Subversion pour LaTeX

Y. Morère

Octobre 2006

Résumé

Cet article présente brièvement l'utilisation de subversion dans un cadre très précis : Avoir ses documents/projets toujours à jour au travail et à la maison sans passer par une clé usb, ou une archive que l'on déplace sur une machine.

Table des matières

1 Introduction

Comme je désire travailler chez moi et à l'université sur mes documents, il faut que je puisse les mettre à jour d'un coté et de l'autre d'une manière simple et rapide. Je dois aussi tenir compte de l'infrastructure réseau mise en place à l'université et notamment de la politique de sécurité qui impose un minimum de ports ouverts vers l'extérieur sur les machines clientes.

Ainsi dans mon bureau (université), ma machine de travail ne possède aucun accès depuis l'extérieur (elle fait fonctionner nombres de services qui sont autant d'attaques potentielles), par contre une autre petite machine possède uniquement un accès SSH (port 22) depuis l'extérieur. Ceci me permet de me connecter sur ma machine de bureau depuis chez moi.

Le schéma suivant résume bien la situation au niveau du réseau.

Je vais donc utiliser ce seul accès par le port 22, pour effectuer mes synchronisations.

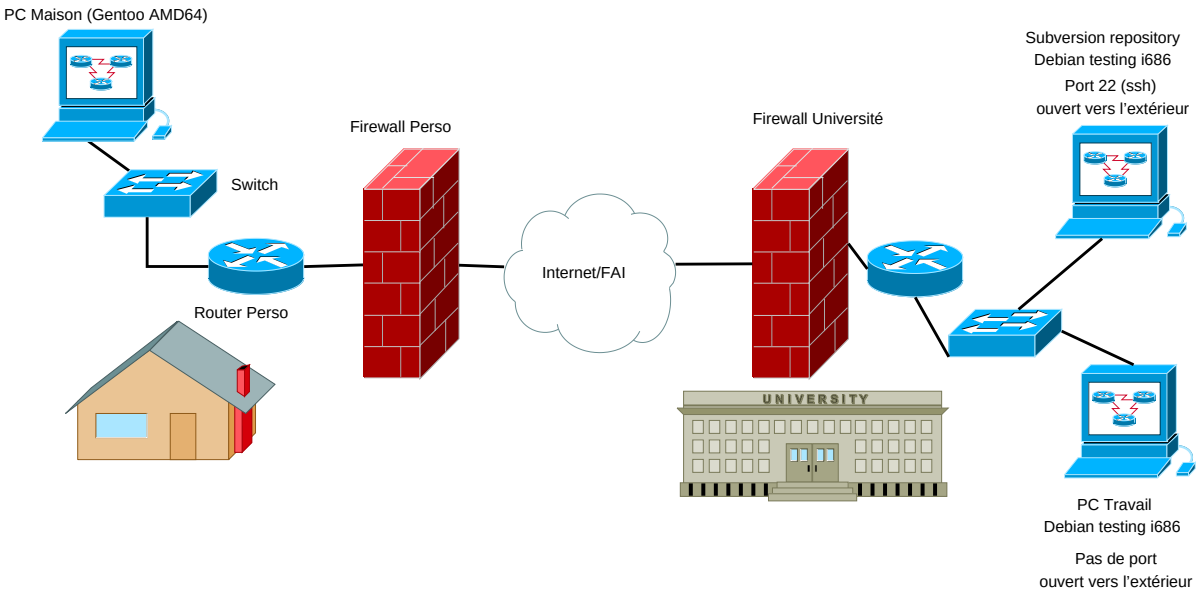

FIG. 1 – Infrastructure du réseau

2 L'existant

Jusqu'à présent, pour maintenir des versions à jour sur chaque machine, j'emportais une archive du répertoire des documents/projets sur une clé USB, ou je le déposais via `scp` sur la machine intermédiaire visible de l'extérieur.

Mais cela peut se réveler long, et souvent, seulement quelques fichiers ont besoin d'être mis à jour. Ce qui est donc une perte de temps. De plus les manipulations deviennent vite fastidieuses :
▷ archivage du répertoire sur la machine source,
▷ copie de l'archive sur la machine intermédiaire,
▷ copie de l'archive sur la machine de destination
▷ désarchivage sur la machine de destination
De plus si le fichier est assez gros, le temp de transfert (surtout du coté maison) n'est pas négligeable.

Il est évident que je pourrais utiliser un outil de type `rsync` comme dans le cas du serveur de sauvegarde http://yann.morere.free.fr/article.php3?id_article=74, mais je n'aurais pas de système de gestion de version et ne pourrais pas revenir en arrière.

Subversion (parfois abrégé SVN) est un logiciel informatique de gestion de versionnement semble tout indiqué.

La gestion de version (en anglais revision control) est une activité qui consiste à maintenir l'ensemble des versions d'un logiciel. Essentiellement utilisée dans le domaine de la création de logiciels, elle est surtout concernée par le code source ; mais elle peut être utilisée pour tout type de document informatique.

Subversion est un logiciel de gestion de sources et de contrôle de versions. Ce type de programmes a plusieurs fonctions, notamment :
▷ garder un historique des différentes versions des fichiers d'un projet ;
▷ permettre le retour à une version antérieure quelconque ;
▷ garder un historique des modifications avec leur nature, leur date, leur auteur... ;
▷ permettre un accès souple à ces fichiers, en local ou via un réseau ;
▷ permettre à des utilisateurs distincts et souvent distants de travailler ensemble sur les mêmes fichiers.
On pourra justifier rapidement le choix de Subversion par les arguments suivants :
▷ il est multiplateforme ;
▷ il s'agit d'un logiciel libre ;
▷ il fonctionne de manière centralisée ;
▷ son utilisation et son administration sont plus faciles que CVS ;
▷ il supporte plusieurs modes d'accès distants, dont SSH et WebDAV via Apache.
C'est pour cela que je l'utilise pour gérer mes documents rédigés avec LaTeX.

3 Subversion

Le site principal de subversion se trouve à l'adresse http://subversion.tigris.org/. La documentation principale se trouve à l'adresse http://svnbook.red-bean.com/. On pourra trouver une bonne présentation de l'utilisation de subversion à l'adresse http://www.crium.univ-metz.fr/docs/devel/svn/, notamment pour l'utilisation des `tags`.

Subversion est un système de gestion de versions, on commence par créer un dépôt qui va servir à stocker un projet. Une fois ce dépôt créé et rempli des fichiers du projet, chaque personne y ayant accès peut charger le contenu sur son disque local. Ensuite on peut effectuer des opérations telles que modifier des fichiers, en effacer, compiler des sources, sans affecter le contenu du dépôt original.

Un cycle de subversion se déroule comme suit :
▷ Créer un dépôt. `svnadmin create`
▷ Importer un projet pour gestion dans le dépôt préalablement créé.`svn import`
▷ Récupérer la copie d'un projet depuis un dépôt. `svn checkout`
▷ Modifier & créer le contenu du projet. `svn add, svn copy, svn move, svn delete`
▷ Consultation des modifications. `svn status`
▷ Soumission des modifications. `svn commit`
▷ Consultation des logs de subversion. `svn log`
▷ Mettre à jour son dépôt local. `svn update`
Pour connaître les commandes disponibles pour svn.

```
$ svn help
```

4 Notions générales

4.1 Dépôt (repository)

Un dépôt Subversion est l'emplacement central où sont stockées toutes les données relatives aux projets gérés. Le dépôt est accédé via une URL locale ou distante.

Le dépôt contient l'historique des versions des fichiers stockés, les logs enregistrés lors des modifications, les dates et auteurs de ces modifications, etc.

S'il a été configuré comme tel, un dépôt apparaît de l'extérieur comme un système de fichiers composé de répertoires au sein desquels on peut naviguer, lire et écrire selon les permissions accordées. Dans le cas d'une configuration type base de données, la navigation n'est possible qu'à travers d'un outil dédié.

Le tableau 1 résume les différences entre une configuration type "Base de données" et une configuration type "système de fichiers".

4.2 Projets

Au sein d'un dépôt se trouvent un ou plusieurs projets. À chaque projet correspond en général un répertoire situé à la racine du dépôt et qui contient lui-même les fichiers et dossiers du projet proprement dit.

Il est aussi possible de stocker toutes arborescences dans une base de données de type Berkeley. En fait nous avons la possibilité de stocker soit dans une base de données soit dans une arborescence du système de fichier de manière standard. Avec la version 1.3 de subversion Debian, l'option choisie par défaut est la base de données de type Berkeley.

Exemple d'arborescence :

```
(dépôt)--+--/svn_for_latex--+--/trunk   (doc LaTeX)
         |                  |
         |                  +--/branches
         |                  |
         |                  +--/tags
         |
         +--/gestgard-------+--/trunk  (projet php/mysql)
         |                  |
         |                  +--/branches
```

Caractéristiques	Berkeley DB (`bdb`)	FSFS (`fsfs`)
Sensibilité aux interruptions	importante ; les problèmes de crash et de permissions peuvent planter la base, et nécessiter une procédure de restauration.	pas sensible.
Utilisable en montage lecture seul	non	oui
Stockage indépendant de la plate-forme	non	oui
Utilisable par un système de fichier réseau	non	oui
Taille du dépôt	légèrement plus grand	légèrement plus petit
Extensibilité : nombre d'arbres de révision	Base de données ; pas de problèmes	certains anciens systèmes de fichiers ne peuvent pas gérer plus de quelques milliers d'entrées dans un seul répertoire.
Extensibilité : répertoire avec un grand nombre de fichiers	plus lent	plus rapide
Rapidité : vérification de mise à jour	plus rapide	plus lent
Rapidité : révision importante	plus lent, mais le travail est terminé	commit plus rapide mais les délais de finalisation peuvent déconnecter le client
Support des permissions de groupe	sensible aux problèmes d'umask utilisateur ; meilleur si accès par un seul utilisateur.	gère les problèmes umask
Maturité du code	utilisé depuis 2001	utilisé depuis 2004

Tab. 1 – Repository Data Store Comparison

```
|                  |
|                  +--/tags
|
+--/gtkrsynconfig------+--/trunk  (projet gtk+/C)
                       |
                       +--/branches
                       |
                       +--/tags
```

4.3 Copie de travail (working copy)

La copie de travail est un répertoire situé en local sur le poste de l'utilisateur et qui contient une copie d'une révision donnée des fichiers du dépôt. C'est cette copie qui sert de base de travail et qui est modifiée en local avant d'être importée (sauvegardée) vers le dépôt.

4.4 Révisions

Chaque modification faite au dépôt constitue une révision. Le numéro de révision commence à 1 et augmente de 1 à chaque opération. Sa valeur n'a aucune importance, mais c'est un indicateur qui permet de revenir à une version donnée d'un ou plusieurs fichiers.

4.5 Opérations

4.5.1 `checkout`

Le checkout est l'opération qui consiste à récupérer pour la première fois les fichiers déjà existant au sein d'un projet du dépôt. Cette opération ne se fait en général qu'une fois par projet.
Le résultat est une copie de travail.

4.5.2 `import`

L'import est l'opération inverse du checkout. Elle consiste à placer dans le dépôt des fichiers locaux déjà existants pour y créer un nouveau projet. Cette opération ne se fait en général qu'une fois par projet.

4.5.3 `update`

L'update consiste à synchroniser la copie de travail locale avec le dépôt en récupérant la dernière version des fichiers du dépôt.
C'est à cette occasion que des conflits de version peuvent apparaître.

4.5.4 `commit`

Un commit est l'opération inverse d'un update. Elle consiste à mettre à jour le dépôt à partir de la copie de travail locale. Une nouvelle révision est alors créée. Un log (simple message texte contenant une description des modifications effectuées) doit être saisi à cette occasion. À noter que pour qu'un commit soit possible, il faut que la copie de travail corresponde à la dernière version du dépôt (modifications locales exceptées). Si ce n'est pas le cas, il est nécessaire d'effectuer d'abord un update et de résoudre les conflits éventuels avant de réessayer le commit.

5 Installation sur Debian

Pour installer subversion et quelques outils associés.

```
$ aptitude install subversion subversion-tools
```

Pour utiliser subversion via le web, il existe un module apache2 qui permet d'utiliser subversion. Il y a aussi des logiciels pour voir les dépôts en ligne tel que ViewSVN ou websvn.

```
$ aptitude install libapache2-svn
```

Le module est chargé dans `/etc/apache2/mods-enabled/dav_svn.conf`. Il faut modifier la configuration du module de subversion pour qu'il corresponde à vos besoin.
Dans mon cas, ce dernier est inutile, car je n'ai pas de serveur Apache sur la machine du dépôt.

6 Installation sur Gentoo

```
# emerge svn
```

Quelques logiciels de gestion en mode graphiques, dont esvn, rapidsvn.

```
$ aptitude install esvn rapidsvn
```

7 Principales commande de svn

Commande	Signification
add	Déclare l'ajout d'une nouvelle ressource pour le prochain commit
blame	Permet de savoir quel contributeur a soumis les lignes d'un fichier
checkout (co)	Récupère en local une révision ainsi que ses méta-données depuis le dépôt
cleanup	Nettoie la copie locale pour la remettre dans un état stable
commit (ci)	Enregistre les modifications locales dans le dépôt créant ainsi une nouvelle révision
copy	Copie des ressources à un autre emplacement (localement ou dans le dépôt)
delete	Déclare la suppression d'une ressource existante pour le prochain commit (ou supprime directement une ressource du dépôt
diff	Calcule la différence entre deux révisions (permet de créer un patch à appliquer sur une copie locale)
export	Récupère une version sans méta-données depuis le dépôt ou la copie locale
import	Envoie une arborescence locale vers le dépôt
info	Donne les informations sur l'origine de la copie locale
log	Donne les messages de commit d'une ressource
merge	Calcule la différence entre deux versions et applique cette différence à la copie locale
move	Déclare le déplacement d'une ressource
resolved	Permet de déclarer un conflit de modifications comme résolu
revert	Revient à une révision donnée d'une ressource. Les modifications locales sont écrasées.
status	Indique les changements qui ont été effectués
update (up)	Met à jour la copie locale existante depuis la dernière révision disponible sur le dépôt

8 Gestion d'un projet/document LaTeX

8.1 Création du dépôt (repository)

Avant de commencer il faut fournir à svn l'endroit ou il va déposer tous ses fichiers pour fonctionner. Pour cela un simple dossier de l'arborescence suffit. On va pouvoir par exemple utiliser un répertoire de son `HOME`.

On ne crée qu'un seul dépôt, et on y placera tous les projets (LaTeX ou autres).

Pour créer un dépôt sur une machine.

```
$ svnadmin create dossier/sousDossier/depot
```

Par défaut avec la version 1.3 de subversion de la distribution Debian, la structure du dépôt sera une base de données de type Berkeley. Si vous désirez utiliser une arborescence de fichier, il faut l'indiquer à la création du dépôt grâce à l'option `--fs-type`

Plus d'information sur cet option à l'adresse http://svnbook.red-bean.com/nightly/en/svn.reposadmin.html#svn.reposadmin.basics.backends

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svnadmin help create
create: usage : svnadmin create CHEMIN_DÉPÔT

Crée un nouveau dépôt vide à CHEMIN_DÉPÔT.

Options valides:
  --bdb-txn-nosync         : désactive fsync aux propagations de transactions [Berkeley
  --bdb-log-keep           : désactive la suppression automatique des fichiers du journa
  --config-dir ARG         : lit les fichiers de configuration utilisateur dans le réper
  --fs-type ARG            : type de dépôt : 'fsfs' (défaut) ou 'bdb'

yann@biboo ~/latex/graoulug/svn_for_latex $

$ svnadmin create --fs-type fsfs dossier/sousDossier/depot
```

Dans mon cas je me suis connecté via ssh sur la machine et j'ai créé le dépôt comme suit pour un stockage dans une base de données :

```
svnadmin create /home/yann/stockage/svn_repository/
```

8.2 Préparation/importation des projets/documents dans svn

Avant d'importer un projet dans svn, il est conseillé d'organiser le répertoire du projet comme suit :

```
trunk/ pour le code de travail
branches/ pour déployer de nouvelles branches de travail / test
tags/ pour identifier du code distribué
```

Les notions de tronc, de branches et de tags sont assez spécifiques aux logiciels de contrôle de versions. C'est ce qui explique que les arborescences des répertoires de projet contiennent souvent comme premier niveau de sous-répertoires les dossiers trunk, branches et tags. En général, on définit par "tronc" la version centrale du programme, le développement principal "officiel". Une « branche » est en général créée lorsqu'un développement "secondaire" est mis en route, que ce soit pour ajouter une nouvelle fonctionnalité ou parce que certains développeurs souhaitent essayer de prendre une autre direction pour certains aspects du développement. Une branche peut, au bout d'un certain temps, soit être à nouveau fusionnée dans le "tronc", soit disparaître, soit donner lieu à un nouveau programme. La notion de tags correspond en partie à celle de release, c'est à dire de marquage d'une certaine révision du projet comme composant une version du projet. Une fois que le développement a atteint une certaine stabilité, on pourra par exemple créer un tag pour marquer la sortie de la version 1.0. Ceci permettra de revenir facilement à cette version, indépendamment du numéro de révision sous-jacent correspondant. Nous n'entrerons pas dans le détail de ces concepts et commandes ici, mais on peut juste citer que la création de branches ou de tags ne sont en fait que des copies créées par la commande svn copy. La commande svn switch, elle, permet de faire passer la copie de travail d'une branche à une autre.

Il est donc possible de les créer en local, puis d'importer le répertoire du projet, oubien de créer les répertoires `trunk, branches, tags` dans le dépôt et ensuite d'y importer juste le code de developpement du projet.

J'utiliserai le premier cas :

```
$ mkdir projet/trunk projet/tags projet/branches
```

Ensuite il suffit d'importer le projet dans le dépôt. Pour cela on dispose de plusieurs moyens de connexion. Les protocoles disponibles avec Subversion sont les suivants :
▷ `file://` : pour un dépôt sur un disque local
▷ `svn://` : pour un dépôt distant avec un serveur Subversion dont l'authentification est gérée par Subversion.
▷ `svn+ssh://` : pour un dépôt distant avec un serveur Subversion dont l'authentification est gérée par ssh.
▷ `http://` : pour un dépôt distant avec un serveur web.
Pour importer un projet dans le dépôt.

```
$ svn import /projet/ file:///chemin_absolu/vers/depot -m "Premier import"
```

Dans mon cas, j'utiliserais le protocole svn+ssh :

```
svn import ~/latex/graoulug/svn_for_latex
 svn+ssh://login@machine/home/yann/stockage/svn_repository/svn_for_latex
 -m "Import initial Doc LaTeX subversion pour LaTeX"
```

A cet instant la machine du dépôt me demande le mot de passe pour autoriser l'action.
Si vous ne spécifiez pas l'option `-m`, `vim` va s'ouvrir pour que vous puissiez mettre une description de l'action. Il faut aussi faire attention de bien mettre le chemin absolu vers le dépôt, sinon l'importation ne fonctionnera pas.

8.3 Création de la copie de travail

Pour récupérer la base, d'un projet du dépôt local, dans un dossier du système. La commande `cd` nous place dans le répertoire racine de l'utilisateur courant.

```
$ cd
$ mkdir monSVN/projet
$ svn checkout file:///dossier/sousDossier/depot/ monSVN/projet
```

Checkout avec svn+ssh.

```
$ svn checkout svn+ssh://machine/dossier/depot monSVN/projet
commit
```

Dans mon cas, je vais utiliser `svn+ssh` et ne rapatrier que le répertoire `trunk` :

```
$ svn checkout
 svn+ssh://login@machine/home/yann/stockage/svn_repository/svn_for_latex/trunk
 svn_for_latex
```

J'ai maintenant une copie locale du projet/document.

8.4 Ajout d'un fichier au projet

Cela se fait très simplement avec la commande `svn add fichier`

```
yann@biboo ~/latex/graoulug/svn_for_latex $ ls
images             svn_for_latex.dvi  ~svn_for_latex.tex  svn_for_latex.toc
svn_for_latex.aux  svn_for_latex.log  svn_for_latex.tex
yann@biboo ~/latex/graoulug/svn_for_latex $ svn add svn_for_latex.dvi
A  (bin)  svn_for_latex.dvi
yann@biboo ~/latex/graoulug/svn_for_latex $
```

8.5 Mise à jour du dépôt (révision)

Cela se fait très simplement avec la commande `svn commit -m "description"`

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svn commit -m "ajout fichier dvi"
Password:
Ajout   (bin)  svn_for_latex.dvi
Envoi          svn_for_latex.tex
Transmission des données ..
Révision 34 propagée.
yann@biboo ~/latex/graoulug/svn_for_latex $
```

8.6 Effacement de fichier

Cela se fait très simplement avec la commande `svn delete fichier/répertoire`

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svn delete svn_for_latex.dvi
D         svn_for_latex.dvi
yann@biboo ~/latex/graoulug/svn_for_latex $ svn commit -m "effacement fichier dvi"
Password:
Suppression    svn_for_latex.dvi
Envoi          svn_for_latex.tex
Transmission des données .
Révision 35 propagée.
yann@biboo ~/latex/graoulug/svn_for_latex $
```

8.7 Exclusion des fichiers inutiles

Dans le cadre d'une utilisation de svn et LaTeX, il est évident qu'il n'est pas nécessaire de stocker sur le dépôt les fichiers temporaires de travail. Il en est de même pour les fichiers objets et exécutables dans le cadre d'un projet informatique.

Ceci est surtout valable losqu'on veut, par exemple, importer dans le dépôt un projet/document qui est déjà bien avancé et qui possède de nombreux fichiers de travail temporaires.

Pour cela il est possible d'informer subversion des fichiers à exclure du dépôt.

Je vais prendre l'exemple d'un document LaTeX. Je travaille actuellement sur l'article suivant, et je désire l'intégrer à subversion.

```
yann@biboo ~/latex/graoulug/svn_for_latex $ ls
images             svn_for_latex.dvi  ~svn_for_latex.tex  svn_for_latex.toc
svn_for_latex.aux  svn_for_latex.log  svn_for_latex.tex
yann@biboo ~/latex/graoulug/svn_for_latex $
```

Il est évident que les fichiers `svn_for_latex.aux`, `svn_for_latex.log`, `svn_for_latex.dvi` et `svn_for_latex.toc`, générés par une compilation latex, n'ont pas besoin d'être stockés dans le dépôt subversion.

On va donc utiliser la commande `propedit` et la propriété `svn:ignore` sur le répertoire courant.

Celle-ci permet de notifier à svn les fichiers à exclure lors d'une révision, ou d'un import.

Plus d'informations à l'adresse http://svnbook.red-bean.com/nightly/en/svn-book.html#svn.advanced.props.special.ignore.

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svn help propedit propedit (pedit, pe): Édit
usage : 1. propedit PROPNOM CHEMIN...
        2. propedit PROPNOM --revprop -r REV [CIBLE]

  1. Édite une propriété versionnée de la copie de travail.
  2. Édite une propriété non versionnée d'une révision du dépôt.
     CIBLE détermine seulement le dépôt concerné.

Options valides:
  -r [--revision] ARG      : ARG (certaines commandes acceptent également une ét endue A
                      L'argument d'une révision peut être :
                        NUMÉRO       numéro de la révision
                        "{" DATE "}" révision disponible à cette date
                        "HEAD"       dernière révision du dépôt
                        "BASE"       rév. de base de la copie de travail
                        "COMMITTED"  dernière propagation à ou avant BASE
                        "PREV"       révision juste avant COMMITTED
  --revprop               : opère sur la propriéte de révision (utiliser avec - r)
  --username ARG          : précise le nom d'utilisateur ARG
  --password ARG          : précise le mot de passe ARG
  --no-auth-cache         : ne conserve pas les éléments d'authentification
  --non-interactive       : pas de demande interactive
  --encoding ARG          : interprète les caractères comme encodés en ARG
  --editor-cmd ARG        : utilise ARG comme éditeur externe
  --force                 : force l'exécution de l'opération
  --config-dir ARG        : lit les fichiers de configuration utilisateur dans le réper

yann@biboo ~/latex/graoulug/svn_for_latex $
```

La commande suivante permet de configurer la liste des fichiers à exclure pour le projet courant (il faut bien sur être dans le répertoire du projet) :

```
svn propedit svn:ignore .
```

Un VI est alors ouvert dans votre terminal. Il faut alors saisir la liste des fichiers/expressions (patterns). Voici ce que j'ai mis pour mes documents LaTeX.

```
*.toc
*.log
*.out
*.aux
*.lot
*.lof
*.blg
*.bbl
*.odt
*.dvi
*.ps
*.pdf
```

Comme cela, seules les sources et images eps seront déposées.

Lettre	Signification
`A`	The file, directory, or symbolic link item has been scheduled for addition into the repository.
`C`	The file item is in a state of conflict. That is, changes received from the server during an update overlap with local changes that you have in your working copy. You must resolve this conflict before committing your changes to the repository.
`D`	The file, directory, or symbolic link item has been scheduled for deletion from the repository.
`M`	The contents of the file item have been modified.
`G`	The contents of the files (repository and local copy) have been merged during the update.

TAB. 2 – Signification du codage SVN

Il est possible d'y indiquer des répertoires, ce qui permet d'exclure rapidement une partie de l'arborescence du projet (fichiers de travail, test etc...)

Dans le cas ou vous avez, comme moi, importé le projet avant de connaître la commande `svn propedit`, il est nécessaire d'effacer les fichiers à exclure du dépôt.

```
svn delete *.aux *.log *. aux *.lot *.lof *.blg *.bbl *.odt *.tns *.dvi *.ps *.pdf
```

puis faire un `commit`

```
svn commit -m "effacement fichiers inutiles"
```

8.8 Information sur la copie de travail

Si vous désirez voir un aperçu de vos modification, utilisez la comamnde `svn status`.

Si vous utilisez la commande `svn status` à la racine de votre projet sans arguments, elle va détecter toutes les modifications (fichiers et arborescence) que vous avez faites.

Le format de sortie de la commande `svn status` comporte 6 colonnes de caractères suivis d'espaces et d'un nom de fichier ou répertoire. La première colonne renseigne l'état d'un fichier/répertoire et/ou son contenu. L'information est codée de la manière suivante

`svn status` possède aussi une commande "verbeuse" par l'intermèdiaire du commutateur `-verbose (-v)`

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svn status -v
?                                       ~svn_for_latex.tex
              169      150 yann         .
              169       31 yann         images
              169       31 yann         images/reseau.dia
              169       31 yann         images/reseau.eps
M             169      150 yann         svn_for_latex.tex
yann@biboo ~/latex/graoulug/svn_for_latex $
```

Dans mon cas, le `?` signifie qu'il s'agit d'un fichier non pris en compte par svn.

Si vous donnez un chamin spécifique à la comamnde `svn status`, vous obtenez l'information relative à cet unique item.

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svn status
?      ~svn_for_latex.tex
M      svn_for_latex.tex
yann@biboo ~/latex/graoulug/svn_for_latex $ svn status svn_for_latex.tex
M      svn_for_latex.tex
yann@biboo ~/latex/graoulug/svn_for_latex $
```

Il existe aussi l'option `-show-updates (-u)`, qui contacte le dépôt svn et ajoute les informations à propos des choses périmées.

```
$ svn status -u -v
M      *        44        23    sally     README
M               44        20    harry     bar.c
       *        44        35    harry     stuff/trout.c
D               44        19    ira       stuff/fish.c
A                0         ?     ?        stuff/things/bloo.h
Status against revision:   46
```

Notez les 2 asterisks (*) : En fait si vous deviez lancer un `svn update` à ce point, les fichiers modifiés seraient `README` et `trout.c`.

Examiner en détal les modifications, on utilise la commande `svn diff`. De plus cette commande formate la sortie d'une manière unifiée `diff` qui permet d'utiliser directement cetet sortie par le programme `patch`.

```
yann@biboo ~/latex/graoulug/svn_for_latex $ svn diff svn_for_latex.tex
Index: svn_for_latex.tex
===================================================================
--- svn_for_latex.tex   (révision 170)
+++ svn_for_latex.tex   (copie de travail)
@@ -655,6 +655,22 @@
 M      svn_for_latex.tex
test
+
+Notez les 2 asterisks (*): En fait si vous deviez lancer un \texttt{svn update} à ce po
+
+Examiner en détal les modificationsAnother way to examine your changes is with the svn
+
 Maintenant, où que vous soyez (travail ou maison), un simple \verb#svn update# vous per

 \section{Gestion des utilisateurs}
yann@biboo ~/latex/graoulug/svn_for_latex $
```

8.9 Revenir en arrière

Si des modifications ne sont pas correctes, il est possible de revenir à une version précédente à l'aide de la commande `svn revert`. Ceci est aussi valable pour l'effacement accidentel d'un fichier.

Maintenant, où que vous soyez (travail ou maison), un simple `svn update` vous permet d'avoir toujours la dernière version de vos documents/projets. Bien sur, il ne faut pas oublier de faire un `svn commit` avant de quitter son poste.

8.10 Résoudre les conflits de la copie de travail après une mise à jour

La commande `svn status -u` peut prédire les conflits. Par exemple voici le résultat de votre commande `svn update` :

```
$ svn update
U  INSTALL
G  README
C  bar.c
Updated to revision 46.
```

Les fichiers marqués `U` et `G` ne sont pas concernés. Ces fichiers sont traités convenablement par le dépôt. Les fichiers marqués d'un `U` ne contenait pas de modification locale et ont été mis à jour avec les modifications du dépôt. La lettre `G` signifie `merGed` (fusionné, mélangé), ce qui signifie que le fichier local possédait desz modification, mais que les modifications qui venaient du dépôt ne rentraient pas en conflit avec les modifications locales.

Par contre la lettre `C` représernte un conflit. Cela signifie que des changements sur le dépôt rentrent en conflit avec les votres. Il va falloir choisir manuellement les modifications à apporter.

Quand un conflit apparaît, il y a 3 choses qui peuvent vous aider à résoudre le conflit :

- ▷ Subversion affiche un `C` pendant la mise à jour, et se souvient que ce fichier est en conflit.
- ▷ Si Subversion considère que les fichiers peuvent être fusionnés il place des chaînes de marqueurs de conflit (chaîne texte) pour délimiter et mettre en valeur les zones de conflit.
- ▷ Pour chaque fichier en conflit, Subversion place 3 fichiers supplémentaires non versionnés sur votre copie de travail :
  - ▷ `filename.mine` Il s'agit de votre fichier identique à votre copie de travail avant la mie à jour, sans marqueurs de conflit. Ce fichier possède seulement *vos* dernières modifications. (Si les fichiers sont considérés comme fusionnables, ce fichier `mime` n'est pas créé.)
  - ▷ `filename.rOLDREV` Il s'agit du fichier de révision (BASE revision) avant la mise à jour de votre copie de travail. C'est à dire la dernière révision avant vos modifications.
  - ▷ `filename.rNEWREV` C'est le fichier reçu du dépôt par votre client subversion lors de la dernière mise à jour. Ce fichier correspond à la dernière révision du dépôt (HEAD revision).

  Ici `OLDREV` est le numéro de révision du fichier dans votre répertoire `.svn` et`NEWREV`, le numéro de révision de la tête (HEAD) du dépôt.

8.11 Création d'une "release" d'un document

Lorsque vous avez bien travailler et que vous considérer qu'il est fini (du moins à un moment précis), il peut être intéressant de créer une archive "release" de de document sans l'architecture `svn` (les répertoires `.svn` cachés dans l'arborescence).

Bien sur ce type de manipulation est bien plus utilisée dans le cadre d'un projet de programmation. Mais cela peut être utile dans le cas d'un manuel d'utilisation que vous mettez sans cesse à jour et pour lequel vous sorté des versions à télécharger.

La commande `svn export` va nous permettre de réaliser cela.

```
Crée une copie non versionnée d'une arborescence.
usage : 1. export [-r REV] URL[@PEGREV] [CHEMIN]
        2. export [-r REV] CHEMIN1[@PEGREV] [CHEMIN2]
```

```
  1. Exporte une arborescence propre à partir du dépét URL, à la révision
```

```
    REV si précisée ou HEAD sinon, vers CHEMIN. Si CHEMIN n'est pas donné
    le dernier composant de l'URL est pris comme nom de répertoire local.

  2. Exporte une arborescence propre à partir de la copie de travail
     CHEMIN1 à la révision REV si précisée ou WORKING sinon, vers CHEMIN2.
     Si CHEMIN2 est omis, le dernier composant de CHEMIN1 est utilisé
     comme nom de répertoire local. Si REV n'est pas précisé, toutes les
     modifications locales sont conservées. Les fichiers hors du
     gestionnaire de version ne sont pas copiés.
```

Nous allons donc créer dans un répertoire un copie non versionnée (mais avec le choix de de la version). Ensuite on pourra créer un tarball de ce répertoire pour diffusion.

```
$ svn export svn+ssh://login@machine/home/yann/stockage/svn_repository/mon_projet projet
$ tar czf projet_0.1.tar.gz projet_0.1
```

Et voila voici une release de votre projet que vous pouvez diffuser.

9 Gestion des utilisateurs

Cette partie reprend le tutoriel très vbien fait disponible à l'adresse http://jay.bertrand.free.fr/blog/index.php?url=archives/27-Installer-Subversion-sous-Debian.html

La toute première chose à faire est de créer un nouveau repository (un endroit spécifique où seront stockées les données). On utilise pour cela la commande d'administration `svnadmin`.

```
# man svnadmin
# mkdir /var/lib/svn
# svnadmin create --fs-type fsfs /var/lib/svn
# ls /var/lib/svn
README.txt  conf  dav  db  format  hooks  locks
```

La commande `svnadmin create` lance la création des structures nécessaires au repository dans le répertoire `/var/lib/svn`. L'option `-fs-type` permet de créer le repository au format 'FSFS', un nouveau système de fichier propre à subversion plus performant que les bases de données BDB. Bon, on a un repository. Tentons d'y ajouter des choses.

```
# su simpleutilisateur
# cd; mkdir test; echo 'ceci est un test' > test/test.txt
# svn import test file:///var/lib/svn/test
svn: Can't create directory '/var/lib/svn/db/transactions/0-1.txn': Permission denied
svn: Your commit message was left in a temporary file:
svn:    'svn-commit.tmp'
```

C'est tout à fait normal. le propriétaire des fichiers est root ! Et comme on utilise le protocole 'file ://' de subversion, on utilise par conséquence le système de permission sur les fichiers de linux. Ici, on a `rwx` pour root et `r_x` pour les autres donc mon simpleutilisateur ne peut pas écrire dans le repository.

```
# ls -al /var/lib/svn
total 36
drwxr-xr-x   7 root root 4096 Oct 25 12:04 .
drwxr-xr-x  16 root root 4096 Oct 20 14:34 ..
-rw-r--r--   1 root root  379 Oct 25 12:04 README.txt
drwxr-xr-x   2 root root 4096 Oct 25 12:04 conf
drwxr-xr-x   2 root root 4096 Oct 25 12:04 dav
drwxr-sr-x   5 root root 4096 Oct 25 12:04 db
-r--r--r--   1 root root    2 Oct 25 12:04 format
drwxr-xr-x   2 root root 4096 Oct 25 12:04 hooks
drwxr-xr-x   2 root root 4096 Oct 25 12:04 locks
```

Pour remédier à cela, on va créer un utilisateur et un groupe dédiés à svn et leur affecter le repository.

```
# addgroup svn
Adding group ‘svn’ (1000)...
Done.
# adduser --no-create-home --system --ingroup svn svn
Adding system user ‘svn’...
Adding new user ‘svn’ (103) with group ‘svn’.
Not creating home directory.
# chown -R svn.svn /var/lib/svn
# chmod -R g+w /var/lib/svn
```

Cette commande indique que tous les membres du groupe ont les droit d'écriture sur l'arborescence svn.

```
# ls -al /var/lib/svn
drwxrwxr-x   7 svn  svn  4096 Oct 25 12:04 .
drwxr-xr-x  16 root root 4096 Oct 20 14:34 ..
-rw-rw-r--   1 svn  svn   379 Oct 25 12:04 README.txt
drwxrwxr-x   2 svn  svn  4096 Oct 25 12:04 conf
drwxrwxr-x   2 svn  svn  4096 Oct 25 12:04 dav
drwxrwsr-x   5 svn  svn  4096 Oct 25 12:04 db
-r--rw-r--   1 svn  svn     2 Oct 25 12:04 format
drwxrwxr-x   2 svn  svn  4096 Oct 25 12:04 hooks
drwxrwxr-x   2 svn  svn  4096 Oct 25 12:04 locks
```

Maintenant, l'utilisateur et/ou le groupe svn peuvent accéder au repository. Démonstration.

```
# adduser simpleutilisateur svn
```

Cette commande permet d'ajouter un utilisateur à un groupe.

```
# su simpleutilisateur
# cd ~
# svn import test file:///var/lib/svn/test
Skipped 'test/.svn'
Adding         test/test.txt

Committed revision 1.
```

Tout utilisateur appartenant au groupe svn peut désormais accéder au repository en local depuis la machine.

Mais tout cela reste faisable dans le cadre de l'utilisation de ssh comme précédemment.

10 Gestion de droits des groupes et projets

Tout cela est très bien, mais dans notre cas, pour l'instant, tous les utilisateurs qui se trouvent dans le groupe `svn` ont accès à tous les projets stockés dans le dépôt.

Cela n'est pas forcément une bonne chose, et il faudrait créer des groupes d'utilisateurs et donner des droits à ces groupes/utilisateurs sur certains projets. Tout cecien conservant notre accès via `ssh`.

10.1 Configuration de `svnserve.conf`

Pour cela nous allons configurer notre dépot grâce aux fichiers `svnserve.conf` et `authz` qui se trouvent dans le répertoire `conf` à la racine de votre dépot.

Pour plus d'information sur la configuration du fichier `svnserve.conf` je vous renvoie à la page http://svnbook.red-bean.com/en/1.4/svn-book.html#svn.serverconfig.svnserve.auth qui donne tous les détails.

```
[general]
anon-access = none #ne donne aucun droits aux personnes non authentifiées
auth-access = write #donne les droits lecture/ecriture aux personnes authentifiées

# la suite est inutile dans notre cas, cas c'est ssh qui s'occupe de l'autentification
password-db = passwd #les mots de passe des utilisateurs sont stockés dans le fichier co

#la gestion des autorisation est stockée dans le fichier  conf/authz
authz-db = authz
```

10.2 Configuration de `authz`

Il convient alors de créer un compte pour chaque utilisateur (accès ssh) et de remplir le fichier `conf/authz` pour gérer les droits d'accès. Voici un exemple de fichier `authz`. Pour plus d'information sur la configuration je vous renvoi à l'adresse http://svnbook.red-bean.com/en/1.4/svn-book.html#svn.serverconfig.pathbasedauthz.

```
[groups]
group_admin = yann #création d'un groupe administrateurs
group_these = yann, regis

[/]
#@group_admin = rw #on donne tous les droits sur tout le dépot aux admins
yann = rw # on donne tous les droits à l'utilisateur yann
* =  # aucun droit pour les autres

[/these_regis]
@group_these = rw # on donne tous les droits
* = # aucun droit pour les autres
```

Voila, c'est un exemple minimal mais qui fonctionne.

11 Ce que cet article ne traite pas

▷ La visualisation graphique du dépôt
▷ L'utilisation de `svn revert`

12 Conclusion

C'en est fini de cet article, toutes remarques et corrections sont les bienvenues à l'adresse morere@univ-metz.fr