Introduction à la gestion de configurations avec CVS

Christian Rinderknecht

4 février 2015

1 Introduction

CVS (*Concurrent Versions System*) est un système de gestion de versions de fichiers textuels. Il archive de façon compacte les versions successives des fichiers placés sous son contrôle, ainsi que l'historique détaillé des modifications.

CVS est utile pour des programmeurs, même solitaires, mais aussi pour gérer la création d'un livre ou d'un site *web* par exemple. Il s'impose lorsque plusieurs personnes travaillent indépendamment les unes des autres.

CVS évite le problème de la multiplicité des copies de travail, il automatise la manipulation des arborescences de fichiers en permettant l'accès aux différentes versions d'un fichier, il permet de comparer deux versions entre elles et de consulter les commentaires associés.

CVS conserve ses données dans une base centralisée, concrètement un répertoire avec des fichiers d'administration. Chaque usager y importe ses fichiers initiaux, s'il y en a, puis en extrait une copie sur laquelle il travaille, et finalement l'archive à nouveau dans la base. Cette méthode permet le travail en équipe et CVS fournit un mécanisme de fusion pour gérer les conflits potentiels entre deux modifications en parallèle d'un même fichier, sinon l'usager doit lui-même réaliser la fusion avant d'archiver sa nouvelle version à jour.

CVS peut fonctionner en mode client-serveur, permettant un travail coopératif distribué, c'est-à-dire qu'il n'est pas nécessaire de partager le même espace disque, ou le même lieu.

Comme le dit la documentation de CVS :
— *CVS is not a build system,*
— *CVS is not a substitute for management,*
— *CVS is not a substitute for developer communication,*
— *CVS does not have change control (like bug-tracking),*
— *CVS is not an automated testing program,*
— *CVS does not have a builtin process model.*
Le but de ce TP est de vous familiariser avec un usage simple de CVS sur un petit exemple : la gestion de quelques pages web.

2 Création et peuplement initial de la base

À la racine de votre compte, créez le répertoire qui contiendra la base CVS :

```
~$ mkdir cvsroot
```

Lors de travail en équipe, il faut prendre soin que tous les membres aient les droits de lecture et d'écriture dans ce répertoire (c'est-à-dire dans la base). On peut pour cela créer un groupe Unix commun par exemple.

Ensuite définissez la variable d'environnement qui donne le mode d'accès et le chemin absolu de cette base :

```
~$ export CVSROOT=:local:$HOME/cvsroot
```

(Ceci vaut pour les usagers du Bourne Shell et dérivés. Le mieux est de placer cette définition dans un fichier qui est exécuté à chaque fois qu'un shell est lancé, par exemple dans `.bashrc`). Le mode d'accès local signifie que les fichiers de la base sont directement accessibles au travers du système de fichiers.

Maintenant vous pouvez créer votre base CVS :

```
~$ cvs init
```

La sous-commande init de cvs a créé un répertoire `$HOME/cvsroot/CVSROOT` qui contient tous les fichiers d'administration de la base, et qui contiendra les archives à venir.

On parle de sous-commande plutôt que de commande pour distinguer, dans le premier cas, une commande de CVS et, dans le dernier cas, une commande Unix (elle peuvent en effet avoir le même nom).

Si vous souhaitez que CVS vous propose Emacs comme éditeur de texte, alors définissez la variable d'environnement :

```
~$ export CVSEDITOR=/usr/bin/emacs
```

Téléchargez l'archive `gi108.tgz` qui se trouve dans l'arborescence à partir de
`http://aldebaran.devinci.fr/~rinderkn/Teaching/index.html`
et ouvrez-là dans `~/tmp` :

```
~$ mkdir -p tmp
~$ cd tmp
~/tmp$ tar xzf gi108.tgz
```

Il s'agit de la correction de deux travaux pratiques que vous allez reprendre à votre compte le temps d'apprendre l'usage de base de CVS.

Placez-vous dans le répertoire où se trouvent les répertoires et fichiers à archiver :

```
~/tmp$ cd gi108
```

Pour importer dans la base CVS les premiers fichiers, on utilise la sous-commande import. La syntaxe simplifiée est

`cvs import -m` <*message*> <*module*> <*branche*> <*version*>

Le message (une chaîne de caractères) explique le but de la sous-commande ; un module est un chemin dénotant le répertoire où les fichiers importés sont insérés dans la base CVS ; une branche est une direction de développement et une version caractérise tous les fichiers importés.

Pour l'instant, seul le choix du module est vraiment important. Il ne faut pas prendre le même chemin où se trouvent les sources à importer (voilà le pourquoi de votre présence dans `~/tmp`), ni le répertoire racine (`$HOME`). Lorsque l'on extrayera le module hors de la base, cela se fera sous le chemin de répertoire donné lors de l'importation. Choisissez `WWW/TD` et vérifiez donc que vous n'avez aucun répertoire déjà existant avec ce nom-là. Ensuite :

```
~/tmp/gi108$ cvs import -m "Originaux" WWW/TD Init V0
N WWW/TD/index.html
cvs import: Importing /home/der_gi/rinderkn/cvsroot/WWW/TD/TD2
N WWW/TD/TD2/datef.sh
N WWW/TD/TD2/emacs.txt
N WWW/TD/TD2/lecture
N WWW/TD/TD2/td2.html

No conflicts created by this import
```

L'annotation `N` indique que le fichier correspondant est nouveau (New). Si l'option `-m` (signifiant « message ») n'avait pas été donnée, CVS aurait ouvert une fenêtre avec Emacs.

D'éventuels liens symboliques auraient été ignorés (annotation `I`) car ils n'ont pas de contenu en eux-même, or CVS se base sur le contenu des fichiers pour travailler.

3 Utilisation courante

3.1 Création d'une copie de travail

La première étape après l'importation initiale est d'extraire une version de travail à partir de laquelle les développements seront faits. Pour cela, créez le répertoire qui vous convient, allez-y (ici nous allons à la racine) :

```
~/tmp$ cd ~
```

et faites :

```
~$ cvs checkout WWW
cvs checkout: Updating WWW
cvs checkout: Updating WWW/TD
U WWW/TD/index.html
cvs checkout: Updating WWW/TD/TD2
U WWW/TD/TD2/datef.sh
U WWW/TD/TD2/emacs.txt
U WWW/TD/TD2/lecture
U WWW/TD/TD2/td2.html
```

L'annotation `U` indique que le fichier correspondant est mis à jour (Updated) — ici il est en fait créé car il n'y avait pas de version précédente. Cette sous-commande checkout a créé une arborescence `WWW` enracinée en `~` (lieu où vous appelez la sous-commande). Si vous faites maintenant :

```
~$ find ~/WWW -type d
/home/der_gi/rinderkn/WWW
/home/der_gi/rinderkn/WWW/CVS
/home/der_gi/rinderkn/WWW/TD
/home/der_gi/rinderkn/WWW/TD/CVS
/home/der_gi/rinderkn/WWW/TD/TD2
/home/der_gi/rinderkn/WWW/TD/TD2/CVS
```

Vous voyez alors qu'en plus de vos fichiers, il y a à chaque niveau de l'arborescence un répertoire nommé CVS. Ces répertoires ne doivent en aucun cas être modifiés car ils servent à l'administration de votre archive et ne sont modifiés que par CVS. C'est pour cette raison aussi qu'il ne fallait pas importer le module initial avec le même chemin que le répertoire où vous étiez.

Règle d'or : il ne faut faire un checkout que dans un espace neuf ou dans une copie de travail.

Si une sous-commande checkout échoue, en général vous devez vous assurer que :
- vous avez le droit d'écrire dans le répertoire courant,
- la variable d'environnement CVSROOT indique la bonne base CVS et la bonne méthode d'accès (local, client-serveur etc.),
- vous avez les droits d'accès aux fichiers de la base.

3.2 Archivage

Éditez `~/WWW/TD/TD2/td2.html` :

```
~$ cd WWW
~/WWW$ emacs TD/TD2/td2.html &
```

et ajoutez quelques lignes à la fin. Pour archiver votre modification, tapez (en vérifiant d'abord que vous êtes dans `~/WWW`) :

```
~/WWW$ cvs commit -m "Ajout de quelques lignes à la fin"
cvs commit: Examining .
cvs commit: Examining TD
cvs commit: Examining TD/TD2
Checking in TD/TD2/td2.html;
/home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v  <-- td2.html
new revision: 1.2; previous revision: 1.1
done
```

CVS a déterminé que `~/WWW/TD/TD2/td2.html` avait été modifié et devait être archivé. Le fichier `td2.html,v` qui apparaît à la cinquième ligne est le fichier d'administration associé à `td2.html`. Nous apprenons au passage que la version précédente était `1.1` et que la nouvelle version est `1.2`. En effet, la version initiale est toujours numérotée `1.1`.

Il aurait été possible de préciser quel fichier vous vouliez archiver :

```
~/WWW$ cvs commit -m "Ajout de quelques lignes" TD/TD2/td2.html
```

En effet, s'il n'y a pas de nom de fichier, alors CVS opère récursivement à partir du répertoire courant, comme vous l'avez constaté. Toutes les sous-commandes opérant sur des fichiers, si elles n'ont pas de nom de fichier en argument, opèrent ainsi, de même que si vous passez un nom de répertoire.

Règles d'or

- Si vous développez un programme, n'archivez jamais des fichiers qui ne compilent pas. Dans le cas de pages web, vérifiez que vous pouver visualisez sans erreur les pages concernées avant de les archiver.
- Archivez toujours un fichier avant une modification importante, de façon à pouvoir éventuellement revenir sur vos pas.

Note CVS n'archive pas le texte complet de votre fichier, mais les différences avec la version précédente.

3.3 Conflit

Imaginez maintenant que vous soyez plusieurs personnes à travailler sur les fichiers archivés. Il se peut que vous et un collègue exportiez une copie de travail d'un *même* fichier. Chacun modifie alors sa copie, votre collègue archive sa modification d'abord et puis vous tentez d'en faire de même. Il se peut alors qu'il y ait un conflit entre vos deux versions si vous avez modifié les mêmes parties du fichier en question. Vous allez simuler cette situation.

Placez-vous dans un répertoire quelconque (ici `~/tmp`) et extrayez une autre copie de travail (correspondant au collègue) :

```
~/WWW/TD/TD2$ cd ~/tmp
~/tmp$ cvs ckeckout WWW
cvs checkout: Updating WWW
cvs checkout: Updating WWW/TD
U WWW/TD/index.html
cvs checkout: Updating WWW/TD/TD2
U WWW/TD/TD2/datef.sh
U WWW/TD/TD2/emacs.txt
U WWW/TD/TD2/lecture
U WWW/TD/TD2/td2.html
```

Éditez `~/tmp/WWW/TD/TD2/td2.html` et profitez de l'occasion pour vérifier la présence des lignes que vous avez précédemment ajoutées à la fin. Ensuite, modifiez `~/tmp/WWW/TD/TD2/td2.html` en supprimant un bloc de texte dont vous repérez bien la position. Par exemple, dans la première partie intitulée *Partie Emacs*, supprimez les items 2 et 3 et sauvegardez le fichier. Puis archivez votre modification :

```
~/tmp$ cd WWW/TD/TD2
~/tmp/WWW/TD/TD2$ cvs commit -m "Suppression des items 2 & 3" td2.html
Checking in td2.html;
/home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v  <-- td2.html
new revision: 1.3; previous revision: 1.2
done
```

Revenez dans votre répertoire de travail initial et supprimez un bloc de texte dans `~/WWW/TD/TD2/td2.html` qui recouvre en partie celui supprimé par votre pseudo-collègue. Par exemple, supprimez les items 1 et 2 puis sauvegardez et tentez d'archiver :

```
~/tmp/WWW/TD/TD2$ cd ~/WWW/TD/TD2
~/WWW/TD/TD2$ cvs commit -m "Suppression des items 1 & 2" td2.html
cvs commit: Up-to-date check failed for ‘td2.html’
cvs [commit aborted]: correct above errors first!
```

CVS vous oblige donc à mettre à jour vous-même le fichier `td2.html` avant de l'archiver. Pour cela il faut lui dire notre intention :

```
~/WWW/TD/TD2$ cvs update td2.html
RCS file: /home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into td2.html
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in td2.html
C td2.html
```

Vous lisez clairement qu'il y a un conflit entre les versions 1.2 (la vôtre avant modification) et 1.3 (la version archivée de votre pseudo-collègue). L'annotation `C` signifie Conflict.

CVS a maintenant inséré dans le fichier conflictuel des marques entre les portions du texte en conflit (rafraîchissez votre *buffer* emacs avec M-x revert-buffer suivi de yes) :

```
<<<<<<< td2.html
3. lorsque l'on ouvre dans un buffer un fichier de type html, on passe
   en <b>mode HTML</b>" ("<tt>(HTML)</tt>" apparait dans la "modeline")
   alors noter la présence de menus spécifiques à  ce mode, avec des
   items comme par exemple "<tt>Href Anchor</tt> (C-c C-c h)</tt>"
   d'insérer la balise <tt>"&lt;a href="</tt> dans son buffer.
</p>

<p>
=======
1. Le fichier <a href="emacs.txt">emacs.txt</a>
</p>

<p>
>>>>>>> 1.3
```

Votre version de la zone en conflit est toujours présentée en premier entre les marques `<<<<<<<` et `=======`. La version de la base vient en second entre les marques `=======` et `>>>>>>>`, suivies de la version dans la base (`1.3`). En effet, dans la base, les items 2 et 3 sont absents, et nous voulons supprimer 1 et 2 : le conflit porte donc sur 1 et 3 (tout le monde semble d'accord pour supprimer 2).

CVS empêche tout archivage avant que le conflit n'aie été résolu :

```
~/WWW/TD/TD2$ cvs commit -m "Suppression des items 1 & 2" td2.html
cvs commit: file ‘td2.html’ had a conflict and has not been modified
cvs [commit aborted]: correct above errors first!
```

Décidez de supprimer 1 et 3 et recommencez

```
~/WWW/TD/TD2$ cvs commit -m "Suppression des items 1 & 2" td2.html
Checking in td2.html;
/home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v  <--  td2.html
new revision: 1.4; previous revision: 1.3
done
```

Vous devez donc vous rappeler qu'*il faut toujours faire un update avant un commit* (rigoureusement, ce n'est réellement nécessaire que si l'on travaille à plusieurs, mais c'est une excellente habitude à prendre dès maintenant).

Si vous ne voulez pas vous retrouver tout de suite après le update avec des conflits à résoudre dans vos fichiers, pensez à utiliser l'option `-n` de la sous-commande update, en saisissant

```
~/WWW/TD/TD2$ cvs -n update td2.html
```

Cette option inhibe en effet toute écriture sur le disque (elle est comparable en ce sens à la même option de l'utilitaire make). Elle est très souvent utilisée car elle permet aussi d'obtenir l'état des fichiers à mettre à jour dans un répertoire de travail (`cvs -n update`) plus compendieusement qu'avec la sous-commande status. C'est donc une bonne habitude à prendre.

Il n'y a pas de règle générale (en tout cas, pas imposée par CVS) pour savoir quand faire une mise à jour de vos fichier : si vous le faites trop souvent, vous risquez de récupérer trop d'erreurs d'autrui, et si vous ne le faites pas assez souvent, vous risquez de trop diverger et donc d'avoir beaucoup de conflits complexes à régler.

3.4 Historique

Il est possible de consulter l'historique des versions d'un fichier à l'aide de la sous-commande log :

```
~/WWW/TD/TD2$ cvs log td2.html

RCS file: /home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v
Working file: td2.html
head: 1.4
branch:
locks: strict
access list:
symbolic names:
        V0: 1.1.1.1
        Init: 1.1.1
keyword substitution: kv
total revisions: 5;     selected revisions: 5
description:
----------------------------
revision 1.4
date: 2003/05/15 10:20:02;  author: rinderkn;  state: Exp;  lines: +0 -2
Suppression des items 1 & 2
----------------------------
revision 1.3
```

```
date: 2003/05/15 09:13:47;  author: rinderkn;  state: Exp;  lines: +0 -25
Suppression des items 2 & 3
----------------------------
revision 1.2
date: 2003/05/15 09:12:43;  author: rinderkn;  state: Exp;  lines: +4 -0
Ajout de quelques lignes à la fin
----------------------------
revision 1.1
date: 2003/05/15 08:57:43;  author: rinderkn;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2003/05/15 08:57:43;  author: rinderkn;  state: Exp;  lines: +0 -0
Originaux
=============================================================================
```

Vous pouvez ainsi lire toutes les informations concernant les auteurs des changements (ici il n'y a qu'un auteur), les dates de création, les numéros utilisés, la description des changements etc.

À tout moment vous pouvez aussi consulter l'état de votre copie de travail pour savoir si vous l'avez modifié, si elle est à jour par rapport à la base ou si une nouvelle version est disponible dans la base :

```
~/WWW/TD/TD2$ cvs status td2.html
===================================================================
File: td2.html          Status: Up-to-date

   Working revision:    1.4     Thu May 15 10:17:23 2003
   Repository revision: 1.4     [...]/cvsroot/WWW/TD/TD2/td2.html,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

Votre version de travail est donc la même que celle dans la base (cf. Up-to-date).

Si vous modifiez `~/WWW/TD/TD2/td2.html` (en ajoutant quelques espaces avant la balise <*html*> par exemple), et demandez à nouveau l'état du fichier par rapport à la base :

```
~/WWW/TD/TD2$ cvs status td2.html
===================================================================
File: td2.html          Status: Locally Modified

   Working revision:    1.4     Thu May 15 10:17:23 2003
   Repository revision: 1.4     [...]/cvsroot/WWW/TD/TD2/td2.html,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

Vous lisez alors que son état est `Locally Modified` et non plus `Up-to-date`, comme précédemment. Si vous faites :

```
~/WWW/TD/TD2$ cvs update
cvs update: Updating .
M td2.html
```

vous lisez que CVS a mis à jour sans problème votre copie locale. L'annotation M signifie Modified. Vous pouvez vérifier la présence des espaces en première ligne de votre fichier. Votre modification ne sera archivée que si vous faites commit.

3.5 Suppression

Vous souhaitez maintenant supprimer le fichier `~/WWW/TD/TD2/emacs.txt` qui est désormais inutile. Pour cela, la procédure est la suivante.

— Supprimez la version de travail ;
— utilisez la sous-commande remove pour signifier votre intention à CVS ;
— utilisez la sous-commande commit pour effectivement éliminer le fichier de la base.

Si vous oubliez de supprimer le fichier localement, alors :

```
~/WWW/TD/TD2$ cvs remove emacs.txt
cvs remove: file ‘emacs.txt’ still in working directory
cvs remove: 1 file exists; remove it first
```

Vous faites donc :

```
~/WWW/TD/TD2$ rm -f emacs.txt
~/WWW/TD/TD2$ cvs remove emacs.txt
cvs remove: scheduling ‘emacs.txt’ for removal
cvs remove: use ’cvs commit’ to remove this file permanently
~/WWW/TD/TD2$ cvs commit -m "Suppression de emacs.txt" emacs.txt
Removing emacs.txt;
/home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/emacs.txt,v  <--  emacs.txt
new revision: delete; previous revision: 1.1.1.1
done
```

Attention *Toutes les mises à jour de ce fichier par d'autres collègues verront sa disparition de leur copie de travail.* Les versions antérieures sont néanmoins toujours disponibles, et vous pouvez éventuellement les extraire à nouveau (à l'aide du numéro de version ou de la date). Vous pouvez aussi ajouter un nouveau fichier qui porte le même nom qu'un fichier supprimé.

Note il n'est pas possible de supprimer directement des répertoires. Pour y parvenir indirectement, vous devez supprimer tous les fichiers qu'il contient, puis lors des mises à jour (update) ou des extractions (checkout), utilisez l'option `-P` qui inhibe la création des répertoires vides. Ce peut être une bonne habitude à prendre que d'utiliser systématiquement cette option.

3.6 Ajout

Copiez maintenant le fichier `copyright` se trouvant dans le répertoire `gi108` vers `~/WWW/TD`.

Vous devez ensuite dire à CVS votre intention de l'ajouter dans la base à l'aide de la sous-commande add :

```
~/WWW/TD/TD2$ cd ~/WWW/TD
~/WWW/TD$ cvs add -m "Copyright des pages" copyright
cvs add: scheduling file ‘copyright’ for addition
cvs add: use ’cvs commit’ to add this file permanently
```

Puis utilisez la sous-commande commit pour l'archiver :

```
~/WWW/TD$ cvs commit -m "Première version" copyright
RCS file: /home/der_gi/rinderkn/cvsroot/WWW/TD/copyright,v
done
Checking in copyright;
/home/der_gi/rinderkn/cvsroot/WWW/TD/copyright,v  <--  copyright
initial revision: 1.1
done
```

Note : *la sous-commande add n'est pas récursive.*
Placez le fichier `td3.tar` se trouvant dans le répertoire `gi108` *hors d'une copie de travail*, par exemple `~/tmp` :

```
~/WWW/TD$ cd ~/tmp
~/tmp$ tar xf td3.tar
```

Cette archive tar contient une arborescence que nous voulons archiver dans la base CVS — précédemment nous avons ajouté un seul fichier, c'est-à-dire copyright.

```
~/tmp$ cd TD3
~/tmp/TD3$ cvs import -m "Première version du TD3" WWW/TD/TD3 Start V0
N WWW/TD/TD3/td3.html
N WWW/TD/TD3/simple_man_ar.html
N WWW/TD/TD3/simple_man_find.html
N WWW/TD/TD3/simple_man_grep.html
N WWW/TD/TD3/simple_man_tar.html
N WWW/TD/TD3/aa.rtf

No conflicts created by this import
```

Puis revenez à votre copie de travail :

```
~/WWW/TD$ cvs update -d
cvs update: Updating .
cvs update: Updating TD2
M TD2/td2.html
cvs update: Updating TD3
U TD3/aa.rtf
U TD3/simple_man_ar.html
U TD3/simple_man_find.html
U TD3/simple_man_grep.html
U TD3/simple_man_tar.html
U TD3/td3.html
```

Par défaut, la sous-commande update ne créé pas les répertoires éventuellement archivés depuis l'extraction de la copie de travail ou la dernière mise à jour avec l'option `-d`. Cette option (*directories*) demande la création éventuelle de ces répertoires. Vous pouvez d'autre part remarquer que `TD2/td2.html` a été modifié (annotation M) : ce sont les espaces supplémentaires que vous aviez ajoutés à la section 3.4.

Vous auriez pu procéder autrement. Pour vous en rendre compte par vous-même, rendez-vous dans `~/tmp` où votre pseudo-collègue avait extrait une copie de travail et saisissez

```
~/tmp$ cvs checkout WWW/TD/TD3
cvs checkout: Updating WWW/TD/TD3
U WWW/TD/TD3/aa.rtf
U WWW/TD/TD3/simple_man_ar.html
U WWW/TD/TD3/simple_man_find.html
U WWW/TD/TD3/simple_man_grep.html
U WWW/TD/TD3/simple_man_tar.html
U WWW/TD/TD3/td3.html
```

Concernant le choix des fichiers à ajouter, il convient de n'archiver que les fichiers textuels qui ne peuvent être produits dynamiquement (par un outil). En effet, CVS utilise l'utilitaire diff pour l'archivage dans le but de réduire ainsi considérablement la taille de la base. Les binaires ne sont pas en général des fichiers que l'on ajoute sans bonne raison. Les liens symboliques ne peuvent être ajoutés (un contournement consiste à archiver un script qui les créé).

3.7 Différence entre deux versions archivées

Il est fréquent que l'on ne se souvienne plus des différences entre la version de travail d'un fichier et une de ses versions archivées (en particulier si on travaille en groupe et que l'on doive comprendre le code d'un autre). Supposons que nous voulions ici voir les différences entre les versions 1.2 et 1.3 du fichier `/WWW/TD/TD2/td2.html`. Nous savons par la sous-commande log (cf. section 3.4) que le message indique la suppression des items 2 et 3. Pour vérifier que le commentaire est pertinent on utilise la sous-commande diff avec deux options `-r` (Release) indiquant les versions à comparer à l'aide de la commande Unix `diff` :

```
~/WWW$ cvs diff -r 1.2 -r 1.3 TD/TD2/td2.html
Index: TD/TD2/td2.html
===================================================================
RCS file: /home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v
retrieving revision 1.2
retrieving revision 1.3
diff -r1.2 -r1.3
16,40d15
< 2. on a remplacé le mot 'buffer' par '"buffer" en anglais' en
<
< <ul>
<   <li>
<     se positionnant au debut du buffer avec la sous-commande nommée
<     '<tt><b>beginning-of-buffer</b></tt>', "bindée" (associée [...]
<     de touches) à  <tt><b>M-&lt;</b></tt> (Meta-inferieur)
<   </li>
<
<   <li>
<      en utilisant la sous-commande '<tt><b>Query replace [...]
<   </li>
< </ul>
< </p>
<
< <p>
< 3. lorsque l'on ouvre dans un buffer un fichier de type html, on passe
```

```
<    en <b>mode HTML</b>" ("<tt>(HTML)</tt>" apparait dans [...]
<    alors noter la présence de menus spécifiques à  ce mode, avec des
<    items comme par exemple "<tt>Href Anchor</tt> (C-c C-c h) [...]
< </p>
<
< <p>
```

3.8 Extraction d'une ancienne version

Un des grands intérêts de CVS est qu'il est possible d'extraire une version antérieure, même si l'unique répertoire de travail a été effacé. Cela se fait bien sûr par le biais de la sous-commande `checkout`, mais avec l'option `-r` suivie du numéro de la version désirée. Supposons dans notre cas que l'on veuille extraire la version initialement importée dans la base et la mettre dans un sous-répertoire `Depart`. On saisit alors :

```
~/WWW$ cvs checkout -d Depart -N -r 1.1 WWW
cvs checkout: Updating Depart/WWW
cvs checkout: Updating Depart/WWW/TD
U Depart/WWW/TD/index.html
cvs checkout: Updating Depart/WWW/TD/TD2
U Depart/WWW/TD/TD2/datef.sh
U Depart/WWW/TD/TD2/emacs.txt
U Depart/WWW/TD/TD2/lecture
U Depart/WWW/TD/TD2/td2.html
cvs checkout: Updating Depart/WWW/TD/TD3
U Depart/WWW/TD/TD3/aa.rtf
U Depart/WWW/TD/TD3/simple_man_ar.html
U Depart/WWW/TD/TD3/simple_man_find.html
U Depart/WWW/TD/TD3/simple_man_grep.html
U Depart/WWW/TD/TD3/simple_man_tar.html
U Depart/WWW/TD/TD3/td3.html
~/WWW$ ls Depart/WWW/TD/TD2
CVS  datef.sh  emacs.txt  lecture  td2.html
```

L'option supplémentaire `-N` fait que le module (ici `WWW`) est extrait dans le répertoire spécifié par `-d` (ici `Depart`) au lieu de remplacer le nom du module par l'argument de `-d`. Autrement dit :

```
~/WWW$ rm -fr Depart
~/WWW$ cvs checkout -d Depart -r 1.1 WWW
cvs checkout: Updating Depart
cvs checkout: Updating Depart/TD
U Depart/TD/index.html
cvs checkout: Updating Depart/TD/TD2
U Depart/TD/TD2/datef.sh
U Depart/TD/TD2/emacs.txt
U Depart/TD/TD2/lecture
U Depart/TD/TD2/td2.html
cvs checkout: Updating Depart/TD/TD3
U Depart/TD/TD3/aa.rtf
U Depart/TD/TD3/simple_man_ar.html
U Depart/TD/TD3/simple_man_find.html
U Depart/TD/TD3/simple_man_grep.html
```

```
U Depart/TD/TD3/simple_man_tar.html
U Depart/TD/TD3/td3.html
```

3.9 Repentir

Après avoir extrait une ancienne version, nous pouvons éprouver un repentir et décider de recommencer notre travail à partir de celle-ci. CVS permet de replacer la version antérieure à la suite de la dernière version dans la base. La marche à suivre consiste à

1. mettre à jour la dernière version,
2. remplacer le fichier par sa version antérieure,
3. replacer finalement celle-ci dans la base.

Supposons ici que l'on veuille repartir de la première version de `td2.html`. Commençons par la première étape :

```
~/WWW$ cd TD/TD2
~/WWW/TD/TD2$ cvs update -A td2.html
M td2.html
```

La première chose à remarquer est l'option `-A`. Elle permet de lever d'éventuelles contraintes posées sur le fichier (nous n'avons pas abordé ce sujet dans ce tutoriel). La seconde est qu'effectivement nous n'avons pas archivé notre dernière modification de `td2.html` (cf. section 3.4). Donc :

```
~/WWW/TD/TD2$ cvs commit -m "Ajout de quelques lignes blanches" td2.html
Checking in td2.html;
/home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v  <--  td2.html
new revision: 1.5; previous revision: 1.4
done
```

La seconde étape est :

```
~/WWW/TD/TD2$ rm -f td2.html
~/WWW/TD/TD2$ cvs update -p -r 1.1 td2.html > td2.html
===================================================================
Checking out td2.html
RCS:  /home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v
VERS: 1.1
***************
```

L'option `-p` envoie le résultat de la commande update sur la sortie standard, dans le but de contourner d'éventuelles contraintes sur le fichier concerné. C'est pourquoi il faut rediriger la sortie standard dans un nouveau fichier `td2.html`. L'option `-r` précise le numéro de version (Release) que l'on souhaite mettre à jour (ici il s'agit en fait d'une mise au jour).
La dernière étape consiste à archiver la version ainsi obtenue, de façon à ce qu'elle devienne la version courante :

```
~/WWW/TD/TD2$ cvs update td2.html
M td2.html
~/WWW/TD/TD2$ cvs commit -m "Retour la version initiale" td2.html
Checking in td2.html;
```

```
/home/der_gi/rinderkn/cvsroot/WWW/TD/TD2/td2.html,v  <--  td2.html
new revision: 1.6; previous revision: 1.5
done
```

3.10 Renommage et déplacement

Il n'y a pas de procédure simple (atomique) pour renommer ou déplacer un fichier ou un répertoire. La règle générale et sûre consiste à renommer le fichier visé dans la copie de travail, puis d'effectuer les opérations de suppression, suivi de celles d'ajout. En résumé, la procédure est :

1. renommer le fichier,
2. informer CVS de notre intention de supprimer le fichier ancien,
3. informer CVS de notre volonté d'ajouter le fichier nouveau,
4. archiver ce dernier.

Supposons que nous voulions renommer le fichier `copyright` en `copyleft` :

```
~/WWW/TD$ mv copyright copyleft
~/WWW/TD$ cvs remove copyright
cvs remove: scheduling ‘copyright’ for removal
cvs remove: use ’cvs commit’ to remove this file permanently
~/WWW/TD$ cvs add copyleft
cvs add: scheduling file ‘copyleft’ for addition
cvs add: use ’cvs commit’ to add this file permanently
~/WWW/TD$ cvs commit -m "Renamed copyright to copyleft" copyleft
RCS file: /home/der_gi/rinderkn/cvsroot/WWW/TD/copyleft,v
done
Checking in copyleft;
/home/der_gi/rinderkn/cvsroot/WWW/TD/copyleft,v  <--  copyleft
initial revision: 1.1
done
```

Références

[1] Frédéric Lepied. *CVS — Configuration et mise en œuvre*. Éditions O'Reilly, 2000. ISBN 2-84177-057-5.