MMFAI
Cours de programmation

TP numéro 7
semaine du 11 janvier 1999

Programmation en binome : de LCS/SES à Diff

Nous allons continuer notre experience de programmation à plusieurs en nous attaquant à un autre problème qui peut être aisément decomposé. Il s'agira ici de reécrire (une version simplifiée) de l'utilitaire Unix qui s'appelle `diff`.

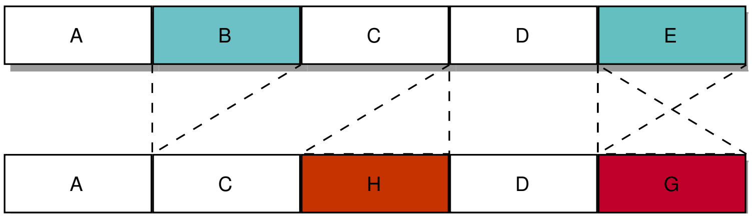
Le problème

Il s'agit de comparer deux fichiers A et B et d'afficher les différences : cette opération est bien pratique pour garder une trace des modifications sur des fichiers, pour transmettre seulement les modifications (—patches—) quand on change de version des sources, pour garder de façon compacte l'ensemble des différentes versions d'un document (voyez la page de manuel de `rcs` par exemple), etc.

Les différences peuvent être affichées de différentes façons, mais la plus utile est sous la forme d'un "edit script", i.e. une séquence de commandes d'éditions ayant la forme suivante

- nai, k Cela indique que à la ligne n du fichier A il faut rajouter les lignes i à k du ficher B qui sont données après la commande (on omet $,k$ si $k = i$)
- n, mci, k Cela indique qu'il faut changer les lignes n à m du fichier A par les lignes i à k du fichier B, et après la commande on donne les lignes du fichier A, suivies d'une ligne de 3 tirets, suivies des lignes du fichier B (on omet $,m$ si $n = m$ et $,k$ si $k = i$)
- n, mdi Cela indique qu'il faut effacer du fichier A les lignes n à m pour se retrouver alignés à la ligne i du fichier B (on omet $,m$ si $n = m$)

Une séquence d'instructions d'éditions pour transformer le fichier A dans le fichier B comme celles-ci a la très agréable propriété de pouvoir être facilement convertie en une sequence d'instruction pour convertir le fichier B dans le fichier A : il suffit de transformer nai, k en i, kdn et viceversa et d'inverser n, mci, k en i, kcn, m (en prénant soin aussi d'inverser les blocs avant et apres la ligne de 3 tirets).
Par exemple, sur les deux fichiers contenant les lignes montrees ici dans le dessin



`diff` rendra la séquence d'instructions d'édition suivante :

```
2d1
< B
3a3
> H
5c5
< E
---
> G
```

Utiliser LCS/SES

Le problème de produire un plus court "edit script" (SES) est isomorphe au problème de calculer une plus longue sous-séquence commune (LCS) : si vous calculez la plus longue sous-séquence commune en prénant soin de garder trace des indices dans les deux séquences à comparer, il est facile de produire le script.

Mais dans les cas d'un fichier, les "éléments" d'une séquence sont des lignes entiéres, et il serait peu pratique (et bien couteux) de calculer la LCS sur des vecteurs contenant les fichiers ligne par ligne : cela necessiterait d'utiliser une fonction de comparaison de ligne, dont le cout n'est pas $O(1)$.

On peut simplifier la tâche de la façon suivante :
- on construit à partir des deux vecteurs de lignes (obtenus en lisant chaque fichier) un dictionnaire associant une clef entiere différente à chaque ligne différente (et le même entier à toutes les lignes égales).
  Pour cela, on peut utiliser une fonction d'hachage.
- En même temps, on construit deux vecteurs d'entiers contenant les clés et non pas les lignes.
- On appelle SES sur les vecteurs des clefs.
- On transforme l'"edit script" sur les clefs rendu par SES en "edit script" sur les lignes.

L'interface

Nous assumons que la fonction SES rendra le script sous la forme d'un vecteur de commandes d'edition simplifiées, sans les lignes à rajouter ou changer, i.e. dans l'exemple d'en haut :

```
2d1
3a3
5c5
```

Et nous separons maintenant le programme en deux modules : un qui calcule la SES, et l'autre qui gêre la lecture des fichiers, la génération des vecteurs de clefs et l'impression du script d'edition complet.
Comme interface, on pourra utiliser par exmple un fichier `ses.h` comme le suivant :

```
/* une structure pour les commandes d'edition*/
typedef struct {
char action; /* a,d ou c */
                int rangea_1; /* n */
                int rangea_2; /* m */
                int rangeb_1; /* i */
                int rangeb_2; /* k */
} e_cmd;

/* la fonction SES */

int SES(int *vect1, int l1, int *vect2, int l2, e_cmd **script);
```

1.1 Tâches du programmeur A

Exercice 1.1 (Programmation de `diff`) *Écrivez une fonction* `int diff(char *fn1, char *fn2)` *qui utilise la fonction* `SES` *pour calculer la différence entre deux fichiers.*

Exercice 1.2 (Améliorations) *Modifier votre fonction pour permettre de ne pas tenir compte des différences produites par des séquences d'espaces de longuers différentes.*

1.2 Tâches du programmeur B

Exercice 1.3 (Programmation de SES) *Programmez la fonction SES (il peut être utile, pour la tester, de prévoir aussi l'impression de la sousequence commune). Quel est sa complexité, en espace et en temps ?*

Exercice 1.4 (Améliorations) *Cherchez "An O(ND) Difference Algorithm and its Variations", par Eugene Myers, qui se trouve dans Algorithmica Vol. 1 No. 2, 1986, pp. 251-266 (il est à la bibliotheque). Implementez la variante qui utilise espace linéaire pour calculer le résultat.*

1.3 Tâches communes à A et B

Exercice 1.5 *Compilez ensembe vos deux modules et . . . comparez avec* `diff`*.*

N'oubliez pas de vous déconnecter avant d'abandonner le clavier (voir notes du stage Unix).