TD 4: Comment maintenir plusieurs versions d'un ficher source?*

ED6 — Licence 3 — Université Paris Diderot

2012–2013

L'objectif de ces travaux dirigés est de manipuler les outils de développement qui permettent la coexistence et l'interaction entre différentes versions d'un même fichier source. Au terme de cette séance, vous devrez savoir :

- Calculer la différence textuelle entre deux versions d'un fichier source et l'utiliser pour passer d'une version à l'autre.
- Calculer les différences textuelles concurrentes entre deux versions d'un fichier de référence.
- Comprendre la notion de conflit entre modifications concurrentes.

Question 0.1. *Récupérez l'archive* `http://upsilon.cc/zack/teaching/1213/ed6/tp4-files.tar.gz`. *Dans un nouveau répertoire* `tp4` *de votre répertoire de travail Git, décompressez l'archive :* `tar xvfz tp4-files.tar.gz`. *Commitez tous les nouveau fichiers que vous venez d'ajouter (et rien d'autre dans le même commit !).*

1 «Arithmétique» entre fichiers textuels

Les outils `diff` et `patch` permettent de calculer et d'appliquer des modifications entre deux fichiers. Le rôle de cette section est d'apprendre à utiliser ces deux outils. `diff` calcule la différence ligne à ligne entre deux fichiers textuels en résolvant le problème de la *plus grande sous-séquence commune* et en exprimant la différence entre les fichiers à l'aide de trois opérations : insertion, suppression et changement.

Les travaux de recherche qui ont permis l'élaboration de cet outil sont expliqués dans le rapport (dont la lecture est recommandée) intitulé *An Algorithm for Differential File Comparison* [1], de J. W. Hunt et M. D. McIlroy. Voici un exemple tiré de l'introduction de ce document. Supposons qu'un fichier F_1 soit défini par la séquence de lignes suivantes :

a
b
c
d
e
f
g

et qu'une nouvelle version de ce fichier F_2 soit définie par :

w
a
b
x
y
z
e

Pour économie d'espace, nous allons plutôt représenter séquence de lignes comme ci-dessus en horizontal, p. ex . :

a b c d e f g

pour la première version et :

w a b x y z e

pour la deuxième.

Pour décrire les modifications pour passer de la première version à la version suivante du fichier, on peut utiliser la séquence $\Delta_{1\to 2}$ de changements suivants :

*Ce sujet de TD est inspiré d'un sujet proposé par Yann Régis-Gianas

Insérer avant la ligne 0	:	w
Remplacer les lignes 3 à 4 dont les contenus sont	:	c d
Par	:	x y z
Supprimer les lignes 6 à 7 dont les contenus sont	:	f g

On peut aussi décrire le passage inverse $\Delta_{2\to1}$ ainsi :

Supprimer la ligne 1 dont le contenu est	:	w
Changer la ligne 4 à la ligne 6 dont les contenus sont	:	x y z
Par	:	c d
Insérer après la ligne 7, les contenus	:	f g

Ces descriptions contiennent un niveau d'information important : on ne dit pas seulement qu'on supprime la ligne 6 mais on indique aussi quel est le contenu que l'on supprime. Grâce à cela, il est très simple de calculer $\Delta_{1\to2}$ à partir de $\Delta_{2\to1}$, et réciproquement. En termes de représentation concrète, l'outil `diff` utilise un format plus succinct que celui des tableaux précédents pour décrire les changements. Par exemple, $\Delta_{1\to2}$ et $\Delta_{2\to1}$ s'expriment par les deux séquences de caractères suivantes :

```
0 a 1,1
> w
3,4 c 4,6
< c
< d
---
> x
> y
> z
6,7 d 7
< f
< g
```

$\Delta_{1\to2}$

```
1,1 d 0
< w
4,6 c 3,4
< x
< y
< z
---
> c
> d
7 a 6,7
> f
> g
```

$\Delta_{2\to1}$

Question 1.1. *Pour compléter votre connaissance de cet outil, lisez la page de manuel de* ***`diff`****.*

Question 1.2. *Calculez la différence entre le fichier* `foo-v1.c` *et le fichier* `foo-v2.c` *et enregistrez-la dans un fichier* `foo.diff`*. Trouvez une option de* ***`diff`*** *affichant la liste des changements sous une forme plus lisible par un humain que le format standard.*

Question 1.3. *Téléchargez deux images sur internet. Que répond* ***`diff`*** *lorsque l'on essaie de calculer leur différence ?*

Question 1.4. *Lisez la page de manuel de l'outil* ***`patch`****. À l'aide du fichier* `foo-v1.c` *et* `foo.diff`*, retrouver le fichier* `foo-v2.c` *et vérifiez (avec* ***`diff`****) que le fichier que vous avez obtenu est identique à la version originelle de* ***`foo-v2.c`****.*

Question 1.5. *Effectuez la procédure inverse (i.e. obtenez* ***`foo-v1.c`*** *à partir de* ***`foo-v2.c`*** *et* ***`foo.diff`****) d'abord en réutilisant* ***`diff`*** *puis à l'aide de l'option* `-R` *de* ***`patch`****.*

Question 1.6. *Écrire un script* Shell `check-expected.sh` *qui attend deux noms de fichiers en argument. Le premier argument correspond à la production d'un programme sur la sortie standard tandis que le second argument correspond à la production attendue pour ce programme. Le script doit réussir si les deux fichiers sont identiques et échouer dans le cas contraire. La différence éventuelle entre les deux fichiers doit être sauvegardée dans un fichier dont le nom est formé à l'aide de la commande* `‘basename $1‘.diff`*.*

Question 1.7. *À l'aide d'une option de* ***`diff`****, sauvegardez la différence de contenu entre le répertoire* `src-v1` *et le répertoire* `src-v2` *dans un fichier* `src.diff`*. Utilisez* ***`patch`*** *pour produire un répertoire* `src-v2-bis` *à partir de* `src-v1` *et* `src.diff`*. Que devez vous faire pour inclure le contenu de* ***`foo.h`*** *dans la sortie de* ***`diff`*** *?*

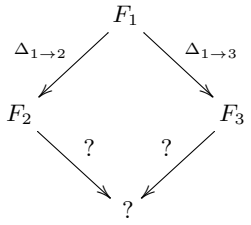
Question 1.8. *(MiniProjet — Bonus) Écrire deux scripts* Shell*,* `save.sh` *et* `undo.sh` *qui attendent un nom de fichier* `filename` *en argument.*

Le rôle de `save.sh` *est de sauvegarder la différence entre* `filename` *et un fichier* `.filename.last-backup` *dans un fichier* `.filename.diff-N` *avec* `N`*, l'entier naturel le plus petit tel qu'il n'existe pas déjà de fichier de ce nom. Si le fichier* `.filename.last-backup` *n'existe pas alors aucune différence n'est enregistrée et le contenu du fichier* `filename` *est copié dans le fichier* `.filename.last-backup`*.*

Le rôle de `undo.sh` *est de restaurer l'état du fichier* `filename` *à l'aide du contenu de* `.filename.last-backup`*. Si ce dernier n'existe pas, un message d'erreur est produit. Une fois la copie effectuée, le fichier* `.filename.last-backup` *est modifié à l'aide de la différence sauvegardée dans le fichier* `.filename.diff-N` *dont l'entier naturel* `N` *est le plus grand parmi tous les noms de fichier de cette forme. Ce fichier* `.filename.diff-N` *est alors détruit. Si aucun fichier de cette forme n'existe, alors le fichier* `.filename.last-backup` *est détruit.*

2 Diplomatie entre fichiers sources

Il existe parfois des **versions concurrentes** d'un fichier source, c'est-à-dire des versions formées à partir de modifications, éventuellement **conflictuelles**, d'une version d'origine. Cette situation intervient naturellement lors d'un travail en équipe : lorsque deux programmeurs travaillent dans la même unité de compilation sur deux blocs de code différents (ce qui devrait être sans risque) ou sur le même bloc (ce qui doit entraîner un conflit si les modifications ne sont pas identiques). La figure suivante schématise cette situation :



Pour pouvoir faire évoluer sans risque la version d'origine du fichier source F_1 dans une version F_{23} réconciliant les modifications faites par $\Delta_{1\to2}$ et $\Delta_{1\to3}$, il faut calculer une différence $\Delta_{1\to2,3}$ qui englobent les modifications non conflictuelles de $\Delta_{1\to2}$ et $\Delta_{1\to3}$ et effectuent un choix entre les changements de $\Delta_{1\to2}$ et $\Delta_{1\to3}$ lorsqu'il y a un conflit. L'outil `diff3` d'aider à calculer cette **fusion de changements** en calculant les différences concurrentes non conflictuelles automatiquement et en exhibant les parties conflictuelles de façon à ce que le programmeur puisse résoudre les conflits manuellement.

Les détails algorithmiques et techniques de l'outil `diff3` sont assez subtils. Pour une présentation formelle, on consultera l'article *A Formal Investigation of Diff3* [2], écrit par Sanjeev Khanna, Keshav Kunal et Benjamin C. Pierce. L'exemple suivant, adapté de cet article, explique un cas d'utilisation de `diff3`.

Soient trois fichiers dont les contenus sont :

F_2	1	4	5	2	3	6
F_1	1	2	3	4	5	6
F_3	1	2	4	5	3	6

La première étape du travail de `diff3` est de calculer les sous-séquences où F_1 coïncide avec F_2 et F_1 coïncide avec F_3. L'intersection de ces sous-séquences fournit les parties du fichier qui n'ont pas été modifiées. Ici, il s'agit des lignes $1, 2 et 6$. Le fichier F_{23} devra donc être de la forme suivante (le point d'interrogation représente une ou plusieurs lignes) :

F_{23}	1	?	2	?	6

Entre la ligne 1 et la ligne 2, seule la modification $\Delta_{1\to2}$ insère les lignes 4 et 5. La modification $\Delta_{1\to3}$ ne s'intéresse pas à cette partie du fichier. La modification de $\Delta_{1\to2}$ est donc considérée comme non-conflictuelle.

Entre les lignes 2 et 6, $\Delta_{1\to2}$ et $\Delta_{1\to3}$ font deux choses totalement différentes. $\Delta_{1\to2}$ change les lignes 3, 4 et 5 en la ligne 3 tandis que $\Delta_{1\to3}$ change les lignes 3, 4 et 5 en les lignes 4, 5 et 3. Il s'agit d'une zone conflictuelle qui doit être signalée par `diff3`.

La production finale de `diff3` (commande : `diff3 -m f2 f1 f3`) est alors :

```
1
4
5
2
<<<<<<< f2
3
||||||| f1
3
4
5
=======
4
5
3
>>>>>>> f3
6
```

La section entre `<<<<<<<` et `>>>>>>>` correspond au conflit. `diff3` indique la nécessité d'un choix.

Dans cet exercice, vous jouez le rôle d'un modérateur de développement : deux développeurs vous ont envoyé deux *patchs* (`joe.patch` et `linus.patch`) qui correspondent à deux modifications du fichier `foo.c` et vous devez produire deux nouveaux patchs pour synchroniser leur travail en officialisant les modifications intégrées dans le développement principal.

Question 2.1. *Calculez la fusion des modifications proposée par* `diff3`.

Question 2.2. *Exhibez le conflit et résolvez-le.*

Question 2.3. *Calculez les patchs de résolution de conflit à envoyer aux deux développeurs.*

Références

[1] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Technical Report CSTR 41, Bell Laboratories, Murray Hill, NJ, 1976.

[2] Sanjeev Khanna, Keshav Kunal, and Benjamin C. Pierce. A formal investigation of diff3. In Arvind and Prasad, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, December 2007.