

CVS

TENENBAUM Sébastien 180747 - HOANG My 180748

Introduction

CVS permet de gérer les programmes sources d'un projet mené par plusieurs programmeurs. Le système conserve dans un endroit centralisé appelé **Repository** les sources considérés comme finalisés.

Chaque programmeur utilise un répertoire dans son arborescence personnelle pour développer. Les sources produits dans ces répertoires sont appelés **fichiers de travail**.

Quand un programmeur souhaite intégrer ses dernières modifications au projet, il demande à CVS de valider les changements. CVS analyse alors les différences avec la précédente révision, vérifie la cohérence des modifications avec celles apportées par les autres programmeurs, puis enregistre ces différences dans le **repository** sous un nouveau **numéro de révision**.

Cas où il n'y a qu'une seule machine

Voyons un exemple simple avec des développeurs travaillant sur la même machine.

On crée le répertoire où CVS va stocker ses fichiers

```
mkdir Repository  
export CVSROOT=/var/tmp/essais-CVS/Repository  
cvsexec init
```

On crée un répertoire de travail, ici pour le projet "toto"

```
mkdir $CVSROOT/toto
```

On lui donne les bonnes protections

```
chgrp -R toto $CVSROOT/toto  
chmod g+w -R $CVSROOT/toto
```

Maintenant, le premier développeur va pouvoir travailler. Il commence

par sortir ("co" = check out) le répertoire.

```
cvsexec co toto
```

Puis, il crée des fichiers, qu'il peut ajouter à CVS et enregistrer

("commit") quant il en est satisfait.

cd toto/

emacs main.c

cvcs add main.c

cvcs commit -m "Premiere version"

Un autre développeur en fait autant. Pour se synchroniser, le

premier développeur met à jour ("update") son répertoire :

cvcs update

C'est cette commande qui se charge de "fusionner" les fichiers. Par exemple, si main.c a été modifié par l'autre développeur avant, cvcs update écrira ses modifications dans son propre fichier main.c

Il voit que main.c a été modifié. Il peut regarder l'historique :

cvcs log main.c

Puis il le change, mais l'autre développeur en fait autant en même

temps. cette fois, il y a conflit :

cvcs commit -m "Declaration de i"

cvcs commit: Examining .

cvcs commit: Up-to-date check failed for `main.c'

cvcs [commit aborted]: correct above errors first!

ludwigV:/var/tmp/essais-CVS/Work/toto> cvcs update

cvcs update: Updating .

RCS file: /var/tmp/essais-CVS/Repository/toto/main.c,v

retrieving revision 1.2

retrieving revision 1.3

Merging differences between 1.2 and 1.3 into main.c

M main.c

Ici, CVS a réussi à résoudre le conflit seul. Il faut parfois

l'aider.

Il suffit de refaire un cvcs update, et il y aura une fusion du fichier. Si les deux développeurs ont modifiés la même ligne, au prochain update, les fichiers main.c prendront compte de ce cas et noteront les 2 modifications. Aux développeurs de se concerter sur laquelle des 2 modifications à garder. :)

Cas où plusieurs machines sont utilisées

Il faut savoir en plus se connecter à la machine disposant du Repository pour pouvoir télécharger et updater les sources.

On suppose que le Repository et le répertoire de travail ont été préalablement créés sur la machine serveur, et que les droits d'accès à ces répertoires ont été modifiés.

- 1 **Récupération d'un projet par SSH. Cela nécessite la présence d'un compte sur le serveur. Cette connexion est sécurisée, mais demande le passwd à chaque commande lancée. On peut contourner ce problème avec une authentification RSA (voir annexe).**
Préciser quel client SSH sera utilisé : `export CVS_RSH=/usr/bin/ssh`
Récupérer les fichiers du projet dans le répertoire courant : `cvscvs -d:ext:nomserveur:/cvsroot checkout nomduprojet`
Les commandes sont ensuite les memes (cvs update, cvs commit...).
- 2 Récupérer les sources sans avoir un compte sur le serveur (peu sécurisée) : `cvscvs -d:pserver:user@server:chemin-vers-le-repository login`
- 3 Connexion anonyme : `export CVSROOT=:pserver:anonymous@cvs.qref.sf.net:/cvsroot/qref`

Alternatives

Création d'un compte Sourceforge, Savaanaah (en plus : mailling list, de forum, suivi de bugs).

Création d'un serveur GForge (idem que Sourceforge, mais sur sa propre machine : pas besoin d'inscription).

Cas où il n'y a qu'une seule machine (version longue)

Voyons un exemple simple avec des développeurs travaillant sur la même machine.

On crée le répertoire où CVS va stocker ses fichiers

```
ludwigV:/var/tmp/essais-CVS> mkdir Repository
```

```
ludwigV:/var/tmp/essais-CVS/Work> export CVSROOT=/var/tmp/essais-CVS/Repository
```

```
ludwigV:/var/tmp/essais-CVS/Work> cvs init
```

On crée un répertoire de travail, ici pour le projet "toto"

```
ludwigV:/var/tmp/essais-CVS/Work> mkdir $CVSROOT/toto
```

On lui donne les bonnes protections

```
ludwigV:/var/tmp/essais-CVS/Work> chgrp -R toto $CVSROOT/toto
```

```
ludwigV:/var/tmp/essais-CVS/Work> chmod g+w -R $CVSROOT/toto
```

Maintenant, le premier développeur va pouvoir travailler. Il commence

par sortir ("co" = check out) le répertoire.

```
ludwigV:/var/tmp/essais-CVS/Work> cvs co toto
```

```
cvs checkout: Updating toto
```

```
ludwigV:/var/tmp/essais-CVS/Work> cd toto/
```

Puis, il crée des fichiers, qu'il peut ajouter à CVS et enregistrer

("commit") quant il en est satisfait.

```
ludwigV:/var/tmp/essais-CVS/Work/toto> emacs main.c
```

```
ludwigV:/var/tmp/essais-CVS/Work/toto> cvs add main.c
```

```
cvs add: scheduling file `main.c' for addition
```

```
cvs add: use 'cvs commit' to add this file permanently
```

```
ludwigV:/var/tmp/essais-CVS/Work/toto> cvs commit -m "Premiere version"
```

```
cvs commit: Examining .
```

```
RCS file: /var/tmp/essais-CVS/Repository/toto/main.c,v
```

```
done
```

```
Checking in main.c;
```

```
/var/tmp/essais-CVS/Repository/toto/main.c,v <-- main.c
```

```
initial revision: 1.1
```

```
done
```

Un autre développeur en fait autant. Pour se synchroniser, le

premier développeur met à jour ("update") son répertoire :

```
ludwigV:/var/tmp/essais-CVS/Work/toto> cvs update
```

```
cvs update: Updating .
```

```
U main.c
```

Il voit que main.c a été modifié. Il peut regarder l'historique :

```
ludwigV:/var/tmp/essais-CVS/Work/toto> cvs log main.c
```

```
RCS file: /var/tmp/essais-CVS/Repository/toto/main.c,v
```

```
Working file:
```

```
main.c head: 1.2 branch: locks: strict access list: symbolic names:
```

```
keyword substitution: kv total revisions: 2; selected revisions: 2
```

```
description: ----- revision 1.2 date:
```

```
1999/03/20 20:10:38; author: eve; state: Exp; lines: +1 -0 Appel de
```

```
toto() ajoute ----- revision 1.1 date:
```

1999/03/20 20:07:21; author: stephane; state: Exp; Premiere version

Puis il le change, mais l'autre développeur en fait autant en même

temps. cette fois, il y a conflit :

```
ludwigV:/var/tmp/essais-CVS/Work/toto> cvs commit -m "Declaration de i"
```

```
cvs commit: Examining .
```

```
cvs commit: Up-to-date check failed for `main.c'
```

```
cvs [commit aborted]: correct above errors first!
```

```
ludwigV:/var/tmp/essais-CVS/Work/toto> cvs update
```

```
cvs update: Updating .
```

```
RCS file: /var/tmp/essais-CVS/Repository/toto/main.c,v
```

```
retrieving revision 1.2
```

```
retrieving revision 1.3
```

```
Merging differences between 1.2 and 1.3 into main.c
```

```
M main.c
```

Ici, CVS a réussi à résoudre le conflit seul. Il faut parfois

l'aider.

Annexe : RSA

On peut éviter de se rappeler le mot de passe pour chaque système distant en utilisant RSAAuthentication (protocole SSH1) ou PubkeyAuthentication (protocole SSH2).

Sur le système distant, le contenu de /etc/ssh/sshd_config doit contenir « RSAAuthentication yes » ou « PubkeyAuthentication yes ».

Générez ensuite les clés d'identification localement et installez la clé publique sur le système distant :

```
$ ssh-keygen      # RSAAuthentication : clé RSA1 pour SSH1
```

```
$ cat .ssh/identity.pub | ssh user1@remote 'cat - >>.ssh/authorized_keys"
```

Sources

<http://www.cvshome.org>

https://www.cvshome.org/new_users.html

<https://www.cvshome.org/docs/overview.html>

<https://www.cvshome.org/docs/blandy.html>

<https://www.cvshome.org/docs/manual/>

<http://qref.sourceforge.net/Debian/reference/ch-vcs.fr.html>