

Table des matières

1	Préambule	2
2	commandes universelles subversion	2
2.1	Mettre à jour sa copie locale : update	2
2.2	Ajouter des fichiers ou dossiers au projet : add	2
2.3	Supprimer des fichiers ou dossiers au projet : delete	2
2.4	Lister les fichiers du dépôt : list	3
2.5	Revenir à une version précédente	3
2.6	Historique des modifications : log	3
3	Subversion sur internet	4
3.1	Récupérer le contenu du projet	4
3.2	Appliquer les changements qu'on vient de faire localement : commit	4
4	Subversion localement	4
4.1	Création du projet	4
4.2	Récupérer le contenu du projet	4
4.3	Appliquer les changements qu'on vient de faire localement : commit	4
5	Importer un arbre subversion	5
	Index	6

1 Préambule

Subversion permet de gérer un projet (de programmation généralement) et de garder en mémoire l'historique de toutes les versions d'un ensemble de fichiers. Il permet de gérer un projet à plusieurs, de programmer afin de pouvoir revenir en arrière, comparer avec d'anciennes versions et cie.

Le principe est d'avoir un serveur subversion (un seul possible) qui va garder en mémoire l'historique de toutes les versions et un client subversion (plusieurs possibles) qui vont se connecter au serveur pour mettre à jour la version des fichiers ou en récupérer les dernières versions.

Remarque : Il est possible que le serveur soit lui aussi client, dans le cas où il n'y aurait qu'un seul développeur et qu'on ne souhaite pas passer par internet.

2 commandes universelles subversion

Ici, je note les commandes qui sont valables à la fois pour svn installé sur un serveur internet, ou sur une machine locale pour un usage personnel

2.1 Mettre à jour sa copie locale : update

```
svn update
```

Pour celà, il faut que le dossier dans lequel on se trouve ait déjà été défini comme un dossier svn via un *checkout* (voir [§ 3.1])

2.2 Ajouter des fichiers ou dossiers au projet : add

Pour ajouter des fichiers il faut faire :

```
svn add latex/ vim/
```

où **latex/** et **vim/** sont deux dossiers existant dans le dossier local de référence

Remarque : Au cas où ça serait pas clair. J'ai créé un dossier **/home/autiwa/Formulaires** grâce à [§ 3.1]. Dans ce dossier, j'ai créé et rempli à la main les sous-dossiers **latex/** et **vim/**. Maintenant, grâce à la commande ci-dessus, je définis ces sous-dossiers comme étant rattachés au projet. En faisant ainsi le contenu est rajouté récursivement.



Cette commande n'agit que sur le répertoire local (la *working copy*). Il faut ensuite faire un **commit** (voir [§ 3.2]) pour valider les changements sur le serveur.

2.3 Supprimer des fichiers ou dossiers au projet : delete

Pour supprimer des fichiers il faut faire :

```
svn delete latex/ vim/
```

où **latex/** et **vim/** sont deux dossiers existant dans le dossier local de référence.



Cette commande n'agit que sur le répertoire local (la *working copy*). Il faut ensuite faire un **commit** (voir [§ 3.2]) pour valider les changements sur le serveur.

Remarque : Supprimer manuellement dans le dossier courant des fichiers n'aura aucune incidence sur le dépôt subversion (je crois). Il faut faire avec **svn delete** si on veut modifier quelque chose.

2.4 Lister les fichiers du dépôt : list

```
svn list
```

va lister les fichiers présent dans le dépôt, mais par défaut, il fait les fichiers présent lors de la première révision.

Pour spécifier un numéro de révision (pour peu que les fichiers aient changés entre temps) :

```
svn list -r 7
```

pour afficher la liste des fichiers de la révision 7.

2.5 Revenir à une version précédente

Pour annuler les modifications locales et revenir à la dernière révision :

```
svn revert *.f90
```

va remettre en l'état tous les fichiers du svn qui sont présents dans le dossier courant (on peut aussi ne préciser qu'un seul fichier).

Si on souhaite forcer la mise à jour vers une révision plus ancienne (en cas de bug) il suffit de préciser le numéro de la révision

```
svn update --force -r39
```

pour revenir à la révision 39 ou

```
svn update --force -r39 fichier.f90
```

Pour ne changer qu'un seul fichier en particulier.

Remarque : Si la révision 39 est la dernière et qu'elle a été envoyée depuis le répertoire local, il est fort probable que la commande ci-dessus ne change rien. Il faut alors utiliser la commande **revert -force** permet de forcer les fichiers de la révision et d'écraser toute éventuelle modification locale. Dans le doute, tester sans cette option, quitte à la rajouter ensuite.

On peut aussi supprimer les modifications de certaines révisions, à condition que celles-ci soient indépendantes des révisions plus récentes :

```
svn merge -r2349:2345
```

ou

```
svn merge -r2349:2345 fichier.f90
```

pour ne modifier qu'un seul fichier (ou un ensemble de fichier restreint).

Cette commande, qui doit être suivie d'un **commit**, va créer une nouvelle révision qui annule les commit entre 2345 et 2349 pour créer une révision 2350.

2.6 Historique des modifications : log

La commande **log** permet de voir un historique des commentaires de révision, soit de l'ensemble des fichiers, soit d'un fichier en particulier.

```
svn log
```

affiche l'historique de toutes les révisions (il est possible de spécifier un nombre maximal de lignes que l'on souhaite (les 10 dernières révisions par exemple via l'option **-l 10**).

```
svn log main.f90
```

va afficher l'historique des modifications du fichier **main.f90**, c'est à dire que n'apparaîtront que les commentaires des révisions où le fichier **main.f90** a été modifié.

3 Subversion sur internet

Je vais prendre l'exemple de google code, qui est celui que j'ai choisi et que je suis en train d'apprendre.

3.1 Récupérer le contenu du projet

Une fois le projet créé (sur la page <http://code.google.com/hosting/createProject>), il faut faire :

```
svn checkout --username autiwa@gmail.com --password "passwd" \
https://autiwa-tutorials.googlecode.com/svn/trunk/ Formulaires
```

Cette commande permet de récupérer le contenu du projet et de le copier dans un dossier **Formulaires** qui sera créé dans le dossier courant.

DÉFINITION 1 (CHECKOUT)

Opération d'extraction d'une version d'un projet du repository vers un répertoire de travail local.

3.2 Appliquer les changements qu'on vient de faire localement : commit

Pour mettre à jour les versions sur serveur à partir des modifications effectuées localement, il faut :

```
svn commit -m "initialisation" --username autiwa@gmail.com --password passwd
```

où **"initialisation"** est le commentaire qui décrit la mise à jour et les modifications effectuées, **passwd** étant le mot de passe associé à votre identifiant (ici **autiwa@gmail.com** pour moi).

Remarque : Je n'ai eu besoin d'entrer le mot de passe que la première fois.

4 Subversion localement

Le serveur ET le client seront alors sur la même machine.

4.1 Création du projet

J'ai créé un dossier **svn** dans mon **\$HOME**, puis je fais, dans mon répertoire utilisateur :

```
svnadmin create svn/mercury
```

pour un projet que j'appelle **mercury**.

4.2 Récupérer le contenu du projet

Une fois le projet créé (sur la page <http://code.google.com/hosting/createProject>), il faut faire :

```
svn checkout file:///home/cossou/svn/mercury
```

où **file:///home/cossou/svn/mercury** est une URL qui désigne le dossier **/home/cossou/svn/mercury**. La présence de **file:///** est à ma connaissance indispensable.

Cette commande permet de récupérer le contenu du projet et de le copier dans un dossier **Formulaires** qui sera créé dans le dossier courant.

DÉFINITION 2 (CHECKOUT)

Opération d'extraction d'une version d'un projet du repository vers un répertoire de travail local.

4.3 Appliquer les changements qu'on vient de faire localement : commit

Pour mettre à jour les versions sur serveur à partir des modifications effectuées localement, il faut :

```
svn commit -m "initialisation"
```

où **"initialisation"** est le commentaire qui décrit la mise à jour et les modifications effectuées, **passwd** étant le mot de passe associé à votre identifiant (ici **autiwa@gmail.com** pour moi).

5 Importer un arbre subversion

À la création d'un dépôt subversion (sur internet par exemple) on peut souhaiter importer un arbre subversion déjà existant (localement par exemple). Afin de l'importer, il suffit d'avoir le dépôt de destination entièrement vide (pas initialisé). Il peut donc être nécessaire de faire un *reset*. Puis il faut faire

```
svnsync init --username YOURUSERNAME \  
https://YOURPROJECT.googlecode.com/svn file:///path/to/localrepos  
Copied properties for revision 0.
```

pour initialiser le dépôt. Vous aurez alors un retour de cette forme :

```
$ svnsync init --username YOURUSERNAME \  
https://mercury-90.googlecode.com/svn file:///home/login/svn/mercury  
Copied properties for revision 0.
```

Puis il faut synchroniser toutes les révisions locales sur le nouveau dépôt :

```
svnsync sync --username YOURUSERNAME https://YOURPROJECT.googlecode.com/svn  
Committed revision 1.  
Copied properties for revision 1.  
Committed revision 2.  
Copied properties for revision 2.  
[...]
```

Remarque : J'ai eu droit à un

svnsync: Le serveur a envoyé une valeur inattendue (502 Bad Gateway)
en réponse à la requête PROPFIND pour '/svn/!svn/bln/21'

J'ai re-exécuté la même commande de synchronisation quelques minutes après et j'ai pu synchroniser, je ne sais pas d'où le problème venait.

Index

Ajouter des fichiers, [2](#)

commande svn

- add, [2](#)
- checkout, [2](#), [4](#)
- commit, [4](#)
- delete, [2](#)
- list, [3](#)
- log, [3](#)
- merge, [3](#)
- revert, [3](#)
- update, [2](#), [3](#)

historique des modifications, [3](#)

Lister les fichiers du dépôt, [3](#)

Mise à jour du dépôt, [2](#)

mise à jour local -> serveur, [4](#)

Revenir à une version précédente, [3](#)

Supprimer des fichiers, [2](#)