

ASR1 - Déroulement TD n°5

Semestre 1
Département Informatique
IUT Bordeaux 1

13 octobre 2010

1 Subversion

1.1 Introduction

Subversion (*svn*) est un outil pour le développement collaboratif qui permet la gestion de sources et le contrôle de versions. Il existe un grand nombre de logiciels du même type (GNU Arch, Bitkeeper, Git, Supersversion, etc).

Ce type de programmes a plusieurs fonctions, notamment :

- permettre à des utilisateurs distincts et souvent distants de travailler ensemble sur les mêmes fichiers (développement concurrent avec gestion des conflits).
- 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.

SVN est le successeur du logiciel de gestion de versions CVS. Il s'agit de la réimplémentation complète de CVS, suivant le même modèle éprouvé, mais corrigeant l'essentiel de ses défauts :

- les commits sont atomiques
- répertoires et méta-données sont versionnés
- `svn status`, `svn diff` et `svn revert` sont des opérations déconnectées
- une gestion extrêmement simple et efficace des étiquettes et des branches
- commit par révision (ou `changeSet`), et non par fichier séparément

Quelques avantages de subversion :

- il est multi-plateforme ;
- 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.

1.2 Notions générales

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.

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.

Création d'un dépôt :

La création d'un dépôt se fait à l'aide de la commande `svnadmin` :

```
svnadmin create REPOS_PATH
```

Par exemple :

```
svnadmin create /net/Bibliotheque/test-svn/REPOS-SVN
```

Attention la création d'un dépôt ne se fait qu'une seule fois.

Le contenu du dépôt ne doit jamais être modifié à la main. Il est mis à jour à l'aide des commandes `svn` effectuées depuis vos espaces de travail personnels.

Pratique

- Observez le contenu du dépôt SVN créé dans `/net/Bibliotheque/test-svn/` par vos enseignants.
- Lisez le fichier `README.txt`

Projets Au sein d'un dépôt se trouvent un ou plusieurs projets.

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.

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.

1.3 Opérations

Syntaxe des commandes Une ressource d'un dépôt (fichiers ou répertoire) est toujours désignée par une URL (`http://...`, `file://...`). Une ressource locale est désignée par un chemin. La syntaxe de la commande `svn` est :

```
svn [options] opération [chemins_locaux] [URL] [options]
```

Chaque opération qui modifie le contenu du dépôt doit être accompagnée d'un commentaire justificatif. En l'absence de l'option `-m`, `svn` ouvre un éditeur pour acquérir le commentaire. Pensez donc à utiliser cette option pour chaque modification

import L'import est l'opération qui 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. La commande `svn` relative est :

```
svn import <chemin_du_projet> <url_du_projet> -m "commentaire"
```

Attention : `url_du_projet` = `url_du_dépôt/nom_du_projet`

Par exemple :

```
svn import test1/ file:///net/Bibliotheque/test-svn/REPOS-SVN/test1 -m "ajout initial du projet test1"
```

```
svn importstage_c++ file:///net/Bibliotheque/test-svn/REPOS-SVN/ -m "import du projet stage_c++"
```

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.

```
svn checkout <url_du_repository/projet> <chemin(+nom)_de_la_copie_locale>
```

checkout peut être abrégé co

Attention, seule la copie de travail récupérée à l'aide d'un checkout est versionnée. Si vous êtes l'auteur initial du projet, le répertoire que vous avez utilisé pour **importer** le projet n'est pas une copie de travail !

Pratique

- Placez-vous dans le répertoire dédié au module ASR1-Systeme.
- Créez un répertoire `test-svn` temporaire dans lequel vous travaillerez (vous pourrez le supprimer en fin de session)
- A l'aide de la commande `svn` récupérez une copie de travail du projet `stage_c++` depuis le dépôt `/net/Bibliotheques/test_svn/REPOS-SVN/`.
- Vous devez voir apparaître la liste des répertoires et fichiers récupérés depuis la base svn précédés d'un **A** indiquant un *ajout*.

Si vous voyez apparaître dans la liste les répertoires `rep1`, `rep2` et `rep3`; vous n'avez pas récupéré le seul projet `stage_C++`, mais aussi `test1`. Effacez votre copie locale, corrigez votre URL et recommencez.

- Placez-vous dans votre copie de travail et listez le contenu. Essayez l'option `-a`. Que constatez vous ?
- Consulter le contenu du fichier `entries`.

```
svn co file:///net/Bibliotheque/test-svn/REPOS-SVN/stage_c++ copie_locale_stage_c++
```

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.

```
svn update
```

Il est également possible de mettre à jour sa copie à une version antérieure :

```
svn up -r num_version
```

Lorsque vous travaillez sur un projet, cette commande est un prérequis indispensable avant tout travail de votre part. Elle vous permet de travailler sur la dernière version et de tenir compte de l'avancée de vos camarades.

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. A 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.

Attention : les opération update et commit prennent en compte le répertoire dans lequel vous vous trouvez... Pensez à vous placer à la racine de la copie locale pour récupérer une mise à jour de tout le projet.

add, rm et mv Il est bien sûr possible de faire évoluer le projet en ajoutant, supprimant ou renommant des fichiers/répertoires. Lorsque l'opération est impossible (fichier déjà existant par exemple) `svn` vous donne un message d'erreur explicite. Pensez à bien les lire.

Ces opérations doivent être suivies d'un commit pour permettre les modifications effectives dans le dépôt.

```
svn add repertoire
svn ci -m "ajout de repertoire"
svn rm fichier
svn ci -m "retrait de fichier du projet"
svn mv repertoire/fichier_A destination
svn ci -m "déplacement de fichier dans destination"
```

Pratique

Dans votre copie de travail :

- Faites une mise à jour.
- Créez un nouveau fichier de contenu quelconque dans le répertoire `creation`.
- Ajoutez ce fichier à la base et faites un commit pour mettre à jour le dépôt (n'oubliez pas de préciser le message de commit).
- Faites une mise à jour de votre copie locale (toute la copie locale).
- Renommez-le et mettez la base svn à jour.
- Modifiez votre fichier. Mettez la base à jour.
- Supprimez votre fichier. Mettez la base à jour.
- Faites à nouveau une mise à jour.

1.4 Conflits et résolution.

Malgré de bonnes précautions de travail (mise à jour régulière, update avant tout commit...), il peut arriver que plusieurs personnes modifient un même fichier en même temps.

- Dans le meilleur des cas, les modifications ont lieu sur deux parties du fichier différentes, et svn est capable de fusionner proprement les changements.
- Dans le pire des cas, les développeurs ont modifiés des lignes identiques, et svn ne peut prendre de décision pour la fusion des modifications. Cela entraîne un conflit et un status particulier pour le fichier.

Au prochain update effectué par un utilisateur, svn signale le problème et demande de le résoudre. Le développeur prend en charge la résolution des modifications "à la main", puis signale le problème résolu à svn :

```
svn resolved fichier
svn ci -m "resolution des conflits sur fichier"
```

En cas de conflit, la copie locale du fichier comprend, entre chevrons, les différentes versions proposées. Des fichiers temporaires sont ajoutés avec les versions complètes "commitées" par les utilisateurs. Ces fichiers sont supprimés de la base après résolution du conflit.

1.5 États des fichiers, différences avec la base

Dans la copie de travail un fichier peut être dans un des 4 cas suivants :

1. Localement non modifié, et inchangé dans le dépôt : non concerné par commit ni par update
2. Localement modifié et inchangé dans le dépôt : il peut être enregistré sans problème dans le dépôt par commit et n'est pas changé par un update
3. Localement non modifié et changé dans le dépôt : commit n'a pas d'effet, mais update le modifie pour le mettre à jour

4. Localement modifié et changé dans le dépôt : commit échoue, update tente de fusionner les modifications et les changements, ou bien laisse l'utilisateur régler le conflit

Il est possible de voir le status d'un fichier grâce à la commande :

```
svn status
```

Quelques états :

- Rien ⇒ pas de modification
- A : ajouté localement (sera ajouté au dépôt par un ci)
- D : supprimé localement (sera supprimé du dépôt par un ci)
- M : modifié localement
- C : en conflit avec les mise à jour reçues du dépôt.
- ? : présent localement mais absent du dépôt.
- ! : Absent localement mais présent dans le dépôt.

1.6 Messages et versions

Pour connaître le numéro de version de la copie locale :

```
svnversion
```

106M ⇒ la version 106, comporte des modifications non commitées

107 ⇒ version 107, pas de modifications.

Pour connaître l'avancement du projet et la teneur des modifications apportées, il est possible de consulter les messages de commit :

```
svn log [fichier]
```

```
svn log -v [fichier]
```

```
svn -r 100 log [fichier]
```

Lister les différences :

```
svn diff [fichier]
```

```
svn diff -r 100:107 fichier
```

Pratique

- Quelle est la version actuelle de votre copie de travail, du dépôt ?
- Regardez les commentaires de commit.
- Quelles sont les commentaires de la première version, de la dernière, de la version 18 ?
- Listez les différences entre deux versions.

1.7 svn à distance

Il est possible d'utiliser svn sur un ordinateur distant par le biais de ssh :

```
svn co svn+ssh://login@info-ssh1.iut.u-bordeaux1.fr/.../REPOS-SVN/projetnom_repertoire_local
```