+ réfère à un ensemble de fonctions regroupées au sein de trois bibliothèques de fonctions :

- la bibliothèque *glib*, qui apporte des fonctionnalités de bases pour la manipulation des structures de données (gestion de listes, tables de hashage, ...),
- la bibliothèque *GDK*, qui permet de dialoguer avec le gestionnaire d'environnement graphique X-Window ou Xorg,
- la bibliothèque *GTK* qui propose un ensemble d'objets graphiques dont la composition permet de créer une interface utilisateur graphique complète.

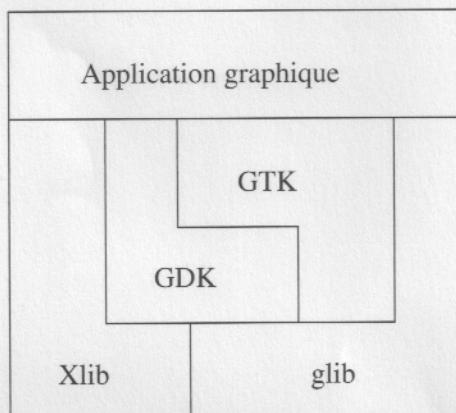


FIGURE 2 – Structuration de GTK+. Les applications peuvent faire appel aux services apportés par les bibliothèques *glib*, *GTK*, *GDK* et *Xlib*. La *Xlib* contient les fonctions de bas niveau pour interagir avec un serveur graphique.

Une application graphique est construite par la composition de *widglets*². Les widglets sont représentés à l'écran par des objets graphiques, avec lesquels l'utilisateur peut interagir.

Les actions de l'utilisateur sur les *widglets* se manifestent par l'émission d'événements logiciels qui pourront être captés par l'application. Ces captures se traduisent par l'exécution d'une fonction attachée à l'événement, appelé fonction de *callback*.

Les types de *widget* sont organisés en une structure hiérarchique, selon une idée similaire à celle des hiérarchies de classes dans les langages objets. Par exemple, un zone de texte (type *GtkTextView*) est aussi un conteneur (type *GtkContainer*), qui lui-même est un *GtkWidget*. Il est possible de changer le type d'un *widget* en un type parent dans la hiérarchie. GTK propose à cette fin des macro-définitions de la forme *<GTK_TYPE>* (*<widget>*). Par exemple, le code suivant permet de "transtyper" un *GtkTextView* en *GtkWidget*.

```
GtkTextView texte =...;  
GtkWidget wid=GTK_WIDGET(texte);
```

2. pour WInDows'GadgET.

Générateur d'interface Glade Le générateur d'interface *glade* est un outil de développement qui prend en charge la production d'une interface graphique décrite sous la forme d'un fichier XML. La figure 3 montre le canevas où ont été agencés les objets graphiques de l'application *follow*. Le fichier XML sera ensuite interprété dans un programme afin de créer les widgets par des appels aux fonctions Gtk correspondante.

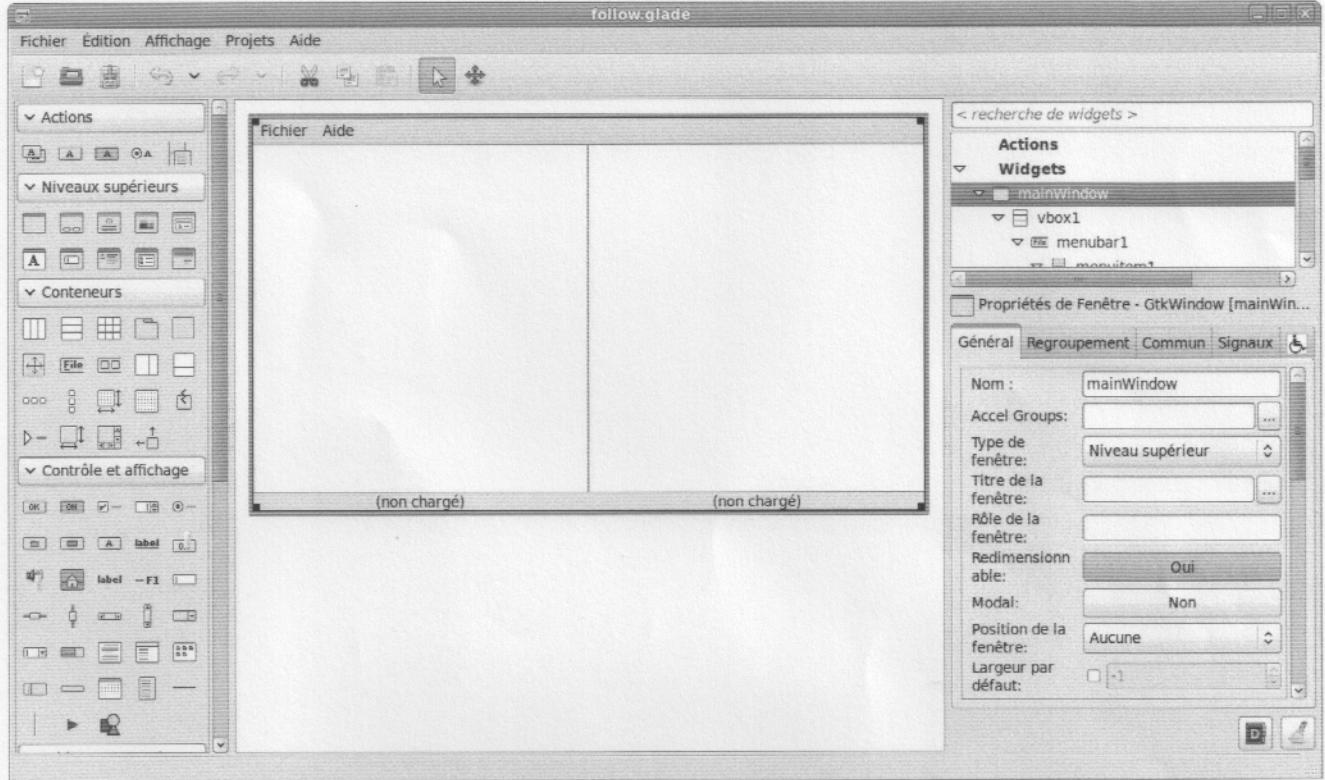


FIGURE 3 – Fenêtre principale de l'application *follow* vues dans le logiciel *glade*

Travail à effectuer

1. Mettre en place l'IHM à partir du générateur d'interface *Glade 3*
2. Tester l'affichage de l'IHM.

Séances 5 et 6

Travail à effectuer

1. Rassembler les différentes parties du code.
2. Intégrer les différentes parties du logiciel ; pour cela, écrire notamment le code des fonctions de *callback*, et mettre à jour les *widgets* d'affichage en fonction de l'alignement des textes.

Séance de Noël

1. Aligner "au mieux" le nouveau texte et le texte de référence au niveau de l'affichage.
Un défilement de l'un des 2 textes se traduit par le ré-alignement en conséquence de l'autre texte.

Le travail de Noël est facultatif, mais peut apporter un cadeau sous la forme de points supplémentaires sur votre note. Le père noël finit son taf annuel une semaine après la fin des vacances de noël ; il faudra donc lui soumettre ce travail avant cette date ...