

## 0.1 Utilisation de CVS

### 0.1.1 1. Qu'est-ce que CVS, à quoi ça sert ?

CVS signifie Concurrent Versions System. C'est un utilitaire qui facilite le développement de projet en équipe, en conservant l'historique des modifications apportées à chaque fichier. Il travaille essentiellement sur des fichiers **textes**, c'est-à-dire qu'il ne sert à rien si vous travaillez sur des images ou sur des fichiers compressés. Cependant, il peut être très utile pour les projet de programmation (C, JAVA, HTML, ...), de rédaction (au format texte, donc pas de .doc), et tous les fichiers XML.

### 0.1.2 2. Installation du client cvs

CVS est surement inclu dans les packages de votre distribution. Pour les inconditionnels des sources, vous pouvez le télécharger sur [www.cvshome.org](http://www.cvshome.org)<sup>1</sup>.

### 0.1.3 3. CVS côté client

#### 0.1.4 3.1 Exemple d'utilisation classique

Le mieux est d'étudier un cas concret (le nom du serveur est fictif).

#### Récupération du projet

Tout d'abord, configurez les variables de cvs (à mettre éventuellement dans le ~/.bashrc ou ~/.bash\_profile). Par exemple :

```
export CVSROOT=morgoth@cvs.creatta.com :/var/cvs/cvsroot
export CVS_RSH=ssh
export CVSEEDITOR=vim
```

Fermez puis relancez votre terminal afin que ce dernier lise ces variables. Ensuite, récupérez une copie de travail du projet. Par exemple :

```
$ cvs checkout CreatTA_java
cvs checkout : Updating CreatTA_java
U CreatTA_java/TestCreation.java
...
```

Notez le "U" devant le nom du fichier. Cela signifie mis-à-jour (Update).

#### Modifications

Vous pouvez faire ensuite des modifications comme vous le souhaitez dans le dossier (ici CreatTA\_java). Par contre, ne modifiez pas les fichiers inclus dans les répertoires CVS. Ils sont nécessaires au bon fonctionnement de CVS.

```
$ cd CreatTA_java
$ vim TestCreation.java
```

Il vous faut maintenant valider ces modifications sur le serveur.

---

<sup>1</sup><https://www.cvshome.org>

```
$ cvs commit nom du fichier
```

ou

```
$ cvs commit
```

CVS vérifie que personne n'a modifié le fichier depuis que vous l'avez récupéré, et ajoute vos modifications en incrémentant le numéro de version du fichier. En cas de conflit, voyez plus bas.

```
$ cvs commit TestCreation.java
```

CVS vous demande de remplir un petit message expliquant vos modifications. N'oubliez pas de sauver ce message.

```
Checking in TestCreation.java ;
/var/cvs/cvsroot/CreatTA_java/TestCreation.java,v <- TestCreation.java
new revision : 1.7 ; previous revision : 1.6
done
```

Ici, tout s'est bien passé. Si on veut ensuite récupérer toutes les dernières versions des fichiers :

```
$ cvs update
cvs update : Updating .
```

Ici, tout est à jour, donc aucun fichier n'est récupéré.

### **0.1.5 3.2 Commandes usuelles**

Récupérer les derniers fichiers du projet dans un dossier du même nom :

```
$ cvs checkout nom du module
```

Valider les changements effectués :

```
$ cvs commit [nom du fichier]
```

Récupérer la dernière version du fichier, en fusionnant éventuellement avec vos modifications (voir Les conflits) :

```
$cvs update [nom du fichier]
```

Ajouter le fichier ou le répertoire au dépôt. Un "commit" est nécessaire pour valider l'ajout :

```
$ cvs add nom de fichier ou de répertoire
```

Retirer du dépôt un fichier ou un répertoire **vide** :

```
$ cvs remove nom de fichier ou de répertoire vide
```

Lister l'ensemble des modifications effectuées sur le fichier :

```
$ cvs log nom du fichier
```



### Accès SSH

L'accès à CVS par SSH est le seul moyen sécurisé pour autoriser des modifications à distance sur votre dépôt. Il suffit de créer le dépôt comme indiqué ci-dessus, puis de renseigner les variables CVSROOT et CVS\_RSH :

```
$ export CVSROOT=morgoth@Angband.com :/var/cvs/cvsroot
$ export CVS_RSH=ssh
```

En remplaçant "morgoth" par votre login ssh, et "Angband.com" par votre serveur.

### Accès :pserver

L'accès par :pserver permet d'autoriser des personnes à se connecter de manière anonyme en lecture seule à votre dépôt. Cette méthode est utilisée par un grand nombre de projets libres. Tout d'abord, vous devez vous assurer d'avoir un compte anonyme (par exemple) et de lui autoriser l'accès en lecture à votre dépôt CVS. Pour ce faire, tapez dans un terminal :

```
# (grep anonymous /etc/passwd || useradd anonymous -s /bin/false)
# echo anonymous : > /var/cvs/cvsroot/CVSROOT/passwd
# echo anonymous > /var/cvs/cvsroot/CVSROOT/readers
```

Maintenant, il ne vous reste plus qu'à activer le service :pserver et à le configurer.

- **Pour Mandriva :** Vous n'avez rien à faire, ou presque. Tout d'abord, vérifiez que le fichier `/etc/cvs/cvs.conf` contient bien : `CVS_REPOS="/var/cvs/cvsroot"` (ou autre si vous n'avez pas choisi ce répertoire). Ensuite, activez :pserver en éditant le fichier `/etc/xinetd.d/cvs` et en mettant `disable = no`. Ensuite, un petit redémarrage de xinetd :

```
# service xinetd restart
```

- **Pour les autres :** Ajoutez le service à xinetd (si il n'existe pas déjà) :

```
# cat » /etc/xinetd.conf « "EOF"
service cvspserver
{
port = 2401
socket_type = stream
protocol = tcp
wait = no
user = root
passenv = PATH
server = /usr/bin/cvs
server_args = -f --allow-root=/var/cvs/cvsroot pserver
}
EOF
```

Puis forcez xinetd à relire sa configuration par : `# killall -HUP xinetd`

Maintenant, pour accéder de manière anonyme à votre dépôt, il suffit de taper la commande suivante (ajustée à votre configuration) :

```
$ cvs -d :pserver :anonymous@nom_serveur :/var/cvs/cvsroot checkout nom_projet
```

### Ajouter un nouveau projet

Pour ajouter un nouveau projet au dépôt CVS, allez dans le répertoire qui contient les sources de votre projet :

```
$ cd mon-projet  
$ cvs import projet-XYZ Morgoth initial
```

Où "projet-XYZ" sera le nom de votre module, et "initial" le tag.

#### 0.1.9 5. Interfaces graphiques

Il existe de nombreuses interfaces graphiques pour cvs. Citons TkCvs (que je n'ai jamais essayé), Cervisia (l'interface CVS de KDE), et enfin WinCVS (sous Windows). CVS s'intègre dans de nombreux outils de développement, comme Eclipse ou Netbeans.